Declaration

This dissertation is the result of my own work and includes nothing, which is the outcome of work done in collaboration except where specifically indicated in the text. It has not been previously submitted, in part or whole, to any university of institution for any degree, diploma or other qualification.

In accordance with the Department of Engineering guidelines, this thesis is does not exceed 80,000 words and it contains less than 150 figures.

Signed:_____

Date:_____

Name: Mohammad Masbah Uddin Chowdhury

University of Hull, Hull, United Kingdom

Abstract

The work described in this thesis, concerns developments to analytical microfluidic Lab-On-Chip platform originally developed by Prof Pamme's research group at the University of Hull. This work aims to move away from traditional laboratory analysis system towards a more effective system design which is fully automated and therefore potentially deployable in applications such as point of care medical diagnosis. The microfluidic chip platform comprises an external permanent magnet and chip with multiple parallel reagent streams through which magnetic micro-particles pass in sequence. These streams may include particles, analyte, fluorescent labels and wash solutions; together they facilitate an on-chip multi-step analytical procedure. Analyte concentration is measured via florescent intensity of the exiting micro-particles. This has previously been experimentally proven for more than one analytical procedure. The work described here has addressed a couple of issues which needed improvement, specifically optimizing the magnetic field and automating the measurement process. These topics are related by the fact that an optimal field will reduce anomalies such as aggregated particles which may degrade automated measurements.

For this system, the optimal magnetic field is homogeneous gradient of sufficient strength to pull the particles across the width of the device during fluid transit of its length. To optimise the magnetic field, COMSOL (a Multiphysics simulation program) was used to evaluate a number of multiple magnet configurations and demonstrate an improved field profile. The simulation approach was validated against experimental data for the original single-magnet design.

To analyse the results automatically, a software tool has been developed using C++ which takes image files generated during an experiment and outputs a calibration curve or specific measurement result. The process involves detection of the particles (using image segmentation) and object tracking. The intensity measurement follows the same procedure as the original manual approach, facilitating comparison, but also includes analysis of particle motion behaviour to allow automatic rejection of data from anomalous particles (*e.g.* stuck particles). For image segmentation a novel texture based technique called Temporal- Adaptive Median Binary Pattern (T-AMBP) combining with Three Frame Difference method to model the background for representing the foreground was proposed. This proposed approached is based on previously developed Adaptive Median Binary Pattern (AMBP) and Gaussian Mixture Model (GMM) approach for image segmentation. The proposed method successfully detects micro-particles even when they have very low fluorescent intensity, while most of the previous approaches failed and is more robust to noise and artefacts. For tracking the micro-particles, we proposed a novel algorithm called "Hybrid Meanshift", which combines Meanshift, Histogram of oriented gradients (HOG) matching and optical flow techniques. Kalman filter was also combined with it to make the tracking robust.

The processing of an experimental data set for generating a calibration curve, getting effectively the same results in less than 5 minutes was demonstrated, without needing experimental experience, compared with at least 2 hours work by an experienced experimenter using the manual approach.

Acknowledgement

At first I would like to thanks to my advisor Dr Ian M Bell who has been very kind and supportive throughout my PhD. Without his support and kind supervision nothing was possible. I also would like to thank my co-supervisor Prof Nicole Pamme for her useful advice in every meeting of research discussion.

I am extremely grateful to The University of Hull for giving me an opportunity for working on this PhD project.

Thank you so much Dr Etienne Joly, Dr Martin Vojtisek, Dr Chayakom and Dr Mark Tarn, who are not only my friends during the work, but also true friends outside the lab.

Finally, I express my deep gratitude to my parents and brother for always standing beside me, supporting me for everything. Special thanks to Dr Fei Gao for her support throughout my entire UK life.

Finally, Alhamdulillah, all praises to Allah for the strengths and His blessing in completing this thesis.

List of Abbreviations

B_s	Residual magnetic flux density
d	Average diameter of the magnetic nanoparticles
ϵ	Permittivity of the fluid medium
F_m	Magnetic buoyancy force
F_w	Wall-induced repulsive force
H	Magnetic field
H_{demag}	Particles self-demagnetization magnetic field
H_{in}	Effective magnetic field inside particle
k_B	Boltzmann constant
M	Magnetization
M_f	Effective magnetization of the Ferro fluid
M_p	Effective magnetization of the particle
M_r	Remanent magnetization of the magnet
M_s	Residual magnetization of the permanent magnet
M_d	Saturation moment of the magnetic nanoparticles
M_{sat}	Saturation magnetization of the Ferro fluid
Q	Flow rate
r_p	Instantaneous position of the centre of a particle
r_0	Initial position of the particle centre
T	Temperature
t	Time
U_p	Particle velocity

U_m	Magnetophoretic particle velocity
U_f	Flow velocity
μ_0	Permeability of free space
μ_m	Dynamic viscosity of the solution medium
V_p	Volume of the particle
χ	Susceptibility
ϕ	Volume fraction of magnetic nanoparticles
δ	Smallest particle centre to wall separation distance
η	Kinetic viscosity of Ferro fluids
$\oplus$	Dilation operator
$\ominus$	Erosion operator
μ_k	Mean for the weight in GMM
${\sigma_k}^2$	Variance for the weight in GMM
AMBP	Adaptive Median Binary Pattern
BG	Background
FD	Frame Difference
FG	Foreground
GMM	Gaussian Mixture Model
HOG	Histogram of Oriented Gradients
KDE	Kernel Density Estimation
LBP	Local Binary Pattern
LOC	Lab On Chip
MBP	Median Binary Pattern
PETS	Data sets for object detection and tracking
ROI	Region of Interest

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Microfluidics is a relatively new scientific discipline involving manipulation of small amounts of liquids for various analyses with a wide range of applications in medicine, environmental monitoring and chemical analysis. Research on microfluidic devices and their fabrication with micromechanics technologies has about five decades of history [1]. From the early 1980s, research activities on microfluidics has expanded considerably and it has become a very popular research topic [2]. However, there are still many challenges to be addressed. This is a multidisciplinary field, requiring input from many different scientific backgrounds. In order to make successful progress within the field, a development team needs to have knowledge of mechanical engineering, material science, surface chemistry and intermolecular interactions, chemical and biochemical reactions and microelectronics.

Microfluidic techniques provide many different advantages for analytical applications, multivariate studies or studying processes at small scale. Moreover, it has some other clear advantages such as small footprint, small volumes of reagents requirement, potential lower reaction times, which are extremely valuable for any analytical application. A typical microfluidic reactor might contain volumes as low as a few micro-litres, nano-litres or lower (10^{-9} to 10^{-18} litres) [3], which obviously reduces the reagent consumption. This can be of particular interest in biochemical or medical applications, where it is difficult to obtain large amount of samples or reagents are very expensive. An additional advantage for low volume reactors is low

amount of waste. Together with small footprints, it is easier to contain and isolate the reactants, making them safer even for application such as radiochemistry. Small volumes also mean fewer requirements on storage, pumping. Moreover, it offers the possibility of other types of liquid manipulation beyond traditional mechanical pumping. Liquid could be transported in the device by syringe pumps, but also by electro-kinetic processes.

Furthermore, miniaturisation offers the possibility of massively multiplex processes, like DNA arrays or device used for sequencing. The key for providing results in a short period of time is to collect signals from millions different sources and with the aid of computer processing resolves such signals [4]. This is a considerable step from traditional process, like use of 96-well plates. For these reasons, microfluidics and microfluidic devices have had a great influence on many branches of chemical and biological analysis, including optics and information technologies [3].

Stanford University, USA and IBM are considered as pioneers of the implementation different microfluidic devices. For example Stanford University used microfluidic chips for doing gas chromatography and IBM developed the nozzles of inkjet printers in 1979 [5].

1.1 Lab on Chip

The small size of microfluidic devices allows easy integration with other components, serving as sensors and actuators, resulting in a complex system capable of performing various processes at the same time within one device. A typical example would be the systems labelled as Lab-On-a-Chip or LOC. The aim of such devices is to include processes such as sample preparation, any required reaction and detection on one device. In such cases, the devices, which are predominantly aimed for medical applications, could be deployed anywhere and remove necessities of sending samples to specialised laboratories. The overall benefits offered by LOC include: reduced amount of reagent which reduces the costs, elimination of labour intensive steps, high purity and decreased processing time compared to conventional methods.

In the early 1960s, when several research groups started working on miniaturized silicon sensors, the idea for developing LOC devices first appeared. Research on LOC techniques became more prominent when Defence Advanced Research Projects Agency, USA (DARPA) invested in research in portable biochemical warfare agent detection systems for the military. The first commercial use was for different life science applications [6]. An example of the newest generation of LOC systems is a miniaturized chip for isolating the rare circulating tumour cells in cancer patients [7]. Initially glass and silicon were the most commonly used fabrication materials for Lab-On-Chip devices, but the use of soft lithography techniques made it easier to use various polymers like PolyDiMethylSiloxane (PDMS). Beside PDMS type polymers, glass and silicon are still used for many systems where PDMS is not suitable due to temperature or chemical in compatibility.

There are many of microchip fabrication process are available, but most commonly used are Photo lithography, UltraViolet (UV) laser photo ablation, Powder blasting for glass substrates, sawing *etc.* After the introduction of photo-lithographic techniques for the fabrication of chemical and biochemical micro devices [8], the number of applications of using different microfluidic chip has increased exponentially [9]. Each of the process contains more than one steps [10]. Depending on the availability of resources and requirement(s) of the experiment, any of the process needs to use for microchip fabrication.

An example of a typical LOC, a microfluidic chemostat (Chemical environment is static) device with numerous pneumatic valves fabricated using PDMS is shown in Figure 1.1:

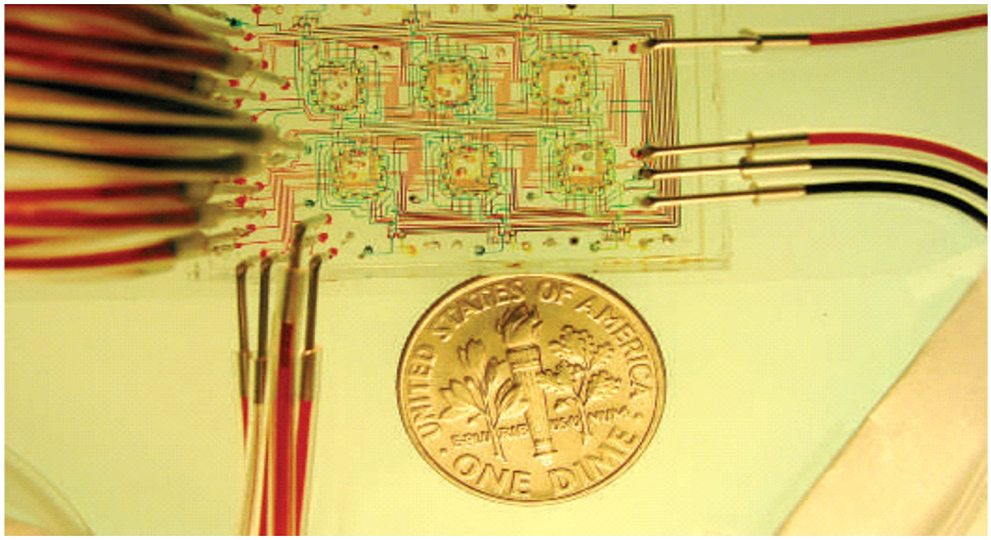


Figure 1.1: A microfluidic chemostat with pneumatic valve [3].

1.2 Magnetophoresis systems

Magnetophoresis is an important branch of LOC microfluidic experiments, where magnetism and microfluidic are fused together for various applications. Separately magnetism and microfluidics are not so new concepts, but they have been combined together only for the last two decades and became a significant research area [1, 11]. For this purpose, a wide range of magnets from simple and cheap external permanent magnets to sophisticated micro-electromagnets are now in use [5].

In numerous microfluidic applications, such as capillary electrophoretic separations, electro-osmotic pumping and dielectrophoretic trapping, electric fields are used [12]. For such experiments it has been proven that use of external permanent magnets do not have any effect on surface charges, pH, ionic concentrations or temperature, as these properties or outputs are generally not affected by magnetic fields [13].

In microfluidic processes magnetic forces are mainly used for transporting and sorting magnetic particles. In this way bio-molecules can be isolated from a sample by attaching them to small magnetic particles, which also facilitates multistep reaction within a very small microfluidic chip [13]. These experiments are widely known

as magnetophoresis.

The work described in this thesis concerns developments and improvements to magnetophoresis LOC devices and experiments based on works by Prof. Pamme and her research group. Most of these magnetophoresis experiments employ multi regent stream system analysis in which several lamina fluid streams are injected into the microfluidic chip with different chemical properties. As magnetic particles pass through the different streams reactions take place on the surface of the particle in different stages. The following sections provide an overview of the operating principles and structure of the LOC devices of interest in this work. This will be followed by a discussion at the issues to be addressed.

Use of magnetic particles and an external magnetic field as a separation technique in a continuous flow microfluidic chip is now widely known as on-chip free-flow magnetophoretic separation and was introduced by Prof. Pamme *et al.* in 2006 [14, 13]. A typical microfluidic chip, based on the free flow magnetophoretic principle is shown in Figure 1.2. It was designed by Prof. Pamme (Department of Chemistry, University of Hull, UK) [15]. The system consists of a microfluidic chip, external permanent magnet, magnetic particles, fluorescent stream, buffer solution and solution of the target particle. The chip may contain several inlets and outlets depending on the application requirement. It is not necessary that the inlets and outlets are equal in number or have the same shape, it can be designed any way depending on the experimental requirements. The materials used for the device are glass, silicon or polymer.

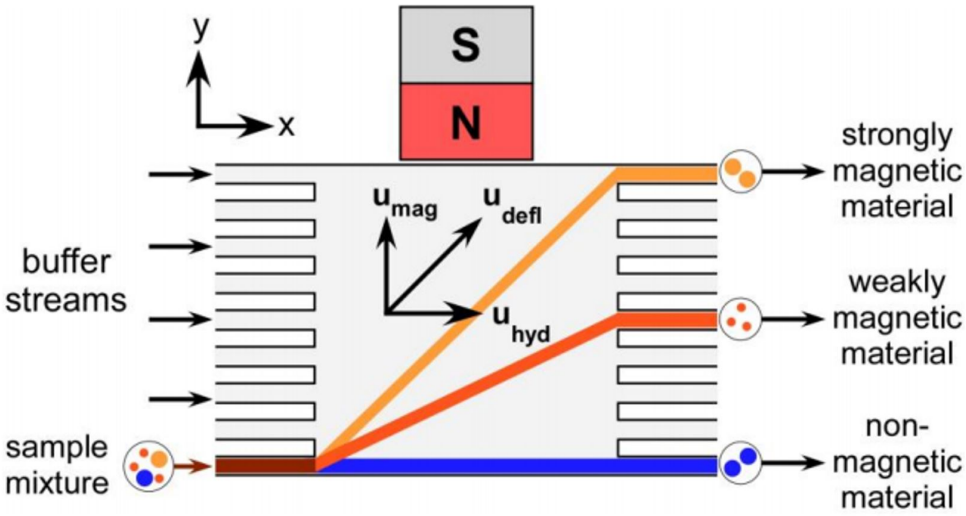


Figure 1.2: Principle of free-flow Magnetophoresis system for magnetic separation. Magnetic particle mixture is introduced into the microfluidic chamber and external magnetic field is applied perpendicular to the flow. For constant flow rate, particles trajectory depends on their size and magnetic susceptibility, which allows different particle types to exit the chip via different outlets. This way particles can be separated from each other [15].

One of the implementation of magnetophoresis principles on LOC experiment is magnetic separation. According to basic principle of magnetic separation shown in Figure 1.2 from the first inlet, a suspension of mixed particles is introduced into the chamber and a permanent magnet is placed at the opposite side of the chamber. Depending of the flow velocity, particle size, internal magnetic content of the particle, magnetic properties of the magnet and media, the particles would then travel across the chamber in both X and Y direction, where the Y axis is perpendicular to the flow (X axis). In such motion, particles would migrate through the separate streams. Using this technique, the magnetic and non-magnetic particles can be precisely separated. This separation happens due to the deflection of the magnetic particles from the direction of the laminar flow, depending on the magnetic susceptibility, size and velocity of the particles and lamina flow. This technique is useful not only for magnetic particle and non-magnetic particle separation, but also particle separation depending on their magnetic properties.

An advantage of this set-up is the possibility to monitor the particles in real-time, for example one could obtain data for kinetic studies in this manner. The advantage is again the ease of use and speed of generating the data.

In addition to magnetic separation, the set up shown in Figure 1.2 can be used to send a set of uniform particles through a sequence of laminar fluid streams. Due to the dimensions and flow velocities typically used in these chips, the flow is laminar and there is relatively little mixing between the individual reagent streams. Therefore, it can be assumed that all reagents are separate reaction media. This can be used to perform a sequence of reactions on the surface of the particles easily in continuous flow. Such procedure removes the necessity for manually transferring the particles between various reaction vessels for example.

The previously described magnetophoresis experiments were developed further by the research group of Prof. Pamme. Using the principle of magnetophoresis and similar experimental setup, additional functionality was added (See Figure 1.3). The inlets were divided into separate channel capable of delivering different reagents into the chamber of the device. Magnetic particles were introduced from inlet at the bottom of the chamber as before and deflected across the width of the chamber by a permanent magnet positioned at the opposite side of the chamber. As the particles travelled across the chamber, they moved through several different reagent streams, which stayed relatively unmixed due to laminar flow regime. In such a way reaction on the surface of the particles could be performed in continuous flow, in relatively simple and rapid manner [16].

As a proof of principle, streptavidin modified particles were sent through a buffer stream, a stream containing fluorescently labelled biotin and another buffer stream. The motion of particles across these streams resulted in increase of fluorescence of the particles, due to strong streptavidin-biotin interactions forming during the particles' residence in the biotin stream. Thus analyte concentration can be measured by detecting the level of fluorescence of the particles as they exit the chamber.

Later other reactions were employed such as DNA hybridisation [17] or antigen-antibody binding [18]. Moreover, additional reaction can be added and thus perform a 2-step reaction process in one continuous flow procedure.

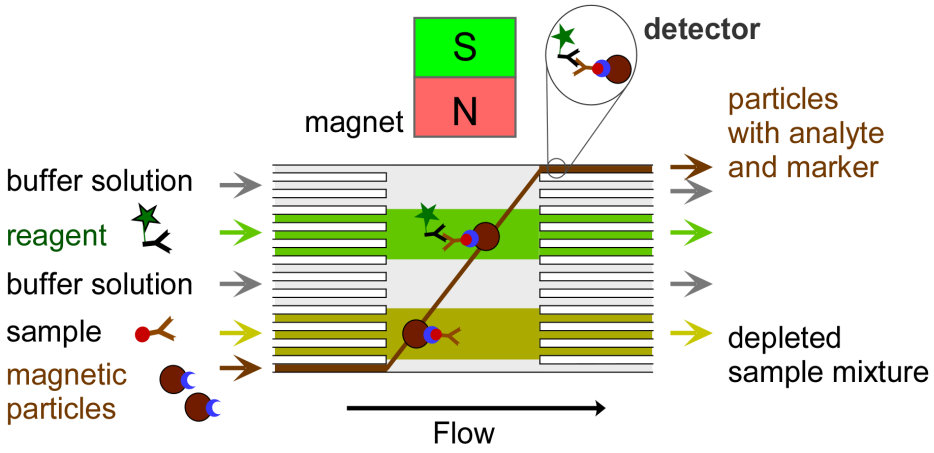


Figure 1.3: Principle of the multi-laminar flow platform. Alternating reagent and buffer streams are generated across a microfluidic chamber and functionalised magnetic particles are deflected through each stream, allowing multiple consecutive reactions to occur on the particle surfaces [16].

The chip shown in Figure 1.4 is an example microfluidic chip of a multi-laminar flow experiments. It was fabricated using glass, but it is also possible to use PDMS (PolyDiMethylSiloxane) instead of glass [19]. The reaction chamber of the microfluidic chip used during DNA hybridization in an experiment performed by Dr Martin Vojtisek in Prof. Pamme's research group was 22 μm in depth, 8 mm long, 3 mm wide and rectangular in shape and supported by pillars [17]. It had five different inlets supplying different solution into the chamber and two outlets to collect the waste. These chips are constructed from two glass plates placed one on the top of another and thermally bonded together - one of them contains the channels and the chamber and the other part is a plane symmetrical plate without chamber and channels.

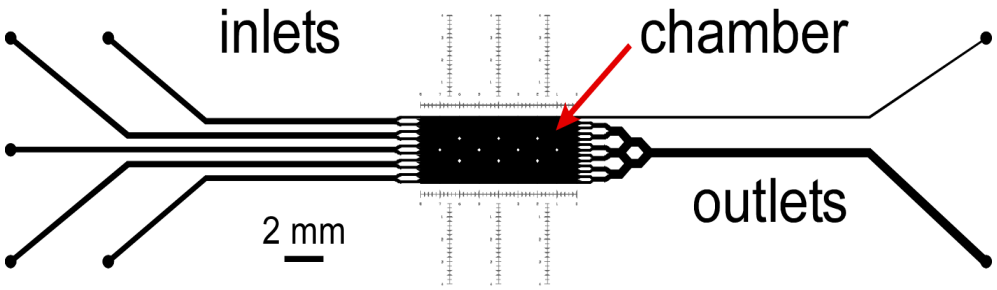


Figure 1.4: Microfluidic Chip designed by Prof. Pamme (NP59) in with 5 inlets and 2 outlets with 8 mm long chamber [20].

The magnetic field throughout the chamber was not uniform, being higher near to the place where the magnet is placed. Hydrodynamic flow also effects the movement of the magnetic particles. So, the observed flow velocity vector of the magnetic particles is the vector summation of magnetically induced flow vector and the applied hydrodynamic flow vector due to the lamina flow. This can be expressed as Equation (1.1).

$$\overrightarrow{U}_{obs} = \overrightarrow{U}_{hydr} + \overrightarrow{U}_{mag} \tag{1.1}$$

Considering all the conditions, the capillaries need to be connected to the chip depending on the experimental requirements as shown in the Figure 1.5.

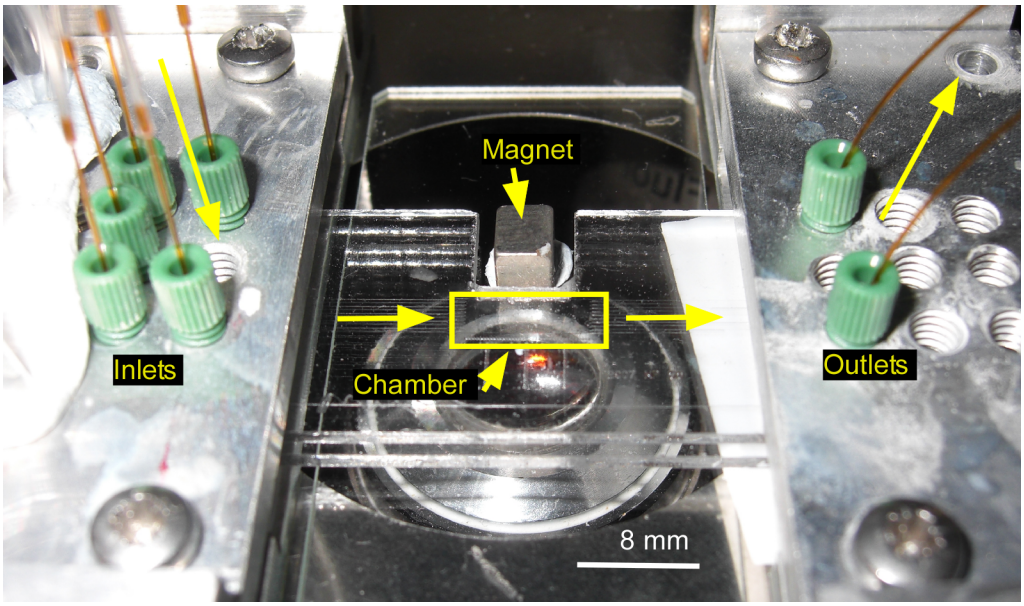


Figure 1.5: Photograph of microfluidic chip connected with capillaries and holders.

In this example experiment, described previously as multi-laminar flow experiments, all the inlet capillaries were connected to the gas-tight glass syringe (SGE, Supelco, USA) and then all of the syringes were fitted to a multi-syringe pump to pump by positive pressure. The capillaries from the outlets are then linked to the glass vial of waste solutions. After assembling the microchip with all the capillaries and holders placed on the stage of an inverted light and Nikon TE2000-U florescence microscope which is shown in Figure 1.6.

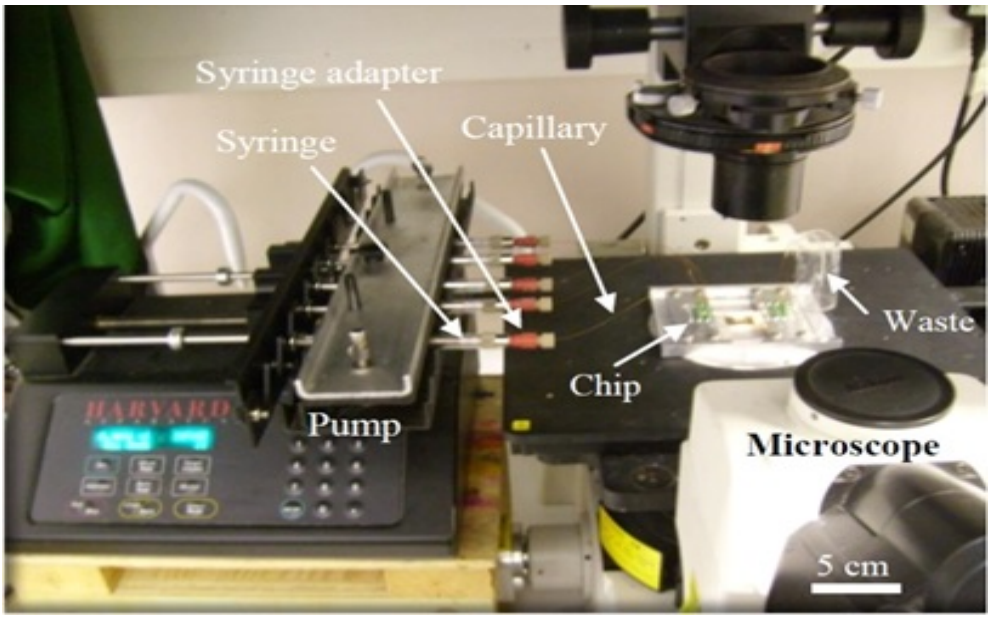


Figure 1.6: Typical experimental setup from Prof. Pamme's research group showing fabricated glass chip connected with tubing, microscope and image capturing camera.

Images and video sequences for the microfluidic experiments were recorded using a Charge-Coupled Device (CCD) camera. These images collected from the CCD camera are analysed to measure particle fluorescence and hence determine analyte concentration. So, these microfluidic experiments are done in two parts on-chip experiment and off-chip experiment. Data analysis of the experiment is known as off-chip experiment. For the current off-chip data analysis, the particles are manually detected and their fluorescence are measured mostly using ImageJ [21] image analysis software. In many cases their grey values are measured to determine the level of fluorescence. The issues to be considered

Particle intensity: Typical micro-particle itself does not have any significant intensity property. However, if the particle surface has been chemically activated, it may produce light by florescence or chemiluminescence. In such cases the light intensity from the micro-particle has a direct relationship with the concentration of the analyte, *i.e.* the intensity of the micro-particle is proportional to the concentration of the analyte. Low background fluorescence of particles is desirable in order to achieve high sensitivity. Instrumental setup could be further improved which would enable even lower detection limits.

Magnetic Force on the particle(s): Magnetic micro particle moves across the reaction chamber for the combinational effect of magnetic force and drag force on it. Considering constant drag force from the fluid flow, the magnetic field gradient is not equal over the whole reaction chamber. So different particles experience different amounts of magnetic force depending on the distance of the particle from the magnet.

Non-moving and stuck particle(s): Within the reaction chamber of the microfluidic chip there are several supporting pillar exits to keep the two parts of the chamber separate from each other. Frequently micro-particle(s) hit those pillars and become non-moving for next few frames or forever. Moreover, depending on the effective force on the particle, particles may stick to each other and act as one large particle.

Image Capture Tools: Cameras used for image capturing can be better. With the current set up, it produces noise with the presence of any external light and changes in any internal condition of the LOC experiments.

Diffusion: Despite maintaining laminar flow regime, diffusion between reagent streams is unavoidable. This has to be accounted for when designing the device. Moreover, diffusion in laminar flow also changes in image scenarios.

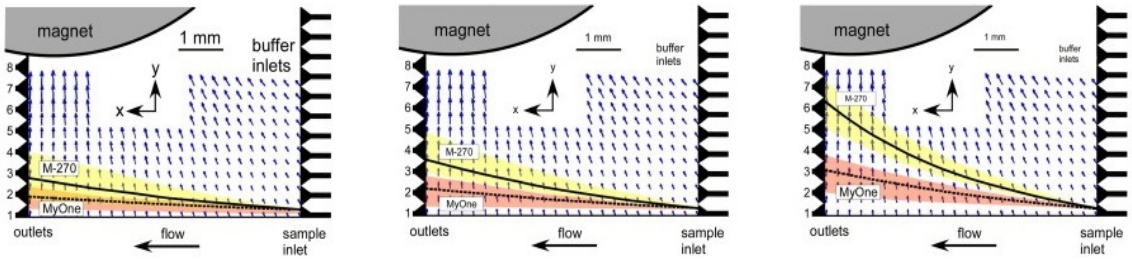
Occlusion and Clutter: There are several potential issues related to the nature of the experiment, such as differences in starting positions of particles which can result in dramatically different trajectories if the magnetic field is generated by a simple permanent magnet. This is because such field does not provide uniform and constant gradient of magnetic field.

1.3 Problems of the current system

There are several practical issues with this Lab-On-Chip systems and associated experiments which will be discussed in the following sections:

1.3.1 Temperature

The temperature of the system needs to be controlled accurately, because the resulting magnetophoretic motion of particles depends on viscosity of the medium considering that all other parameters being constant. Viscosity in turn can change dramatically with temperature. Nevertheless, this problem can be circumvented by using the devices in a regulated environment or using dedicated temperature regulating systems for the experiments. In Figure 1.7, particle trajectory for 5^oC, 20^oC and 50^oC temperature inside the reaction chamber is shown in Figure 1.7a, 1.7b and 1.7c respectively. Temperature control is relatively straightforward to achieve and was not addressed as part of this work.



(a) At temperature 5^oC. (b) At temperature 20^oC. (c) At temperature 50^oC.

Figure 1.7: Effect of Temperature of the magnetic particle flow [22].

1.3.2 Magnetic Field Profile

The principle of the magnetophoresis lies in characteristic motion of magnetic particles subjected to inhomogeneous magnetic field and other forces perpendicular to the magnetic force. Placing a permanent magnet at the side of a chamber of a microfluidic chip generates magnetic gradient across the width of the chamber, however the magnetic forces diminish rapidly with distance.

For practical reasons, the inlet where particles enter the chip is generally 200 - 300 μm wide. The magnetic force acting on particle therefore depends heavily where exactly it enters the chip, as even such relatively small distance makes ultimately large difference. Additionally, particles could be subjected to magnetic fields already

in the inlet channels and being stuck against the walls of the inlets. In short, the magnetic field should be designed so that it only provides magnetic forces in the region of interest and ideally the forces are independent on the distance from the magnet. Such magnetic field cannot however be obtained using regular shaped commonly available permanent magnets. Modelling techniques can therefore be employed in order to establish the required design of the magnet so that it can be custom made for use with the magnetophoretic chips. Developing custom shaped magnet for producing uniform magnetic field gradient over the microfluidic chip was a considerable part for this project.

1.3.3 Microfluidic result analysis

Using ImageJ for manually detecting and analysing the particle is a time consuming process and cannot be extended to process more samples. Furthermore, it is also dependent on subjective interpretation by the individual experimenter. This may make a difference to the results. As this current system is user dependent, the results are not reproducible. Therefore, it was urgent to develop an automatically image analysis software which could recognize and track the micro particles, analyse their movement and measure their light intensity. So developing an algorithm to detect the particles and analyse (including grey scale value, motion *etc.*) them was a part of this work.

1.4 Aims and Objectives of the research

One of the main targets for this project was to develop a miniaturised system using comprehensive computer aided design tools to facilitate efficient development of new Lab-On-a-Chip (LOC) systems. Beside this computational simulation for LOC another task for this project was to develop a system to make the analysis of the experiment faster and more reliable, efficient and reproducible. So the overall goals of this PhD project from engineering view are 1) carrying out computational simulation of the work to predict the optimized experimental conditions and 2)

developing software to analysing the results from Lab-On-Chip experiment automatically. These two goals are described in more detail in the following section, also relevant background will be discussed in more detail in later chapters.

1.4.1 Lab On Chip simulation for magnetic field optimization

In the LOC device(s) relevant to this study (Figure 1.3) particles enter the chamber at slightly different locations they may be anywhere within the cross-sectional area of the inlet channel. This will apply to any similar devices using the same general approach and is not unique to this design. Under these conditions, if the magnetic field gradient is not uniform, then the particles will not follow perfectly parallel paths. Therefore they may collide with one another. Furthermore, they may deviate sufficiently from the optimal path to hit the micro-pillar support structures. This may lead to the following problems:

1. Some particles might not reach the correct outlet/detector and be usable for analysis.

2. Automatic tracking is more difficult due to the possibility of particles crossing paths in close proximity and becoming indistinguishable after separation or aggregating to form a single combined particle, which would not be usable for analysis, but would have to be identified by the image processing as such.

3. Residence time in reagent streams may be variable, potentially introducing errors.

The result of these issues is fewer "good" particles available for result analysis and greater potential for image processing difficulties, particularly at lower analyte concentrations where particle intensity in low. Therefore, it was considered desirable to optimise the magnetic field to provide the best conditions for automatic tracking and measurement of the particles. A simulation approach was chosen because setting

up large numbers of experiments with different magnets arrangements would be too difficult and time consuming.

Computational simulation helps researchers to perform experiment with different conditions and predict results, which enables the researchers for reducing the numerical optimization steps [23]. Such simulation can be assisted by the use of commercial multiphysics simulation tools. There are numbers of commercial and open source computational software for microfluidics and magnetic field simulation, such as Finite Element Method Magnetics (FEMM) (www.femm.info), Ansys (www.ansys.com), COMSOL (uk.comsol.com), Electromagnetic Simulation(EMS) (www.emworks.com), MagNet (www.infolytica.com) *etc.* COMSOL Multiphysics allow users to perform microfluidics modelling and magnetic field simulation in three dimensions.

As mentioned earlier, unequal magnetic field gradient over the microfluidic chip makes the motion path of the micro particle differ from the desired one. Also magnetic field strength (*i.e.* field gradient) reduces sharply with the increase of distance from the magnet. So it was required to have custom shaped magnet placed for moving the particles from inlet to the desired outlet. To develop the custom shape magnet, the only way was to use simulation tool (COMSOL). So using COMSOL simulation tool to simulate magnetophoresis experiment with the influence of magnetic field from custom shaped magnet was one of the aims of this thesis.

1.4.2 Automatic particle measurement and tracking

It was required to develop an algorithm to detect micro-particle automatically. For detecting micro-particle automatically, a novel algorithm was be developed which can differentiate the micro-particles from the background and different noises. Developing an algorithm for automatically detecting micro-particles and measuring their light intensity associated with fluorescence or chemiluminescence was one of the main aims of this project.

Automatic object tracking has been widely used in many critical tasks, such

as surveillance systems including vehicle tracking [24], human recognition[25] *etc.* As the result of high demand for object tracking, many algorithms has been developed to deal with tracking object (such as vehicle registration number plate, human face, human body *etc.*) and tracking environment (such as dynamic background, illumination changes, shadows, moving camera *etc.*). However, there is no ready to use algorithm for tracking micro-particles used in microfluidic chip, which are all of similar colour, similar shape and low contrast difference with environment. Development of such algorithm will be a key part of this work.

1.5 Thesis Organization

This thesis was organised as follows:

Chapter 2 provides an overview of microfluidic flow and magnetic theories, which will be required for modelling simulation and magnetic field optimization.

Chapter 3 presents the work on simulation and optimizing magnetic fields affecting micro particle trajectories in the Lab-On-Chip devices.

Chapter 4 provides a comprehensive review for different background subtraction image processing techniques. The basics of background subtractions, its process and challenges will be introduced. Widely used background subtraction techniques and their implementations and drawbacks will be presented and their applicability to this work evaluated.

Chapter 5 provides brief description of proposed approach for detecting micro-particles and compares the results with other existing techniques. This proposed technique combines the newly developed texture segmentation technique with the widely used background modelling techniques: Gaussian Mixture Model and three frame difference. This will also provide an idea about the improvement achieved for background modelling and subtraction, over the current available systems for detecting micro-particles with very low concentration analyte.

Chapter 6 will present the principle of multiple object tracking. Challenges of multiple object tracking, steps of doing it and widely used object tracking algorithms

will be described here.

Chapter 7 will explain the proposed approach for micro-particle object tracking. This proposed approach combines the modified Meanshift object tracking technique with HOG, optical flow and Kalman filter for robust tracking. It will also demonstrate the results of tracking for micro particles used in microfluidic chips.

Chapter 8 will show the performance of our approach for analysing data sequences from magnetophoresis experiment, this will include the process of generating Error bar and calibration curves for those experiments using automated data analysis software. Also the time requirement and accuracy for this system will be discussed here.

Chapter 9 will provide overall conclusion of this whole project and future plan of this work.

1.6 Thesis Contribution

This thesis contributes a method for background modelling to detect micro-particles, tracking them and measuring their intensity to find the concentration of an unknown analyte. Also a simulation of a magnetophoresis experiment is represented here. All these works in this thesis can be summarized using following four sections:

Detecting Moving Object: A novel technique is proposed for detecting moving object(s) called "Texture based GMM combined with frame differencing". This object detector is able to detect object from a scenario where the objects are barely visible. Moreover, this technique is robust to sudden changes in lighting condition, as it uses texture for modelling the objects. This technique is developed mainly to detect micro-particles from magnetophoresis experiments, but it provides an efficient performance for detecting other moving objects as well.

Tracking Multiple Objects: After a moving micro-particle is detected first time, a foreground mask is generated for initiating tracking using proposed "Hybrid

Meanshift" combined with "Kalman filter". This proposed method is efficient for tracking featureless multiple micro-particles even they are moving by keeping very small distance with each other.

Measuring concentration of unknown analyte: The manual method for measuring concentration for any unknown analyte is time consuming and the result is not reproducible. An automated process has been developed, which can calculate the concentration of an unknown analyte within a very short period of time and the results are reproducible.

Simulating magnetophoresis and homogeneous magnetic field gradient: A DNA hybridization magnetophoresis experiment was simulated using COMSOL Multiphysics. This experiment was previously done in the Chemistry laboratory by Dr Martin Vojtisek. The magnetic flux distribution and field gradient over the reaction chamber for rectangular shaped magnet and Halbach array of magnet were investigated. Moreover, for generating homogeneous magnetic field gradient across the reaction chamber, combination of two magnets was simulated.

Chapter 2

MICROFLUIDIC THEORY FOR SIMULATION USING COMSOL MULTIPHYSICS

For simulating magnetophoresis microfluidic experiments with microfluidic devices it is required to understand the relevant theories and characteristics of fluid flow, magnetism and various forces on the micro particles. This chapter will describe the relevant fluidic and magnetic theories. Also relevant forces on particles which lead the particle to change the path of flow will be discussed here. Detailed explanations of current microfluidic experiments and set-up will already have been provided in previous chapter (Chapter 1) and later in the next chapter it will introduce the implementation of COMSOL multiphysics for simulating microfluidic experiments. Overall this chapter will provide a literature review and demonstrate different models and applications relevant to this work.

2.1 Microfluidics Theory

2.1.1 Fluid flow Theory

A fluid is a substance that continuously changes or deforms in response to any applied force to its surface. Moreover, fluid behaviour also can change depending on its molecular states, like gas, liquid and plasma states. So fluids can be characterized by several properties like: density, specific weight, pressure, viscosity and surface tension, which will be discussed in this section.

The density of a fluid represents its mass per volume and is expressed by a unit of Kilograms per cubic metre (kgm^{-3}). It can change depending on fluid's temperature, pressure and its state. Depending on the changing behaviour of density property of fluid they can be divided into compressible and incompressible fluids. A fluid with a constant density is an incompressible fluid and for compressible fluids densities vary with changes in temperature and pressure. The density of a fluid relates to gravity using specific weight of a fluid, which is the weight per unit volume of a fluid and expressed by a unit of Newton per cubic metre (Nm^{-3}).

Pressure is force per unit area and expressed by Newton per metre squared (Nm^{-2}). Pressure (hydrostatic) on static fluid at a given depth is the resultant weight of the liquid acting on a unit area at that depth and any pressure acting on the surface of the liquid. It does not vary horizontally within a fluid, but it does vary vertically depending on fluid type. For incompressible fluid it varies linearly with fluid's depth, as incompressible fluid density is constant with change in location or pressure or both.

Viscosity (η) is a property of fluid which provide resistance against its continuous deformation. Its SI unit is Newton second per metre squared (Nsm^{-2}). Though there are two types of viscosities (dynamic and kinematic) exist, but all the discussion in this chapter are on dynamic viscosity (absolute viscosity). A fluid within a channel such as a microfluidic device shows a velocity gradient due to its deformation which is the result of frictional force (also known as shearing stress) between fluid's velocity and flow surface. Dynamic viscosity for a fluid can be shown using

following Figure 2.1:

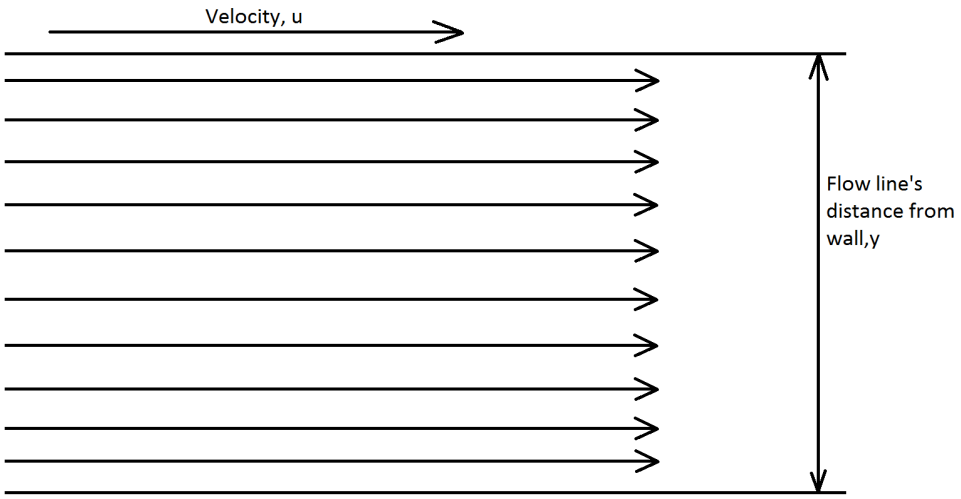


Figure 2.1: Dynamic viscosity of a fluid.

From Figure 2.1, a fluid with viscosity (η) and applied shearing stress (τ) can be related to it's velocity gradient through the relationship shown in Equation (2.1):

$$\tau = \eta \frac{\partial u}{\partial y} \tag{2.1}$$

Here, u is the velocity of the flow line which is parallel to the wall of the microfluidic chip ($\frac{\partial u}{\partial y}$ is the velocity gradient), y is the distance of the flow line from the wall. The equation shown in Equation (2.1) relates to shear stress and viscosity of fluid also known as Newton's law of viscosity. Depending on the relationship between the shearing stress and velocity gradient shown in Equation (2.1) fluids can be classified into two ways Newtonian and non-Newtonian fluids. Newtonian fluids have a linear relationship between the shearing stress (like water) and velocity gradient, when the relationship is non-linear for non-Newtonian fluids (like blood). The fluids used for LOC experiments are Newtonian fluids.

Moreover, when the boundary walls are stationary the fluid tends to stick to the boundary of the channel and results in a no slip condition. On the other hand, when both walls of the channel are stationary, this produces a parabolic velocity profile.

2.1.2 Reynolds number and Flow Types

The Reynolds number is a dimensionless parameter used to define the flow regime of the fluid. It was named after a British engineer and physicist Osbourne Reynolds in 1883 [26, 27]. It is the ratio between kinetic energy to overcome inertia and the energy produced by friction due to viscous forces, that is

$$\text{Reynolds Number} = \frac{\text{Inertial Force}}{\text{Viscous Force}} = \frac{\text{Velocity} \times \text{Distance}}{\text{Viscous Force}}$$

This is now widely used for predicting different flow types, such as laminar, transitional or turbulent flow (shown in Figure 2.2). Reynolds number also relates the fluid property to the geometric properties of a channel (pipe) as well. Thus the Reynolds number can be expressed mathematically as Equation (2.2)

$$R_e = \frac{lu\rho}{\eta} \tag{2.2}$$

Here, l is a characteristic dimension (usually diameter for a flow in a pipe) of the channel(m), u is magnitude of velocity (average velocity) of flow (ms^{-1}), ρ is fluid density ($kg\ m^{-3}$), η is viscosity of fluid (Nsm^{-2}). If the channels are non-circular, then the characteristic dimension in Reynolds number can be expressed as Equation (2.3):

$$l = \frac{4A}{P} \tag{2.3}$$

Where A is the cross section of the channel and P is its wetted perimeter.

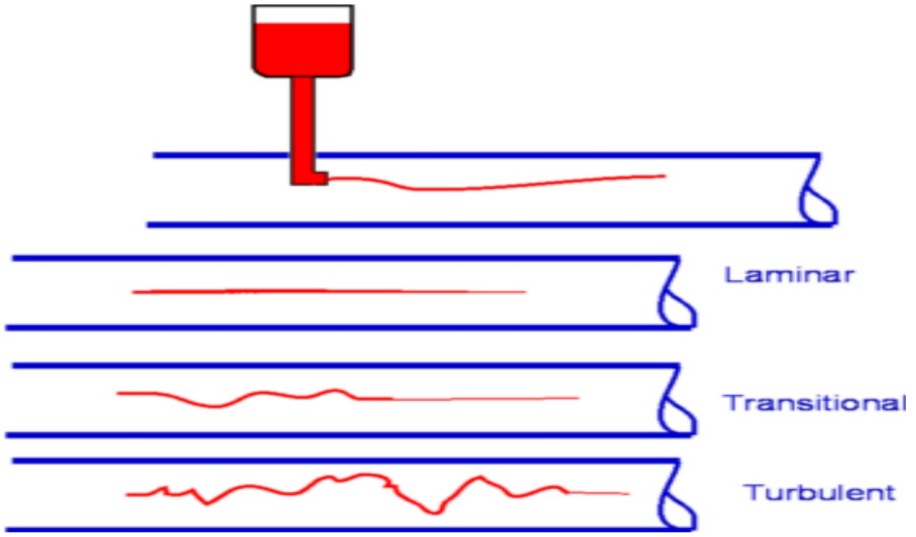


Figure 2.2: General representation of three different types of flow depending on Reynolds number [28].

Depending on the value of R_e, flow can be described as -

- If $1 < R_e < 2000$ then the viscous forces are dominant, meaning the flow is smooth and constant (do not change over time), which means the flow is laminar. The layers of fluid flow in laminar are in parallel to each other inside the channel.

- If $2000 < R_e < 4000$, both the viscous and inertial forces are close to each other. In this case the flow can be defined as transitional flow.

- If R_e is greater than 4000 then the flow is dominated by inertial forces (producing chaotic eddies, vortices and other flow instabilities), which means the flow is turbulent flow and flow becomes unstable and results in eddies and swirls fluid with time.

In microfluidics, the channel diameters and flow velocity are normally very small (micrometre scale), which makes the value R_e very small. Mostly observed microfluidic flow has $R_e < 10$ and are dominated by viscous forces (in most cases inertial forces become negligible). The flow thus results in smooth laminar flow without turbulence [29, 30]. One of the benefits for having a very small value of R_e is that it makes easier for transporting sensitive materials on magnetic particles without mixing inside LOC devices [26].

2.1.3 Diffusion

In the multi stream lamina flow experiments described in Chapter 1, Section 1.2, it is required to pass more than one layer of fluids through the microfluidic chip's channel and chamber with minimal intermixing. However, when multiple layers of fluids are passing through the chip mixing between adjacent fluid flows occurs due to diffusion. Diffusion takes place due to Brownian motion of molecules within the flow. Mathematically diffusion can be expressed as Equation (2.4):

$$\frac{\partial}{\partial t}c(r,t) = D\nabla^2 c(r,t) \tag{2.4}$$

Here c is the concentration of the fluid, r is position vector, t is time for diffusion and D is diffusion coefficient. A general view for the microfluidic flow with diffusion inside an example reaction chamber is shown in Figure 2.3 [1].

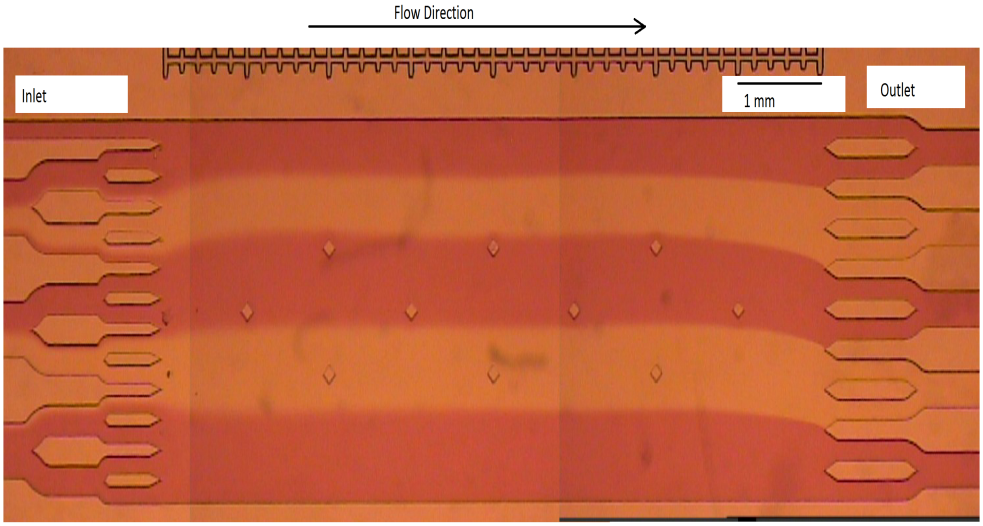


Figure 2.3: General Microfluidic flow with applied flow rate of 10 μLh^{-1} through four different inlets.

Diffusion occurs more between the laminar streams as they cross the reaction chamber and the amount of diffusion depends on the time of travel and diffusion coefficient. For a distance of x for a lamina to cross with diffusion coefficient D, if the required time is t, then it can be described by the Einstein-Smoluchowski (derived

[1] In this figure the diffusion is upward as the upper inlet has smaller width compared to other inlet, resulting smaller fluid flow pressure compare to other inlets.

by Albert Einstein and Polish physicist Marian Ritter von Smolan-Smoluchowski) equation as Equation (2.5):

$$D = \frac{x^2}{2t} \tag{2.5}$$

In Equation (2.5), $\frac{x}{t}$ is mean speed of the particles and x their mean free path, *i.e.,* with the increase in free path, will result in increased diffusion. Diffusion also depends on the molecules' size. Small molecules diffuses more quicker compared with larger particles, because small particles have bigger diffusion coefficient D compared with larger particles [31]. For example at room temperature any aqueous solution with spherical organic dye molecule (MW = 330 $g\ mol^{-1}$) diffuses 10 μm in 0.2 second, but solution with larger particle of bacterium (0.5 μm diameter) require about 200 second to diffuse 10 μm [32].

2.1.4 Flow profile

For any pressure driven microfluidics experiment (fluid is injected using pump or syringe), the profile of flow velocity across a channel has a parabolic shape (Figure 2.4), which is known as Poiseuille flow.

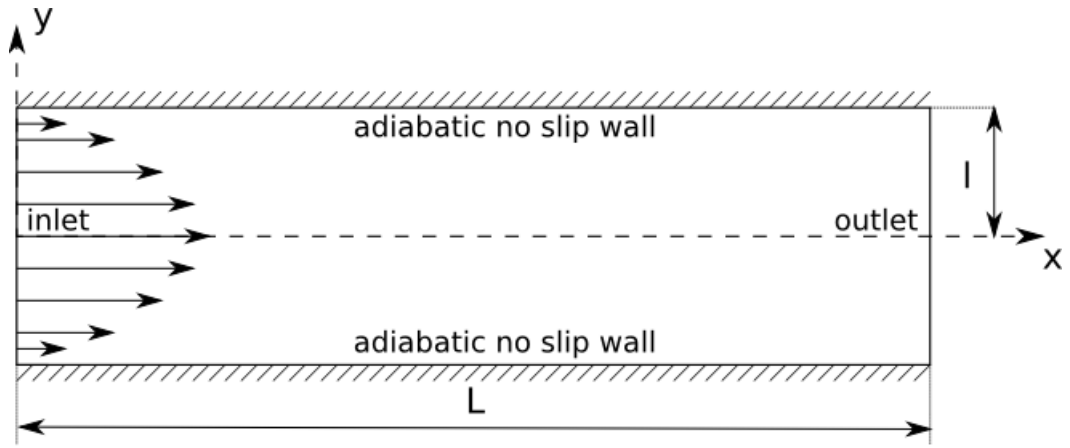


Figure 2.4: Poiseuille flow of fluids across the microfluidic channel [33].

For any fully developed laminar flow (Poiseuille flow) in a circular cross sectioned chamber the velocity profile can be obtained using Navier-stokes (N-S) equations,

which is given by Equation (2.6)

$$v(x,y) = \frac{\Delta p}{4\eta L}(d^2 - y^2 - z^2) \tag{2.6}$$

Here Δp is the pressure difference inside the fluid across length of L, η is the dynamic viscosity of the liquid, d is diameter of the channel and y, z are coordinates of the Cartesian system. From the N-S equation (shown in Equation (2.6)) the maximum velocity is found to be at the centre of the channel, which is $v_{max} = \frac{\Delta d^2}{4\eta L}$ and the average velocity throughout the channel is $v_{avg,x} = \frac{v_{max,x}}{2}$.

In summary for the flow velocity, with a no-slip (having solid boundary) condition and a channel with circular cross section, for any pressure driven flow get the highest velocity at the middle of the channel and nearly zero velocity at the walls of the channel.

The pressure across the channel starts to drop with the increase of the channel length and hydrodynamic resistance from the flow of fluid. The pressure drop (Δp) with flow rate of Q and hydrodynamic resistance (R_{hyd}) across the channel is given by Equation (2.7):

$$\Delta p = R_{hyd} Q \tag{2.7}$$

For a circular cross sectioned channel with length L and of radius a, hydrodynamic resistance (R_{hyd}) is given by Equation (2.8)

$$R_{hyd} = \frac{8\eta L}{\phi a^4} \tag{2.8}$$

But very commonly microfluidic chips contain channels with rectangular cross section, so the hydrodynamic resistance (R_{hyd}) is given by Equation (2.9)

$$R_{hyd} \approx \frac{12\eta L}{1 - 0.63(\frac{h}{w})} \frac{1}{h^3 w} \tag{2.9}$$

Here h and w are the height and width of the channel respectively.

2.2 Magnetism Theory

Permanent magnet plays a very important rule in magnetophoresis experiments, as the magnetic particles in these experiments move across the reaction chamber from inlet to outlet due to the application of external magnetic field. But one of the main problems for using conventional permanent magnets is they do not produce homogeneous magnetic field gradients over the reaction chamber and it is one of the issues for this PhD research. Before going further about this issue, related magnetic field theories will be discussed in the following section.

2.2.1 Magnetic Field

A magnetic field (H) can be generated from any permanent magnet or conductive coil carrying electrical current. A magnet has North and South poles and its strength can be represented by magnetic field flux lines(φ). The density of magnetic field flux lines (φ) indicates the strength of magnetic field, means with a high flux density indicates a strong field. Magnetic flux lines are invisible and considered that they run from North Pole of the magnet to the South Pole of it. Figure 2.5 shows the magnetic field flux lines (φ) via attracting iron filings to a magnet.

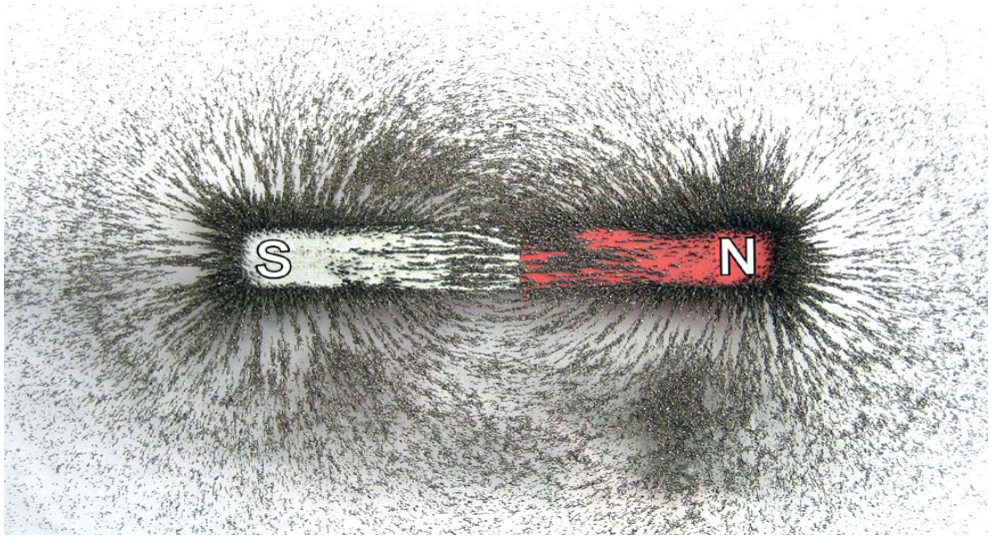


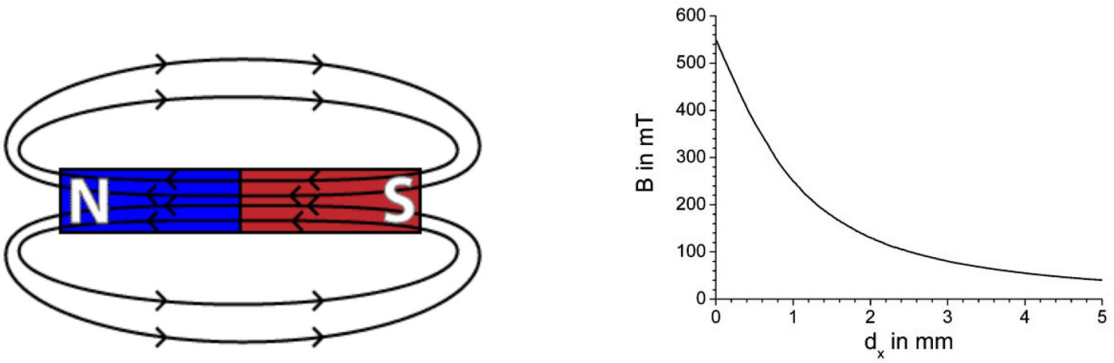
Figure 2.5: The Magnetic Field of a Permanent Magnet as a Result of Loose Iron Filings.

A magnetic field is represented by two quantities, they are magnetic flux lines

(φ) and magnetic flux density (B). The magnetic flux lines (φ) is a measurement of the total magnetic field for a material and magnetic flux density (B) represents the number of field lines present in per unit area. The unit of magnetic flux density is Tesla (T) and mathematically can be expressed as Equation (2.10):

$$B = \frac{\varphi}{A} \tag{2.10}$$

Here A is the area over which the magnetic flux lines are distributed. In addition, the strength of the magnetic flux density (B) has an inverse relation with the distance (d) from the magnet surface, *i.e.* with the increase of distance from the magnet surface, the flux density decrease very quickly. This relation is shown in the graph in Figure 2.6.



(a) The magnetic flux density (B)- in and around the magnet.

(b) The decrease in B with increasing distance from the magnet surface in the x-direction.

Figure 2.6: Characteristics of a typical rectangular magnet [13].

The magnetic flux density (B) is also directly related with magnetic field intensity (H) as Equation (2.11):

$$B = \mu H \tag{2.11}$$

Here $\mu = \mu_0 \mu_r$ is the magnetic permeability of the medium (in free space $\mu_0 = 4\pi \times 10^{-7}$ Henry per metre (Hm^{-1}), μ_r is the relative magnetic permeability) which varies for different materials. Depending on material, μ_r can be either a constant

number or a function of H. If μ_r for a material is independent of H, the material is magnetically linear, otherwise it is magnetically non-linear. Relative permeability for some widely used materials can be found in Table 2.1.

Material	**Permeability (μ_r)**
Supermalloy	1,000,000
Permalloy	70,000
Permendur	5,000
Iron	4,000
Manganese-zinc ferrite	750
Nickel-zinc ferrite	650
Cobalt	600
Manganese	1.001
Tungsten	1.00008
Air	1.00000037
Water	0.9999912

Table 2.1: A relative permeability of different magnetic material [34].

In most cases, when calculating magnetic properties and effects, a vector unit called magnetic dipole with the unit of Am^2 is used for presenting the magnetic moment (permanent or induced) per unit volume in a magnetic material. If a magnetic dipole is placed within a magnetic field H with magnetic induction B, the induction will apply a torque (τ) to align the dipole in such way that the magnetic moment (m_d) of the dipole is parallel to the induction as Equation (2.12):

$$\tau = m_d \times B \tag{2.12}$$

For a magnetic dipole with length l, the moment m_d can be defined as Equation (2.13):

$$m_d = \frac{\phi l}{\mu_0} \tag{2.13}$$

For expressing the effect of magnetic field on any magnetic material placed near to magnet 'magnetization' or 'magnetic polarization' is used. Magnetization is calculated on a magnetic dipole and it varies between different points inside the same body of material. Mathematically magnetization M is the number of magnetic dipole moment within per unit volume, which can be expressed as Equation (2.14):

$$M = \frac{Nm_d}{V} = \frac{B}{\mu_0} \tag{2.14}$$

Here V is the volume, N is number of magnetic moment inside the volume V. Magnetization (M) also describes how a material responds to an applied magnetic field. If a magnetic material is present within the range of a magnetic field, the density of the magnetic field lines will change (Figure 2.7).

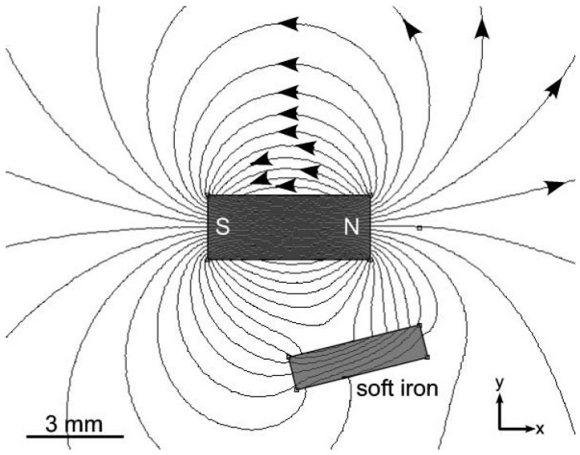


Figure 2.7: The magnetic field line changed by the high magnetic permeability of one soft iron [13].

The magnetic induction field (B) inside a material situated in free space is due to an applied field H generated by the source in the air ($\mu_0 H$) and local induction due to the magnetisation M of the medium ($\mu_0 M$). This can be expressed as Equation (2.15):

$$B = \mu_0 H + \mu_0 M = \mu_0(1 + \chi)H \tag{2.15}$$

Here $\chi(\chi = \frac{M}{H})$ is the magnetic susceptibility of the material. Moreover, relative

permeability related to susceptibility as Equation (2.16):

$$\mu_r = 1 + \chi \tag{2.16}$$

Combining Equation (2.15) and (2.16), B can be expressed as Equation (2.17):

$$B = \mu_0 \mu_r H \tag{2.17}$$

From Equation (2.13) and (2.15) it can be easily seen that induced magnetic field inside the material depends on the relative permeability or magnetic susceptibility of that material.

In microfluidics, permanent magnets are always preferable for their external magnetic field properties to create magnetic field over the microfluidic chamber and manipulate the magnetic material inside the chamber. Using some alloys such as Neodymium Iron Boron (NdFeB), Samarium Cobalt (SmCo) *etc.*, strongest magnetic fields can be achieved. There are some very significant reasons for using permanent magnets rather than electro magnets, though electromagnets are easy to control. For example, when current passes through any wire it generates electromagnetic fields around the wire, also increases the temperature across the wire, which may affects the insulation of the core and surface of the microfluidic chip [13]. Increment in temperature surrounding the reaction chamber also can result in changing the concentration and diffusion coefficient of the analyte(s) and buffer(s). used for the experiment.

2.2.2 Magnetic materials

Exhibition of magnetic properties for any material depends on the orientation and number of magnetic dipoles inside the material, *i.e.* magnetic susceptibility (Equation (2.13)) of the material. Materials with higher magnetic susceptibility show stronger response to any applied magnetic field. So depending on the magnetic susceptibility (χ) of materials, their behaviour can be classified into five broad categories: Diamagnetism (Diamagnetic materials), Paramagnetism (Paramagnetic

materials), Ferromagnetism (Ferromagnetic materials), Antiferromagnetic and Ferrimagnetic [13]. Magnetic susceptibility (χ) for any magnetic material also has relation with the relative permeability (μ_r) (Equation (2.14)). Classifications of different materials according to their response to an applied magnetic field are given in Table 2.2:

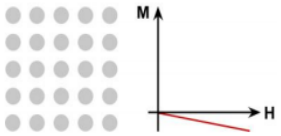
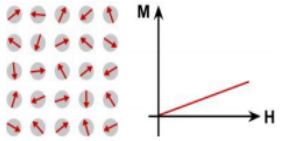
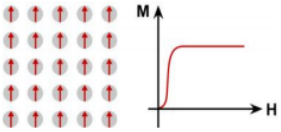
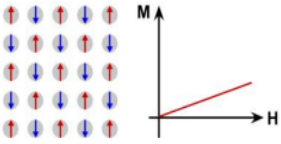
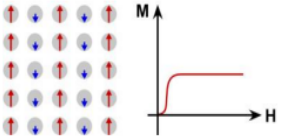
Classification of materials	**Susceptibility (χ)**	**Magnetic characteristics**	
Diamagnetic	Small negative ($\chi < 0$)	Atoms have no magnetic moments.	
Paramagnetic	Slightly positive ($\chi > 0$)	Atoms have randomly orientated magnetic moments.	
Ferromagnetic	Strong positive ($\chi >> 0$) which is function of applied field	Atoms have parallel aligned magnetic moments within domains.	
Antiferromagnetic	Small positive ($\chi > 0$)	Atoms have antiparallel aligned moments.	
Ferrimagnetic	Strong positive ($\chi >> 0$), which is function of applied field	Mixed parallel and antiparallel moments that do not totally cancel.	

Table 2.2: A classification of various types of magnetic materials depending on susceptibility (χ) value.

Diamagnetic Materials

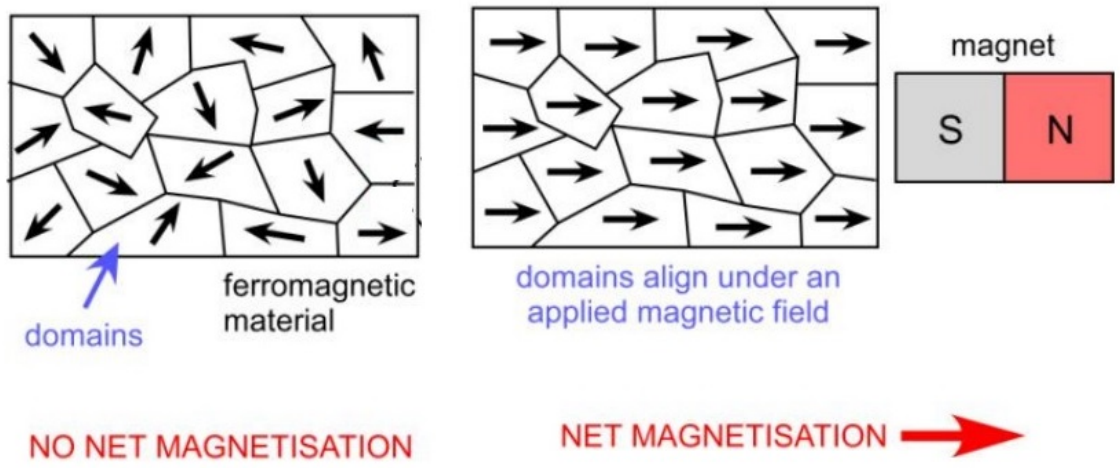
Diamagnetic materials such as water, glass, mercury, silver, copper, bismuth, carbon dioxide and gold *etc.*, do not show the presence of any net magnetic dipole moment in the absence of an external magnetic field. However, in the presence of a magnetic field, diamagnetic materials will exhibit a small number of net negative magnetic dipoles [35]. Relative permeability (μ_r) for diamagnetic materials is less than one and the magnetic susceptibility (χ) is negative, means diamagnetic materials experience slight repulsive force with the presence of permanent magnets.

Paramagnetic material

In contrast to diamagnetic materials, paramagnetic materials do not show the presence of any net magnetic dipole in the absence of external magnetic field, however, they show small net positive magnetic dipoles in the presence of an external magnetic field. So, paramagnetic materials are slightly attracted by a permanent magnet. For such material the relative permeability (μ_r) is slightly greater than one with positive susceptibility (χ). Air, tungsten, manganese, oxygen, sodium, aluminium *etc.* exhibits the characteristic of paramagnetic materials.

Ferromagnetic Materials

Similar to paramagnetic material, ferromagnetic materials do not show any net magnetic response in the absence of external magnetic field. When an external magnetic field is applied, ferromagnetic materials will become a strong magnetic material which is either reversible or irreversible depending on the strength of external magnetic field. If the external magnetic field is not very strong, ferromagnetic material will return to non-magnetic status when the external magnetic field is removed, otherwise, it will stay in a magnetic state. Behaviour for magnetic domains with respect to applied magnetic field is shown in Figure 2.8



(a) Demagnetised State (b) Magnetised State.

Figure 2.8: a) Domains point in random directions in the demagnetised state of a ferromagnetic material, resulting in no net magnetisation. b) When a magnetic field is applied, the domains align in the same direction, resulting a net magnetisation that persists even after removal of the applied field [36].

The magnetization and demagnetization behaviour for a ferromagnetic material can be described using a hysteresis loop (also known as a magnetisation curve) produced as either $M - H$ or $B - H$ curves shown in Figure 2.9. According to this curve if a ferromagnetic material's magnetization starts in its demagnetised *i.e.* with no net magnetisation state, then without any applied magnetic field the magnetisation is also zero. Now if the applied magnetic field (H) increases, the magnetisation (M) of the material also increases and aligns its magnetic domains in the same direction with the applied field. If the strength of the applied magnetic field keeps increasing, more domains will align to the direction of magnetic field till all the domains are oriented towards the same direction as Figure 2.8. At this point after all the domains are aligned towards the same direction, the material can not be more magnetised, this situation is known as Magnetic saturation M_s. After ferromagnetic material reaches its magnetic saturation, if the applied field is reduced to zero, the magnetisation instead of going back to the origin it goes back to a value known as magnetic remanence (M_r). With further increase applied magnetic field in the opposite direction to original direction, after a certain increment the material's net magnetization becomes zero. The magnetic field required to achieve this is

known as the coercivity, H_c. If this negative increment continues another point for magnetisation saturation (opposite direction to the original M_s value) is found. The field is then positively increased again for reaching $M = 0$, thus the hysteresis loop of the $M - H$ curve is finished.

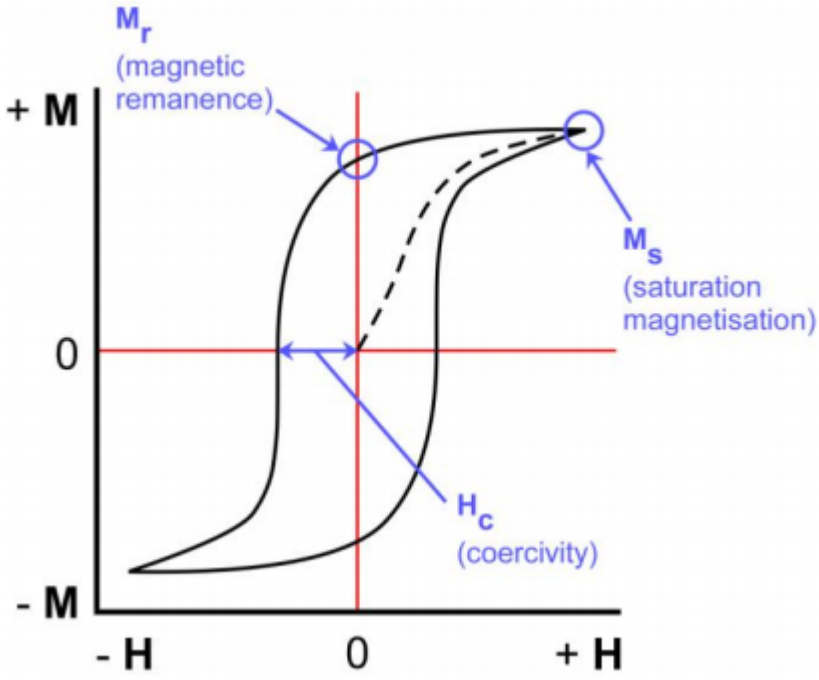


Figure 2.9: Hysteresis loop observed in the M-H curve for a ferromagnetic material. The magnetisation (M) of the material increases with the applied field (H), until it reaches saturation (Ms). As the field is reduced to zero, the material remains magnetised to the value of its magnetic remanence. The field strength required to drive the magnetisation back to zero is known as the coercivity. The positive and negative values of H and M correspond to opposite directions, *e.g.* +M refers to the magnetisation in one direction while M refers to the magnetisation in the opposite direction [37].

Ferromagnetic materials also possess a temperature, known as Curie temperature (T_c) above which its permeability drops, magnetisation is destroyed and the magnetic remanence and coercivity become zero. After this temperature a ferromagnetic material starts to act as paramagnetic.

Antiferromagnetic Materials

Antiferromagnetic materials are similar to ferromagnetic materials in structure. However, the alignment of adjacent magnetic dipoles are anti-parallel to the mag-

netic field, which leading to no net magnetic moment for such material and for Antiferromagnetic materials the relative permeability (μ_r) is about 1 and have small but positive susceptibility (χ). Chromium is an example of antiferromagnetic material.

Ferrimagnetic Materials

Like antiferromagnetic materials, ferrimagnetic materials also have the anti-parallel dipole alignment for adjacent magnetic dipoles. But the magnetic moments are not equal in magnitude, so they do not cancel each other. In this way a net positive magnetic moment is produced for these materials. Manganese-zinc ferrite and nickel-zinc ferrite are examples of these materials.

Superparamagnetic Materials

Superparamagnetic materials are composed from ferromagnetic or ferrimagnetic particles and covered by a dielectric medium. These are from nanometre to few micrometres in sizes without having any magnetic memory. In the presence of external magnetic field, superparamagnetic materials' magnetic moments get aligned in the same direction of the applied magnetic field and become magnetized. It also returns back to the initial demagnetized state with the removal of external magnetic field (Figure 2.10).

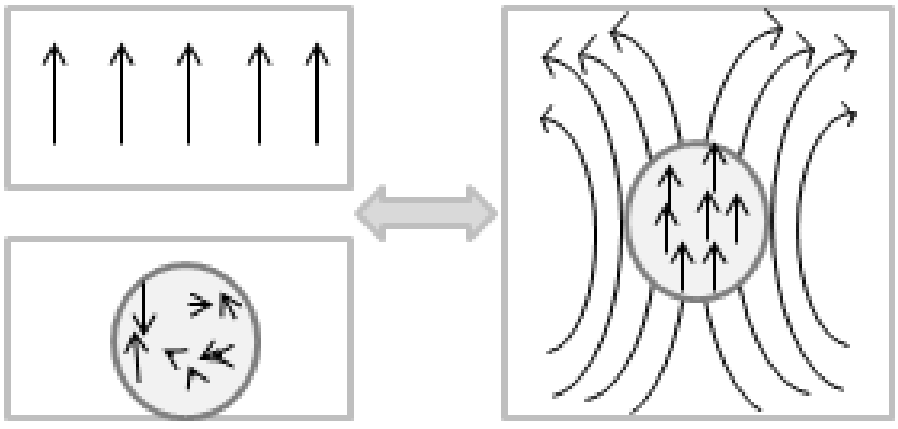


Figure 2.10: Superparamagnetic materials magnetization and demagnetization in magnetic fields [38].

It is found that the observed magnetisation M for superparamagnetic materials

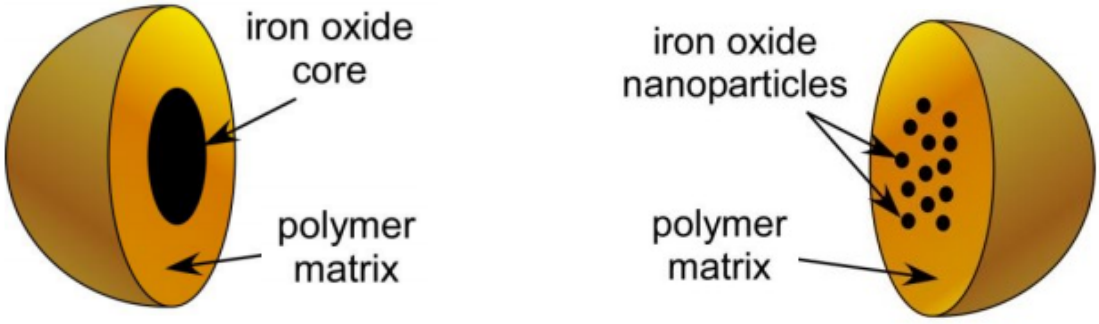
is a function of applied magnetic field H. Iron oxide, commercial Dynabeads are examples of commonly used superparamagnetic materials. The magnetic susceptibility for these materials is very high, so the attraction toward the field is very strong.

Superparamagnetic materials are widely used for magnetophoresis, trapping or focusing microfluidics experiments and normally a coating of polystyrene materials on its surrounding act as the dielectric medium. This polystyrene coating on superparamagnetic materials allows covalent bonding to carboxyl or amino groups for antibody attachment which is useful for different biological and chemical applications.

2.2.3 Magnetic Micro Particles

The performance of microfluidic experiments can be further improved by introducing surface based reactions or specifically by the use of micro-particle, that provide a larger surface area for the reaction. Performing reactions on the surface of moving particles results in higher reaction rates due to higher surface density compared to non-moving chemical reactants. Moving particles provide better sensitivity as well, as it is possible to control the concentration of the sample(s) in different stages of the reaction. One of the advantages of using such particles is the large availability of materials and processes required for making those particles. Such particles are small (diameter in the order of micrometres) spheres, fabricated from various materials and with various surface properties. Such diversity of surface chemistries is crucial for flexibility of the systems, as the materials for fabrication of microfluidic devices are limited and their subsequent modification is often not straightforward or even not available at all. The details magnetic properties of such materials and forces act on them were described in Section 2.2 in this chapter. The widely used materials for producing them are discussed in following paragraph. Also the use of magnetic particles and their benefits of using them are already demonstrated by number of publications from Prof Nicole Pamme [15, 20] and others.

The radius for different magnetic particles can vary from few nm to several μm depending on their uses [39]. Often these magnetic particles are not made from a single component, most of them are complex alloys such as maghemite ($\gamma - Fe_2O_3$) or more complex alloys $MnFe_2O_4$ or $(MO)_n.Fe_2O_3$ (where γ, M and M_n stands for some component like Co, Li, Zn, Ni *etc.*) [40]. Though these magnetic particles are made from ferromagnetic materials, but in practice they show super paramagnetic behaviour [41]. A sample of magnetic particle is shown in Figure 2.11.



(a) An iron oxide core is encased in a polymer matrix

(b) Iron oxide nanoparticles are dispersed throughout the polymer matrix, before being sealed inside with an extra layer of material.

Figure 2.11: Cross-section of two types of superparamagnetic micro-particle.

Most of the commonly used magnetic particles are now produced commercially and lots of research are going on for the development of new particles and surface modifications [42, 43] . Carboxyl groups or amino groups of materials are normally used to cover the surface of super paramagnetic particles. Some well-known producer and suppliers of magnetic particles are Estapor (www.estapor.com), Dynal(www.dynalbiotech.com), Bangs Laboratories (www.bangslabs.com), Micromod (www.micromod.de), Seradyn (www.seradyn.com), Polysciences (www.polysciences.com) *etc* [13].

2.3 Forces on the particles

2.3.1 Drag force on the particle

For passing micro-particles through a LOC chip, it is required that the particles are evenly diluted in a buffer solution. It is considered that the fluidic property for mixer of micro-particles plus buffer is same as the buffer alone and the friction between the particles can be neglected. In this way, when no external force has been applied all the particles have same average velocity due to the flow of the carrier liquid. But if any external force(s) (like a magnetic field) are applied on the particles then a dragging force acts on the particles. If any force is applied on the particle which is perpendicular to the flow, then the drag force to balance a spherical particle is the force shown in Equation (2.18):

$$F_{drag} = -6\pi\eta_m r_p C_w U_{eff} \tag{2.18}$$

Here r_p is the radius of the particles, U_{eff} is the effective velocity of particles (vector summation of velocity caused by an external force and velocity for the flow of the medium) and C_w is the drag coefficient (resistance of an object in a fluid environment). C_w is a dimensionless number and can be calculated if the dimensions of a channel (depth and width) are comparable to the diameter of a particle. If the ratio between dimensions of the channel and diameter of the particle are not negligible then the drag coefficient can be expressed by [44] Equation (2.19):

$$C_w = \frac{1}{\left[1 - 1.001\left(\frac{r_p}{h_z}\right) + 0.418\left(\frac{r_p}{h_z}\right)^3 - 0.21\left(\frac{r_p}{h_z}\right)^4 - 0.169\left(\frac{r_p}{h_z}\right)^5\right]} \tag{2.19}$$

In the equation for drag co-efficient the position of the particle is assumed to be in between two parallel planes and h_z is the closest distance from the parallel wall of the channel, which is also parallel to the flow direction. For a general micro particle inside a channel, whose radius is comparable to the depth and width of the channel, value for drag coefficient is 0.47.

2.3.2 Magnetic Forces

The force acting on a magnetic particle due to the application of an external magnetic field of strength B depends on the volume (V) of the magnetic component inside the micro-particle and the effective magnetic susceptibility (χ_{eff}). Mathematically the force on a magnetic particle can be expressed as Equation (2.20):

$$F_{mag} = \frac{V.\chi_{eff}}{\mu_0}(B.\nabla)B \tag{2.20}$$

Effective magnetic susceptibility (χ_{eff}) is the difference between magnetic susceptibilities of the particle (χ_p) and surrounding medium (χ_m). For diamagnetic materials ($\chi_p < 0$) in a diamagnetic medium ($\chi_m < 0$), the term χ_{eff} can be positive or negative, *i.e.* the particle can be attracted or repelled by the magnetic field. For a microfluidic experiment, where the magnetic particles are diluted in a non-magnetic suspension like water then the effective magnetic susceptibility is less than or equal to one ($\chi_{eff} \leq 1$) and in turn the relative permeability is very close to one ($\mu_r \approx 1$). Normally, the range of applied forces on the magnetic particles can be from a few pN to a few tens of pN [13]. Moreover, the buffer can act as a paramagnetic substance after adding some positive ions like Mn^{2+} [45, 46], Gd^{3+} [47] in the buffer medium.

In practice it is not easy to determine the force on a magnetic particle (due to having not exact information about magnetic volume inside the particle and some other issues (like distance of the particle from magnet surface), so magnetic force on a magnetic particle is expressed by replacing it with an equivalent point dipole with a moment of m_d. So after mathematical manipulation of Equation (2.20) for magnetic dipole, it can be rewritten as Equation (2.21):

$$F_{mag} = m_d.(\nabla B) \tag{2.21}$$

So the magnetic field gradient (∇B) plays here the most significant role in attracting the magnetic particles to the higher magnetic field regions. The higher the value for ∇B, the higher the magnetic force on the particle. But the force on

any magnetic particle is also affected by some short range forces like electric field, though in microscale region magnetic force is strong enough to overcome other short range forces [29].

2.3.3 Viscous Force

During the flow of diluted magnetic particles, the moving velocities of micro-particles are not same as the input flow velocity at the inlet. This happens due to the viscous forces on the particles. The viscous force acting on a particle can be expressed as Equation (2.22) [48]:

$$F_d = 6\pi\eta a(u_f - u_p) = 6\pi\eta a\Delta v \tag{2.22}$$

In Equation (2.22), η is the viscosity of the fluid surrounding the magnetic particle, a is the radius of the particle and Δv is the difference of velocities between the surrounding fluid and the particles. The resultant force acting on the particle is the difference between the magnetic force and viscous force acting on the particle. For efficient detection and to drag the magnetic particles from inlet to the outlet along a desired path, it is necessary that the resultant magnetic force on the particle must be able to overcome such viscous force (also known as hydrodynamic force) that is applied by the surrounding liquid to the magnetic particle.

2.3.4 Gravity

For a magnetic particle with mass m, the gravity force acting on it can be determined via the following Equation (2.23):

$$F_g = mg \tag{2.23}$$

Here g is the acceleration of gravity. Gravity force has a very little effect on the magnetic particle's movement. As comparing to the Z-directional movement, X-Y movement is much higher for the magnetic particles. For example, when a magnetic particle with 1μm diameter is surrounded by a fluid with 0.00089 $Pa -$

S and has the relative velocity of 1 mm/s, it has drag forces of approximately 8.4 pN and the gravitational force becomes approximately 0.004 pN (considering particle density,ρ_p=1800 kg/m^3 and fluid density,ρ_l=1000 $kg\ m^{-3}$). Comparing to the flowing velocity created by 8.4 pN drag force, the downward motion created by 0.004 pN gravity force is too small and thus can be negligible [49].

2.3.5 Diffusion Force

The normal diffusivity value for any molecules (Like ions) which are soluble in water is around $10^{-9}m^2/s$ [50]. This value is smaller for larger particles. It was found that for any microfluidics experiments with micro particles of 1-10 μm diameter the diffusivity changes from 10^{-13} to $10^{-14}m^2/s$ [51, 52]. That means the diffusion of the magnetic particles inside the chamber due to diffusivity property of the fluid is not significant. For example, in widely used Nernst-Einstein relationship equation for measuring particle mobility the diffusion constant for Potassium Chloride (KCl) component in water was predicted as $1.78\times10^{-9}m^2/s$ [53]. According to this prediction the movement of the magnetic particle due to the magnetic flux is several orders of magnitude higher than the mobility due to fluid diffusion [54].

2.4 Application of microfluidics flow and magnetic theories in simulation and modelling

COMSOL Multiphysics software has become a common computer simulation tool for modelling complicated microfluidics experiments, especially for simulating magnetophoresis microfluidic experiments. This software was designed with some customization ability and friendly to use for many Multiphysics (includes chemistry and many engineering application) applications without limiting into specific equations. Therefore, users also can define their own equations for combining with experimental system design. There are also lots of predefined equations and functions for different modules within COMSOL Multiphysics software. The first thing required

when setting up COMSOL Multiphysics to work is to choose the model(s) which contains predefined equations for solving a specific experiment. Therefore, before performing the computational simulation work for any experiment, it is necessary to understand the basic structures and conditions for that experiment. In addition, all the specifications (dimensions and materials) of microfluidic chips and used experimental reagents' properties are required for every design stage to ensure if the design will correctly matches the practical experiment. After the first conceptual model is correctly set-up, different parameters within the model can be changed to perform the simulation for a variety of experimental conditions.

An understanding of all the equations and effects discussed in this chapter is necessary for simulating microfluidics fluid flow using COMSOL Multiphysics, particularly for setting up boundary conditions and predicting the velocity and pressure of fluid flow within the microfluidics chip. As many predefined equations in COMSOL model are not suitable for all kinds of microfluidics chips, especially those with complex geometries. For example, the Navier-Stokes equations are valid when the microchannel length (L) is much larger than the fluid molecules' free path (λ), which means the Knudsen number, ($K_n = \lambda/L$) is less than 0.01. When $0.01 < K_n < 0.1$ special boundary conditions must be used with Navier-Stokes theorem, while fluid with $K_n > 0.1$ cannot be solved with Navier-Stokes equations.

Therefore, during the simulation modelling, the Navier-Stokes equations can be adjusted (simplified or additionally combined with other equations) depending on the nature of the flow regime. For example, considering the velocity of lamina flow is constant and very small, thus the N-S equation in microfluidics is reduced to a much simpler equation. However, in the situation of flow driven by higher pressure at an inlet compared with an outlet, it needs to be solved by balancing the pressure force and viscous forces with N-S equations, which requires a Reynolds-Averaged Navier-Stokes (RNAS) formulation instead of N-S Equations alone as Equation (2.24) and

(2.25):

$$\rho(u.\nabla)u = \nabla.\left[-pl + /mu\left(\nabla u + (\nabla u)^T\right) - \frac{2}{3}\mu(\nabla.u)l\right] + F \tag{2.24}$$

$$\nabla.(\rho u) = 0 \tag{2.25}$$

Where u is the time-averaged velocity, p is the average pressure, μ stands for turbulent viscosity and F stands for external force.

Moreover, before moving forwards for simulating experimental studies, COMSOL provides an excellent simulation practice starting point with its built-in model libraries that model micro particle movement in fluid flow in the presence of a magnetic field. This helps users to understand the particles' movement with effect of external force.

2.5 Summary

This chapter has summarised the basics of the physics behind the magnetophoresis microfluidic experiments, which induces properties of the materials inside microparticles, fluid flow characteristics, forces acting on the micro-particles to make it move. It was required to understand those theories and characteristics, as without them it was not possible to simulate a standard microfluidic experiment in COMSOL Multiphysics. In real-life experiments many factors affect the experimental results. But it was not possible to consider all of those factors during the process of magnetophoresis experimental simulation, that's why they were not mentioned here. The following chapter concerns the actual simulations performed as part of this work and explains its links to image processing work described in the chapters after that.

Chapter 3

MAGNETOPHORESIS SIMULATION

For the magnetophoresis experiments of interest it is required to have a magnetic field gradient which is homogeneous [13], so that most of the magnetic particles can reach the desired outlet. But one of the main problems for most of the magnetophoresis experiments is their magnetic field across the reaction chamber does not decrease linearly, implying an inhomogeneous magnetic field gradient. This non-linear change in magnetic field distribution results in many magnetic micro-particles failing to reach the desired outlet. This also may result in aggregation of micro-particles, if the flow rate is not controlled efficiently. Many literature sourcs [55] proposed their way for achieving homogeneous magnetic field gradient. However, for most of the cases the approach was to develop a specially shaped magnet, which could be easy to use for computer simulation purpose only. For real-life use of such magnets are either expensive or difficult to produce or require special instrumental set-up to use them. In this situation, a way is to be found to use a regular shaped magnet to produce a homogeneous magnetic field gradient. Keeping in mind the target of using conventional permanent magnets and experimental set-up several magnetic field simulations, including Halbach magnet arrays with many different orientations were investigated.

High levels of particle transport and aggregation issues may result in complete failure of a magnetophoresis experiment. However, as will be seen in later discussions, even moderate particle aggregation may cause problems for an automated tracking and measurement system. It is therefore highly desirable to design a chip and magnetics combination which provides the best possible magnetic field conditions, particularly within the measurement region near the outlet. The ability to accurately simulate the magnetic and fluidic forces on particles may also lead to improvements in tracking capability based on a physical model, however this was beyond the scope of this work.

3.1 Simulation procedure

In a basic LOC magnetophoresis experiment, the particles are usually assumed to move along the direction of the vector combination of magnetic force and force from fluid flow. In this chapter the aim is to represent by simulation the process of LOC experiments considering the same parameters used in real-life experiments. COMSOL Multiphysics (www.comsol.com/comsol-multiphysics) was used for simulating these microfluidic experiment(s) including magnetophoresis experiments. Using COMSOL simulation for magnetophoresis microfluidic experiments it was possible to analyse the effect(s) of many different size and shaped magnets on experimental result(s) placed in different positions with respect to reaction chamber. Also effects of several magnetic orientations were also checked using COMSOL.

In the simulations the forces acting on the micro-particles were considered to only be the magnetic force and the force from the fluid flow. There are other small factors that also affect the motion - like gravity, but these effects were relatively negligible.

A couple of microfluidic chip designs were simulated, but in this thesis only the results using a LOC chip designed by Prof. Nicole Pamme (Department of Chemistry, University of Hull) are presented. The code name for this chip is NP55 and the CAD diagram for NP55 microfluidic chip is shown in Figure 3.1:

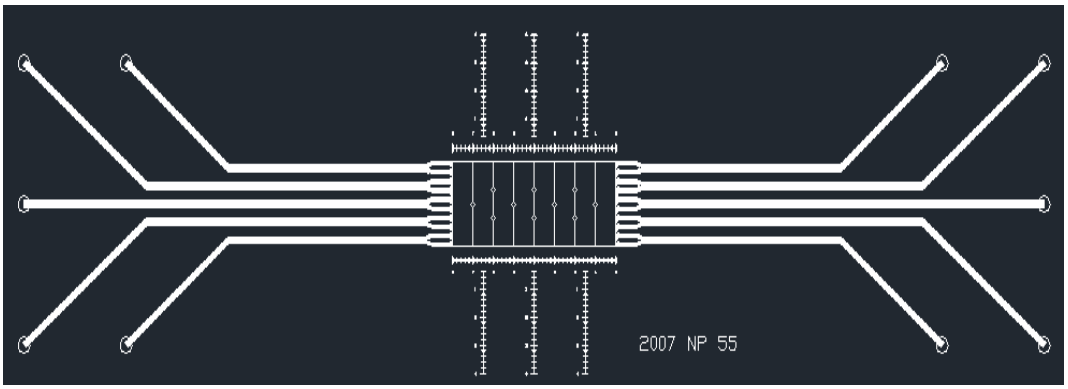


Figure 3.1: 2D CAD diagram for NP55 microfluidic chip designed by Prof. Nicole Pamme.

NP55 microfluidic chip contains five inlet and five outlet channels. The length of these inlet and outlet channels are different, but the width and depth are the same.

The parameters for different simulations vary depending upon the requirement(s), but other parameters are common for most of the cases, the common parameters are listed in Table 3.1.

Parameter		**Description**	**Value**	**Unit**
Diamagnetic particles	R_p	Particle radius	1.4 (Dynabeads M-270 Streptavidin)	μm
Reaction Chamber	L_{RC}	Length of the Reaction Chamber	8000	μm
	W_{RC}	Width of the Reaction Chamber	3000	μm
	D_{RC}	Depth of the Reaction Chamber	20	μm
Microchannel	C_w	Channel width	200	μm
	C_d	Channel Depth	20	μm
	C_l	Channel Length	Three different lengths	μm
	V_{fl}	Volume flow rate	40	$\mu L/hr$

Table 3.1: List of the parameters used for making the analytical model. Some of the parameters are varied in the experiment, and their specific values are referred in the text.

Using the parameters from (Table 3.1) and the NP55 (Figure 3.1) microfluidic chip a simulation of an experiment for the rapid detection of the inflammatory biomarker (C-Reactive protein) was performed. The physical experiment was set up by Dr Chayakom Phurimsak *et al.* (Chemistry lab, University of Hull, UK) [56]. For simulating this experiment, the approach can be divided into following three steps:

1. Simulation for observing flow distribution and diffusion.
2. Simulation of magnetic field distribution over the reaction chamber using rectangular shaped permanent magnet (NdFeB).
3. Magnetic field simulations for achieving homogeneous magnetic field gradient.

Each of these steps is going to be discussed in the following sections with respect to a known and experimentally verified chip and magnet configuration. Later alternative magnetic designs using the same simulation procedure will be explored.

3.1.1 Fluid flow

For the typical fluid flow rates, flows within the LOC chip are laminar (as Reynolds number, $R_e << 1$), so mixing occurs due to diffusion only. For most of the magnetophoresis experiments, almost all the fluids used are similar in colour (without fluorescence signal). So the diffusion is not easy to see with the naked eye. For visualising the diffusion inside the chip a simple experiment was carried out by Dr Martin Vojtek (Department of Chemistry, University of Hull, UK), where two different coloured fluids (blue and yellow) were injected inside the microfluidic chamber at a flow rate of 300 μLh^{-1}. The experimental result for diffusion can be seen in Figure 3.2.

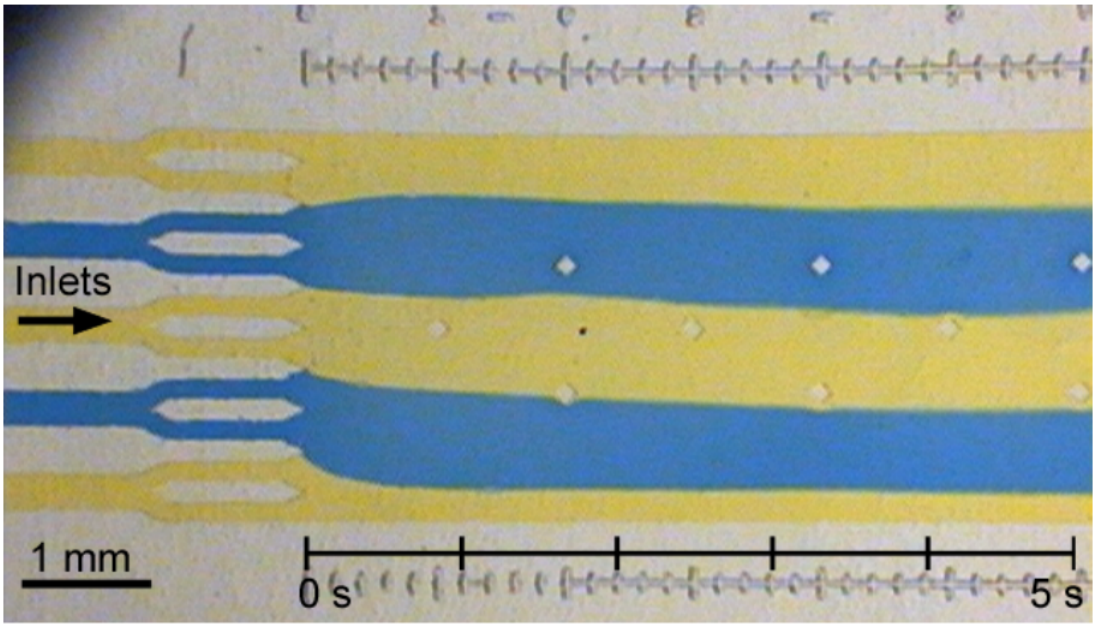


Figure 3.2: Flow and diffusion with the microfluidic chip filled with blue and yellow coloured inks at flow rate of 300 μLh^{-1} (velocity 1.4 $mm\ s^{-1}$).

From Figure 3.2 it can be seen that the stream's flow path gets diffused but they do not mix completely.

The NP55 microfluidic chip (which is used for simulation) contains channels (both inlets and outlets) with same depth and width, but with different lengths. The difference in length changes their fluid flow velocity within the reaction chamber. This change in velocity inside reaction chamber effects the diffusion. For the simulation of the diffusion of fluid flow using COMSOL Multiphysics the parameters showed in Table 3.2 were used.

Parameter		**Description**	**Value**	**Unit**
Fluid	D_c	Diffusion co-efficient	1.554×10^{-10}	$m^2 s^{-1}$
	V_{fl}	Volume flow rate (per channel)	40	μLhr^{-1}

Table 3.2: Parameters used for simulating fluid flow diffusion.

In COMSOL Multiphysics the fluid flow was defined as an incompressible flow following Navier-Stokes theorem. The flow rate was varied from 20 to 100 μLh^{-1} through each channel keeping all the other parameters same. After defining all the

required boundary conditions, the simulations were done. Though the flow rate was varied with small steps, in this thesis it was desired to compare the result with a published research paper from Prof. Pamme's research group [56]. According to Pamme *et al.* [56], for their experiment done in the Chemistry lab the optimum flow rate for most of the magnetic microparticles to cross the entire reaction was 40 μLh^{-1}. So in this section only the results for an inflow rate of 40 μLh^{-1} per channel are presented.

For simulating the diffusion of diluted species inside the NP55 chip for velocity of fluid using COMSOL Multiphysics it was required to solve the Navier-Stokes equations for the laminar flow first. Laminar Flow under the CFD module in COMSOL was used to do this simulation. In this way, for simulating the laminar flow due to inflow pressure was calculated using Equations (2.24) and (2.25) in Chapter 2. For ease of presentation we recall Equations (2.24) and (2.25) as Equations (3.1) and (3.2) as follows:

$$\rho(u.\nabla)u = \nabla.\left[-pl + /mu\left(\nabla u + (\nabla u)^T\right) - \frac{2}{3}\mu(\nabla.u)l\right] + F \tag{3.1}$$

$$\nabla.(\rho u) = 0 \tag{3.2}$$

After getting the pressure distribution (via magnitude of velocity) across the reaction chamber, the diffusion of the input stream (diluted species) can be measured. In this situation the Laminar Flow module was coupled with Transport of Diluted Species modules under the Chemical Reaction Engineering Module. This coupling facilitates the simulation of concentration distribution for diluted species across the reaction chamber. For simulating the concentration distribution (diffusion) the used equations are shown as Equation (3.3) and (3.4).

$$\nabla.(D_i\nabla c_i) + u.\nabla c_i = R_i \tag{3.3}$$

$$N_i = -D_i\nabla c_i + uc_i \tag{3.4}$$

In Equation (3.3) and (3.4), c is the concentration of the species ($molm^{-3}$), D is the diffusion constant (m^2s^{-1}), u is the velocity of the flow, R is the quantity used to describe if the chemical reaction is producing ($R > 0$) or destroying ($R < 0$) more of the species, ∇ is the gradient and ($\nabla .$) is the divergence. For magnetophoresis LOC experiment R was set as zero, as no chemical reaction occurred to produce or destroy species. In Equation (3.4), N is the molar flux ($molm^2s^{-1}$). In both these equations i is representing different species. Equation (3.4) is also known as Ficks first law of diffusion.

After defining all the required parameters in COMSOL Multiphysics for computing all the equations shown in Equation (3.1), (3.2), (3.3) and (3.4), the geometry of NP55 chip was divided into small parts using "triangular" meshing. Figure 3.3 is showing the snap shot for structural mesh for the whole NP55 microfluidic chip.

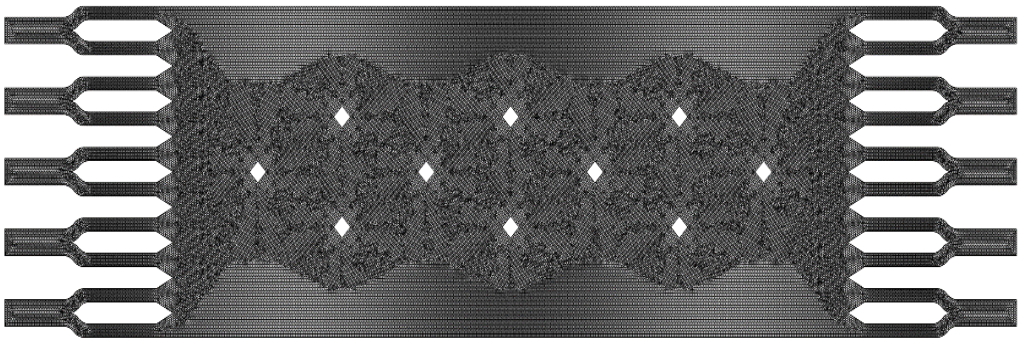


Figure 3.3: Meshing of NP55 microfluidic chip.

Here within the Figure 3.3, the "White" coloured rectangles are representing the micropilers. For the meshing of the microchip, it was done with an adaptive "triangular" size of the individual block.

By coupling the incompressible Navier-Stokes multiphysics from Laminar Flow module with Transport of Diluted Species module the simulated velocity distribution and concentration of the diluted species is shown in Figure 3.4 and Figure 3.5

respectively.

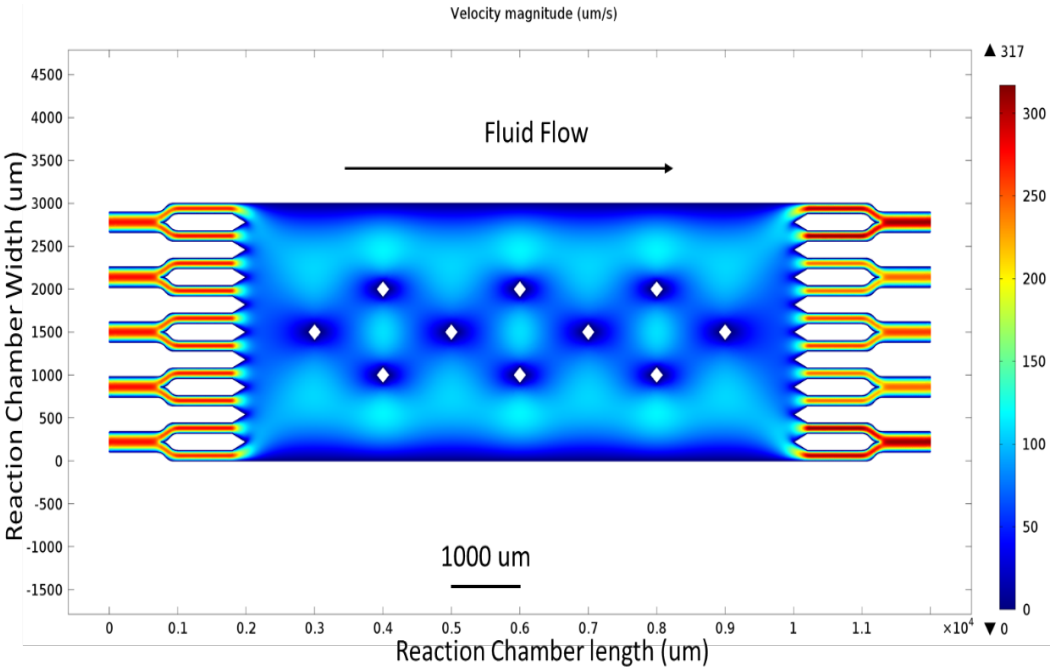


Figure 3.4: : Simulated velocity distribution for the flow rate of 40 μLh^{-1}. In the right side of the image, the colour range is showing the relation of the colour and velocity magnitude.

From Figure 3.4 it can be seen that the velocity of the flow is higher across the inlets and outlets, but lower across the reaction chamber. The reason for the velocity to be lower within the reaction chamber is reaction chamber has bigger area (also volume) compare to the areas of the inlets and outlets. This relation between the velocity and area can be shown as Equation (3.4)

$$Au = Constant \tag{3.5}$$

Here, A is the area and u is the velocity.

After the fluid passes the reaction chamber and enter the outlet, the area for the fluid to flow reduces. This reduction in area increases the velocity of the flow again.

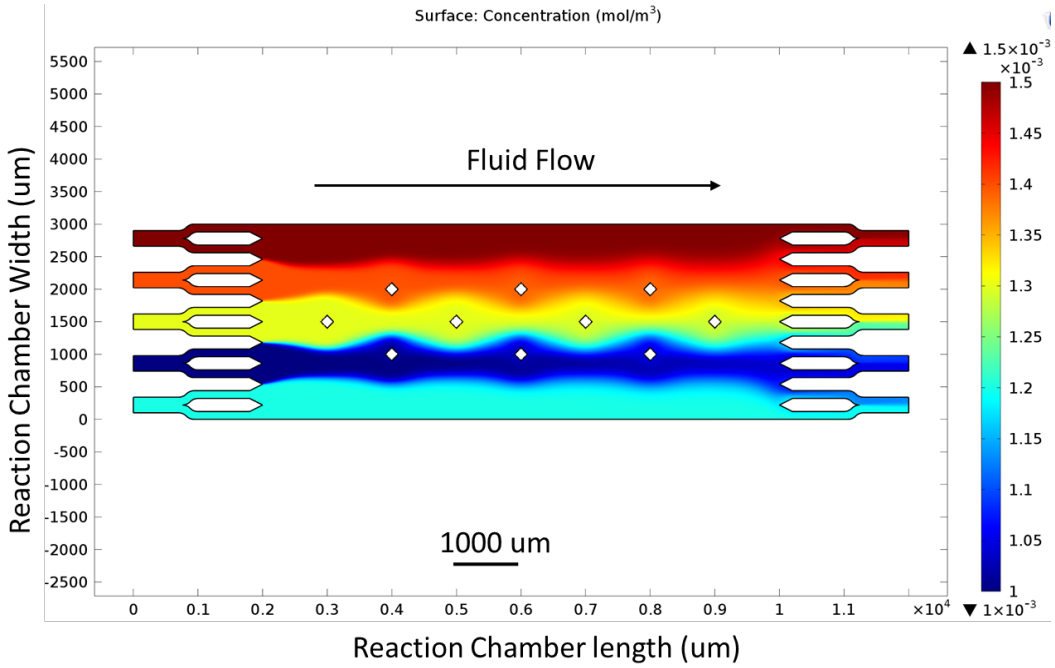


Figure 3.5: Simulated result for diluted species across the reaction chamber and channels with flow rate of 40 μLh^{-1}. In the right side of the image, the colour range is showing the relation of the colour and concentration.

Figure 3.5 is showing the concentration of the diluted species across the reaction chamber including inlets and outlets. Here the concentration distribution also represents the diffusion of the species. Though this simulation was done using several inflow velocities, the result shown here is only for an inflow velocity of 40 μLh^{-1}. For the same volume of the chip, it was found that after the inflow velocity reaches 8 μLh^{-1}, there was no significant change in the diffusion.

Diffusion occurs due to the hydrodynamic resistance (act against the flow) and the fluid's diffusivity property. Fluid with lower velocity uses more (as a ratio, not as a value) of its energy to overcome the resistance, which results in more diffusion. But for higher flow rates the fluid uses less energy (as a ratio, not as a value) to overcome the hydrodynamic resistance, which results in less diffusion. When the inflow velocity reaches 8 μLh^{-1}, hydrodynamic resistance does not make any further change in diffusion and also for every fluid the diffusion property is constant. For this reason, after the inflow velocity reaches 8 μLh^{-1} the diffusion of the inflow species are similar for almost all the flow velocities.

3.1.2 Magnetic Field simulation for known chip

In the second stage of simulating magnetophoresis experiment it was required to compare the magnetic field distribution across the reaction chamber with that described in [56]. In this article by Prof. Nicole Pamme's research group, the magnet was a permanent Neodymium-Iron-Boron (NdFeB) magnet of grade 50 (Code Name N50) (www.magnetsales.com/Neo/Neoprops.htm). NdFeB is one of the most widely used permanent magnets which was first developed in 1982 by General Motors and Sumitomo Special Metals. The magnetic properties of NdFeB are described in [57]. According to [56], the magnet used was $4{\times}4{\times}5\ mm^3$ in size and was placed near to the reaction chamber as shown in Figure 3.6:

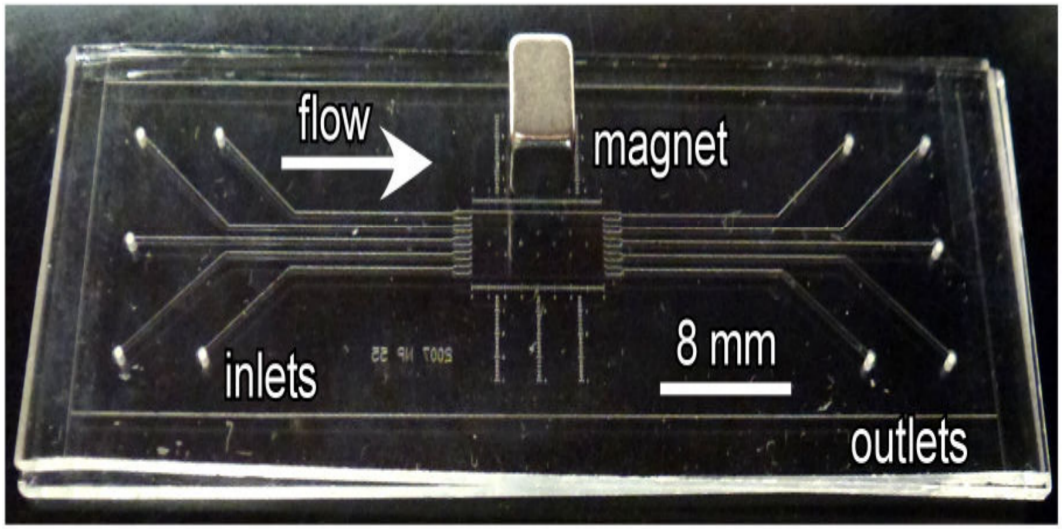


Figure 3.6: Photograph of NP55 microfluidic chip with a $4{\times}4{\times}5\ mm^3$ NdFeB (N50) magnet placed on top of the chip. The scales on the chip showing the position of the magnet [56].

From the Figure 3.6, an idea was generated about the position of the magnet on the microfluidic chip, after that it was discussed with one of the experimenters (Dr Chayakom Phurimsak) to get more detail about the magnet and its position. According to the discussion the magnet set-up during the experiment and its parameters can be presented in Table 3.3:

Parameter		Description	Value	Unit
Magnet (Rectangular)	M_s	Residual magnetization	1.1×10^6	A/m
	H_m	Height	5000	μm
	W_m	Width	4000	μm
	D_m	Depth	4000	μm
	∇L	Distance between the magnet edge and the edge of reaction chamber	500	μm
Microchannel	C_w	Channel width	200	μm
	C_d	Channel Depth	20	μm
	C_l	Channel Length	Three different lengths	μm
	V_{fl}	Volume flow rate	40	$\mu L/hr$

Table 3.3: Parameters used for simulating magnetic field distribution across the microfluidic chip.

The parameters shown in Table 3.3, were used for the simulation in COMSOL Multiphysics. For simulating only magnetic field, COMSOL Multiphysics only requires "Magnetic Fields" module under "AC/DC" section. However, it was required to combine magnetic field simulation result with the results from "fluid flow" and "Transport of Diluted Species". For this, both these modules were coupled so that, the pressure acting on the fluid (in transport of diluted species module) comes from the fluid flow (fluid module). Also the boundary for magnetic field distribution, does not have any effect on the fluid flow or concentration of the species. Moreover, to observe the magnetic field distribution across the reaction chamber from NdFeB magnet, so the reaction chamber area and magnet as were selected as areas of interest. Figure 3.7 is showing the position of the NdFeB magnet with respect to the NP55 microfluidic chip.

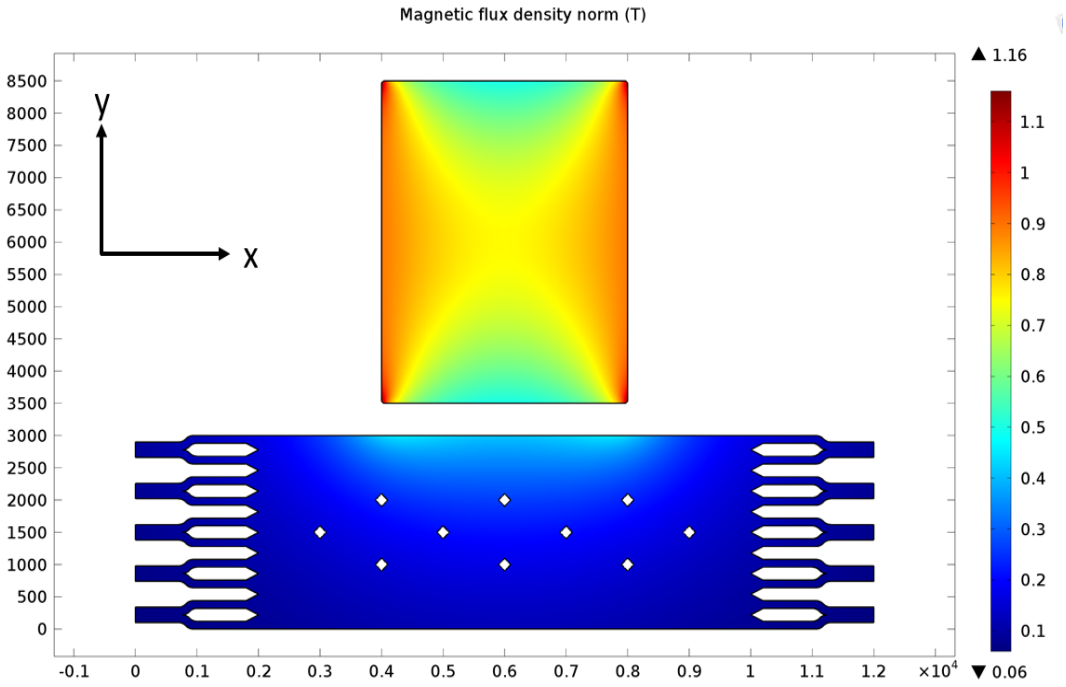


Figure 3.7: Position of NdFeB magnet with the NP55 microfluidic chip.

In this figure (Figure 3.7) the boundary conditions are not displayed. The boundary is many times larger compared to the magnet and chip, so the display of boundary will make the chip and magnet too small to see. According to [56], the magnet was placed with the chip in such way that the magnetic flux comes out from the side of the magnet, which is most close to the chip, that is the north pole of the magnet was close to the reaction chamber. The magnetization process of the magnet during the simulation is shown in Figure 3.8.

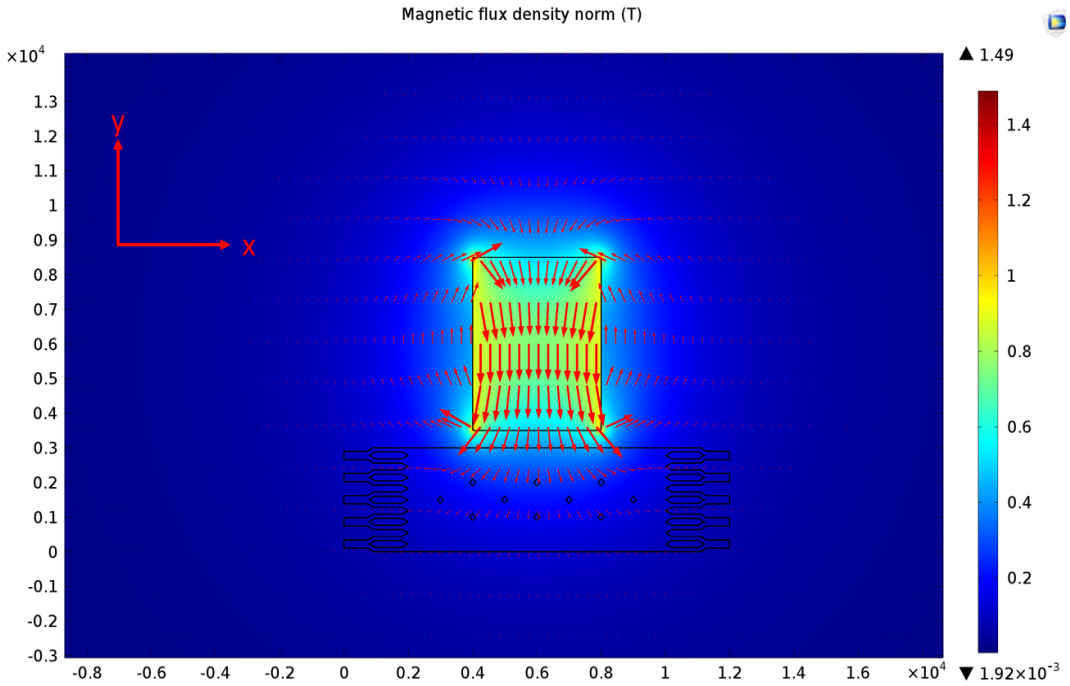


Figure 3.8: Magnetization direction of the NdFeB magnet.

In Figure 3.8, the arrows are showing the direction of the magnetic flux. In this figure it can be seen that the magnetic fluxes are coming out from the north pole (near to the chip) of the magnet and after travelling through the chip and air they are reaching to the south pole of the magnet. Here the lengths of the arrows are proportional to the strength of the magnetic field on that position. However, the main interest was to see the magnetic field distribution within the reaction chamber. So, as an area of interest, reaction chamber was selected and shown in Figure 3.9:

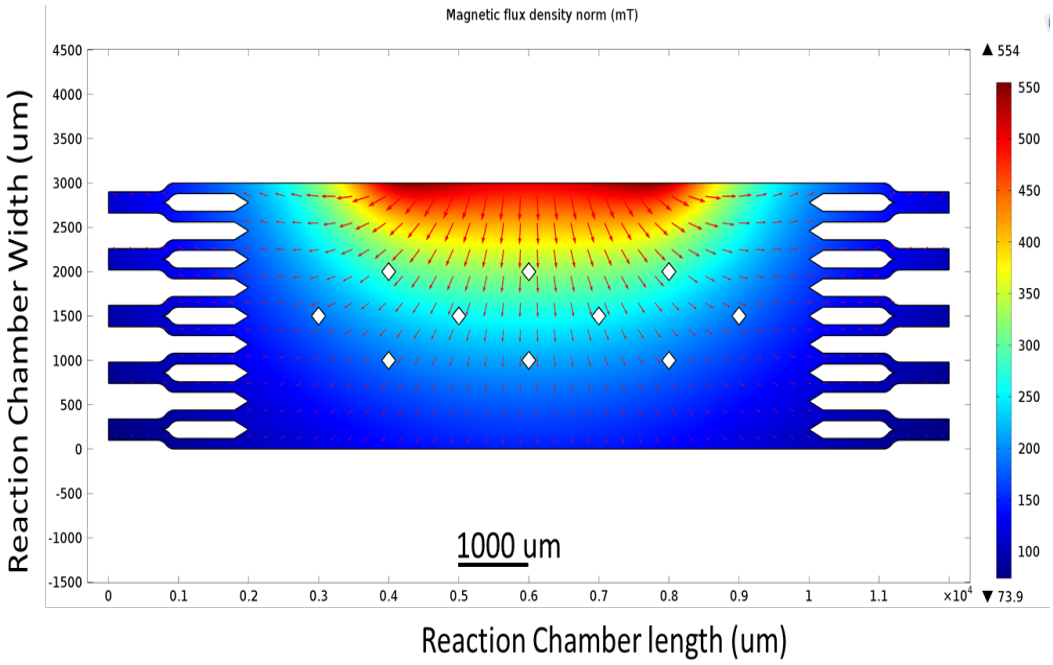


Figure 3.9: Magnetic field distribution across the LOC reaction chamber and channels.

From Figure 3.9, it can be seen that maximum value of magnetic flux across the reaction chamber is 554 mT, which is very close to the experimental value of 561 mT [56]. As before the arrows are showing both the direction and magnitude of magnetic flux distribution across the reaction chamber. For a better understanding and presentation, the change in magnetic flux was calculated across vertical centre of the reaction chamber. The position of the vertical line is shown in Figure 3.10

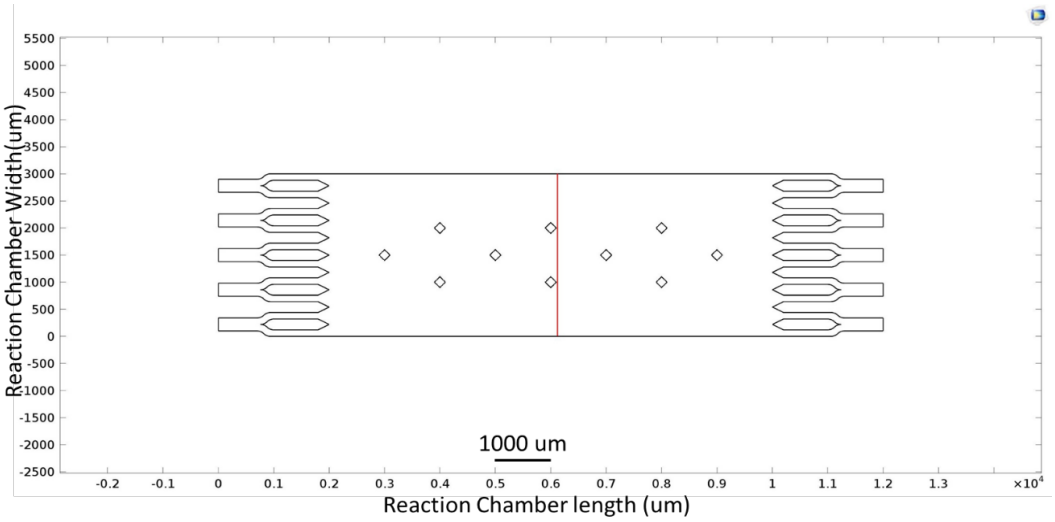


Figure 3.10: Visualization of a vertical line across the centre of the reaction chamber.

The change is magnetic flux density across the central vertical line is shown in Figure 3.11.

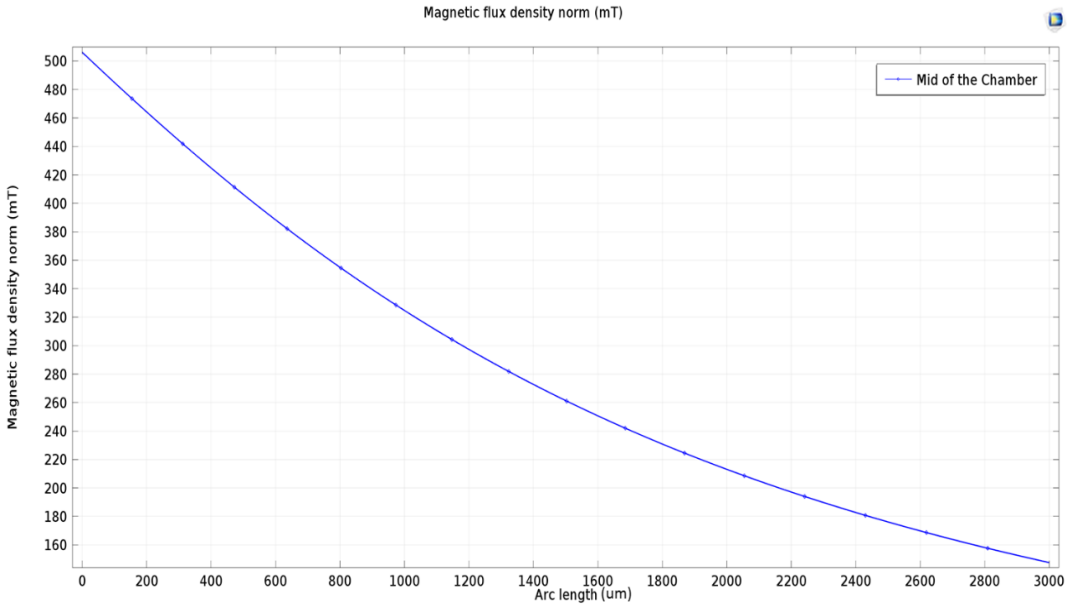


Figure 3.11: Magnetic flux density across the vertical line through the centre of the microfluidic chip.

Change in magnetic flux density across the vertical line through the centre of the chip is shown in Figure 3.11. According to Miwa *et al.* [58], the strength of magnetic field decreases rapidly, with increasing distance from the surface of a magnet. Figure 3.11 verifies that the magnetization process in our simulation follows all the theory of magnetic field and experiment done by Prof. Nicole Pamme's research

group [56]. Moreover, according to Heinrich *et al.* [59], the specific magnetic field gradient plays a very important role in determining the movement of the magnetic nanoparticles from the inlet to outlet. Even after applying a sufficient inflow velocity and magnetic field, lots of particles get aggregated even with a small amount of non-homogeneity in the magnetic field gradient.

Part of the purpose of the simulation work was to develop a magnet arrangements which would provide a more homogeneous gradient. These simulations were used to check that the results obtained with a known configuration matched experimental reality. Figure 3.12 shows the change in magnetic field gradient with respect to distance (across the centre of the reaction chamber shown in Figure 3.10).

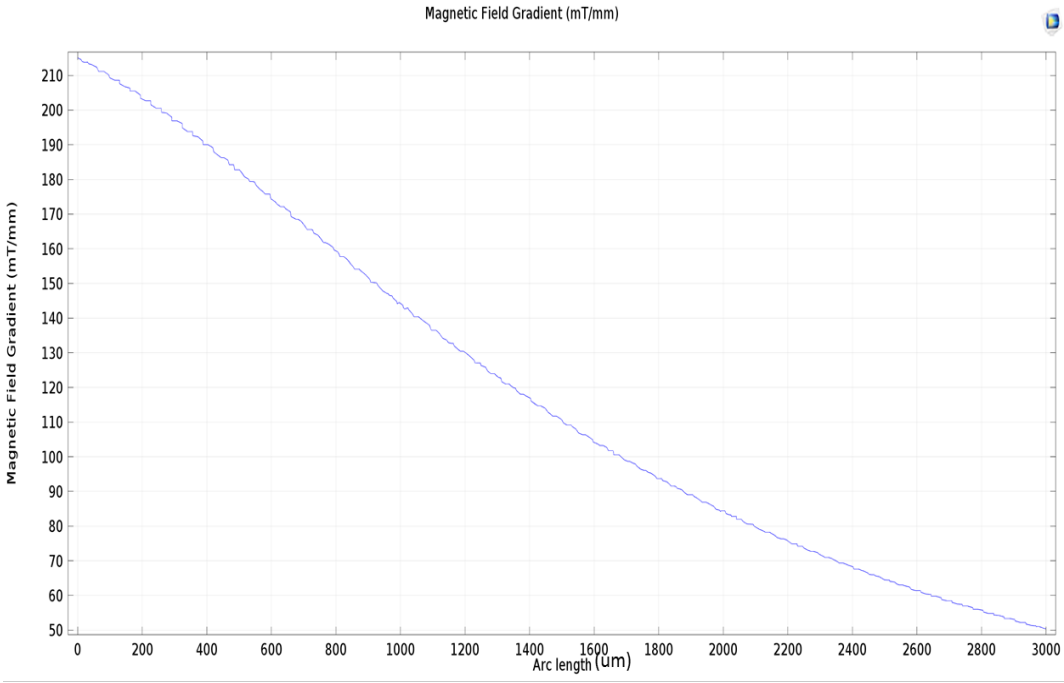


Figure 3.12: Magnetic field gradient plot across the middle of the reaction chamber.

According to the simulation result the gradient value changes approximately from 210 mT/mm (near to the magnet) to 50 mT/mm (at the most distant point from magnet in the reaction chamber). This result shows that magnetic field gradient decreases by around 160 mT/mm . This non-homogeneous magnetic field gradient is one of the main reasons for many magnetic microparticles not reaching the outlet.

3.1.3 Design for homogeneous magnetic field gradient

The Halbach array provides a promising approach to the problem of creating a homogeneous field gradient [60]. In a Halbach array, the directions of magnetization for the various magnets are different. The number of magnets used in Halbach array can be varied, but the shape of all the magnets used in a given Halbach array is the same. For the experiment of interest, which concerned DNA hybridization and determination of C-Reactive protein the dimension of the magnet used was $4 \times 4 \times 5\ mm^3$ and the total effective length of the reaction chamber was 12 mm. So for simulating Halbach array magnetization, it was not required to use more than four magnets. Several COMSOL simulations were done on Halbach array magnetic field by changing magnetization directions, but most result did not provide magnetic fields that were suitable. However, one of the simulations showed a change in magnetic field gradient which was smaller than the single magnet case. So in this chapter only that simulated result is discussed. For this simulation the direction of magnetization for the four magnets was set up as shown in Figure 3.13.

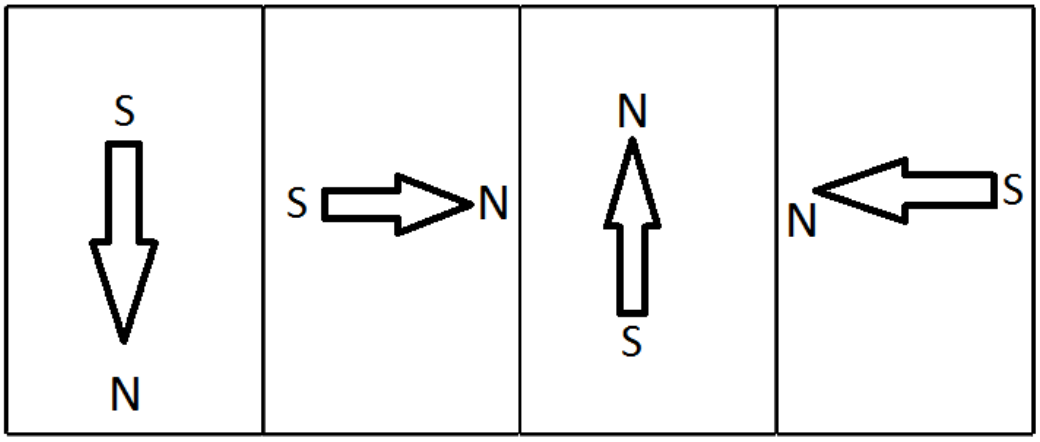


Figure 3.13: Magnetization direction for magnets in Halbach array.

For simulation purposes the magnetization value was kept the same as used in previous section. The magnetic field distribution across the reaction chamber for the combination shown in Figure 3.13, is shown in Figure 3.14.

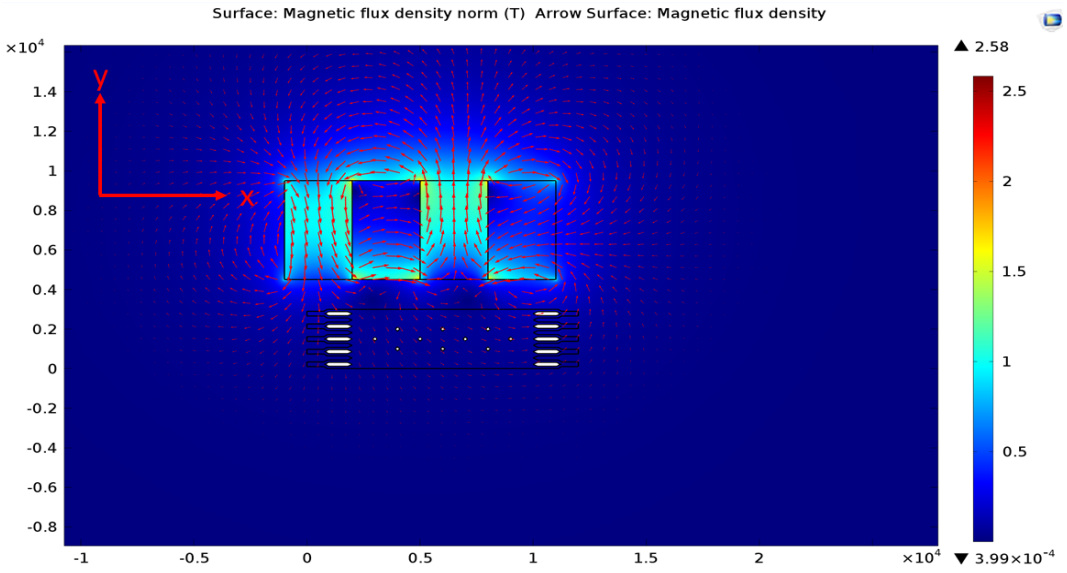


Figure 3.14: Magnetic field distribution and direction across the reaction chamber and surrounding.

In Figure 3.14, the arrows are showing the direction of the magnetic flux distribution and the length of the arrow is proportional to the magnetic flux density at that point. As the magnetic field strength was much smaller within the reaction chamber compared to the inside and edge of the magnets, so the flux lines are not clearly differentiable in Figure 3.14. For this reason, the flux line distribution across the reaction chamber is shown in Figure 3.15.

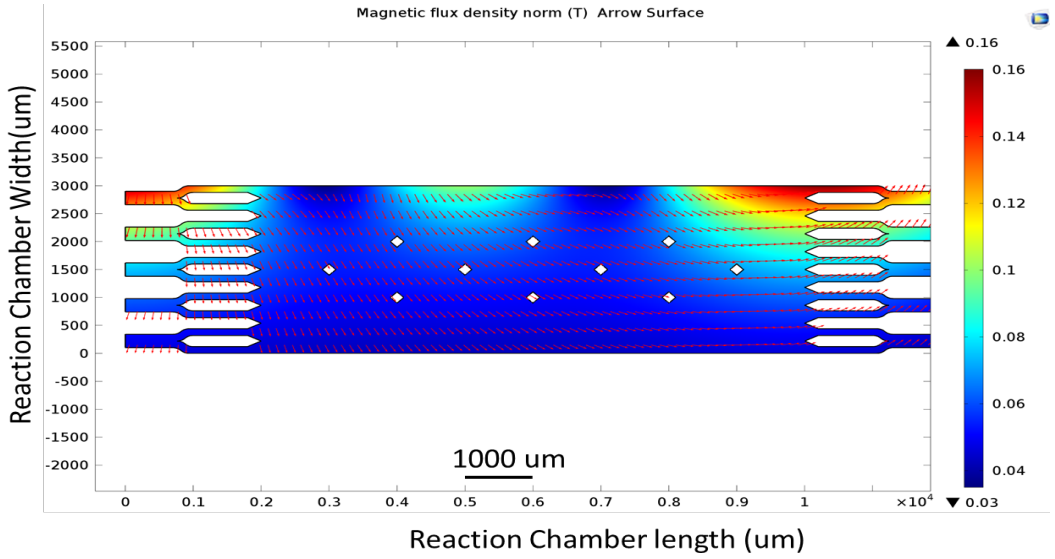


Figure 3.15: Magnetic flux line distribution across the reaction chamber only.

The magnetic field distribution through the central line of the reaction chamber

(shown in Figure 3.10) was considered and is shown in Figure 3.16.

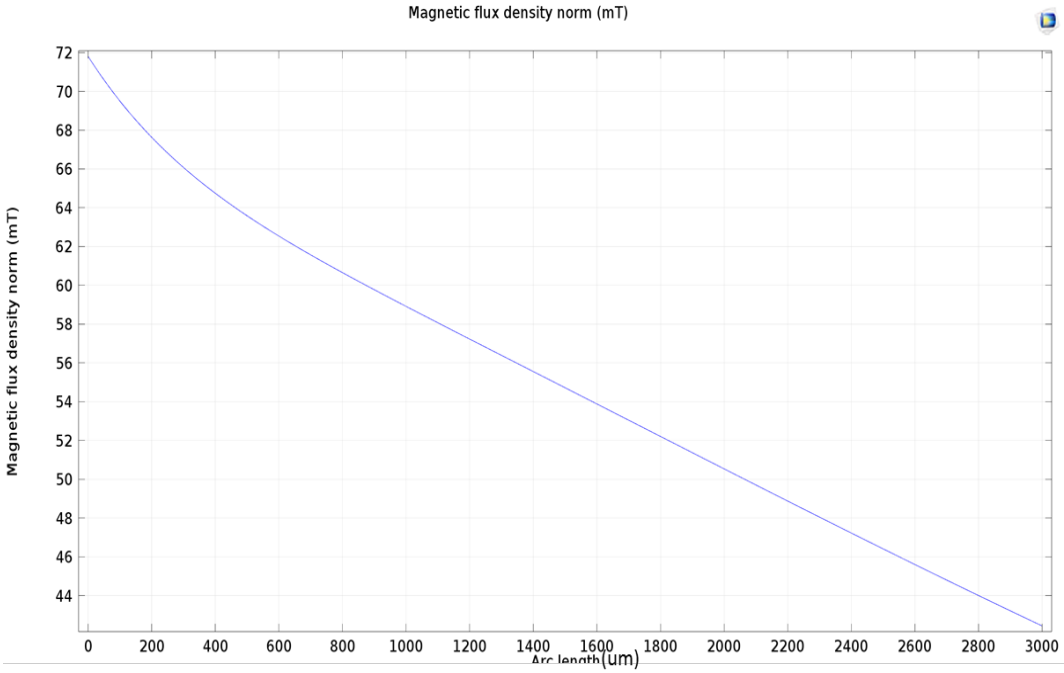


Figure 3.16: Magnetic flux density across the central line of the reaction chamber.

From Figure 3.15, it can be easily seen that the maximum flux density in the reaction chamber (near to the magnet's edge) is 72 mT, which is much lower than that shown in Figure 3.11. When using only one $4 \times 4 \times 5 mm^3$ magnet the maximum magnetic flux density across the reaction chamber is more than 500 mT. The magnetic field gradient across this central line (shown in Figure 3.10) was also calculated and is shown in Figure 3.17:

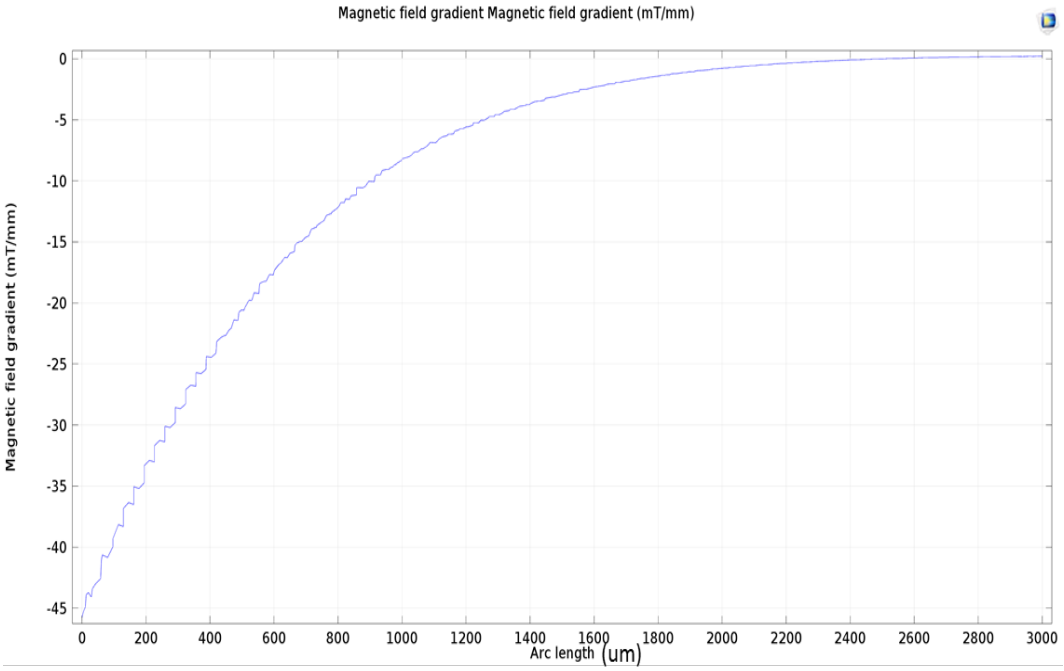


Figure 3.17: Magnetic field gradient across the central line of the reaction chamber.

In Figure 3.17 the field gradient has a negative value, this is due to the directions of the orientation of the magnets in the Halbach array. Here the magnetic field gradient change for Halbach magnetic array was much less (approximately 45 mT/mm to 0 mT/mm) than compare with from 210 mT/mm to 160 mT/mm (Figure 3.12).

Use of Halbach magnet array reduced the change in field gradient, but it also reduced the field strength within the reaction chamber. Reduced field strength will not help the microparticles to reach the desired outlet. It means the use of such magnet orientation cannot provide a suitable solution.

In this situation, after reviewing lots of literature [61, 62, 63] it was found that it is possible to add magnetic flux at the far end of the reaction chamber, which is in the same direction as the applied magnetic field. This is shown in Figure 3.7 and Figure 3.8. This approach can reduce the rapid reduction in magnetic field gradient across the reaction chamber. Considering this instead of using one permanent magnet, two regular shaped $4 \times 4 \times 5mm^3$ magnets were used for the new arrangement. In this simulation, one of the magnets was placed on the top of the reaction chamber and the other was place on the bottom. In both cases, the magnets were placed

500 μm from the edge of the reaction chamber. The new set-up of the magnet and reaction chamber is shown in Figure 3.18:

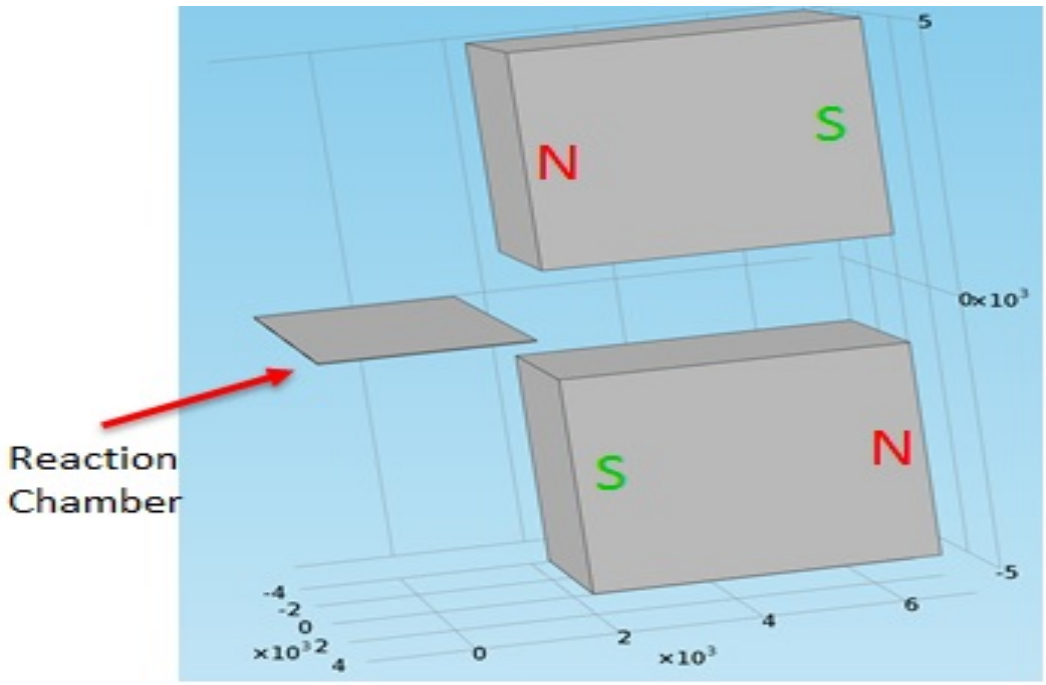


Figure 3.18: Two of $4\times4\times5mm^3$ permanent magnet placed in upper and lower part of the reaction chamber.

For this simulation only the magnetic field distribution was of interest. It was required to do all the calculations in three dimensional space, so to reduce the complexity the inlets and outlets form the reaction chamber were removed. The Upper magnet was magnetized in the same direction as before, this is from North to South pole direction. However, for the magnet in the lower part of the reaction chamber the magnet was magnetized from South to North pole direction. Using this letup, the magnetic field distribution across the reaction chamber is shown in Figure 3.19.

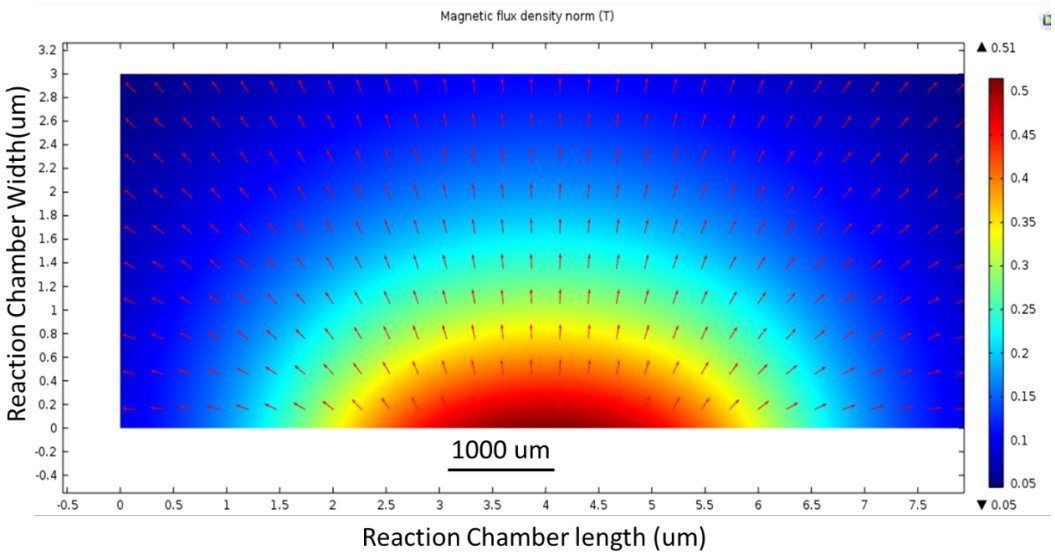


Figure 3.19: Magnetic flus distribution across the reaction chamber for using two permanent magnets.

Like previous simulations in Figure 3.19 the arrows are presenting the magnitude and direction of the magnetic flux line on the point. Now the magnetic field distribution through the central line of the reaction chamber was considered and shown in Figure 3.20:

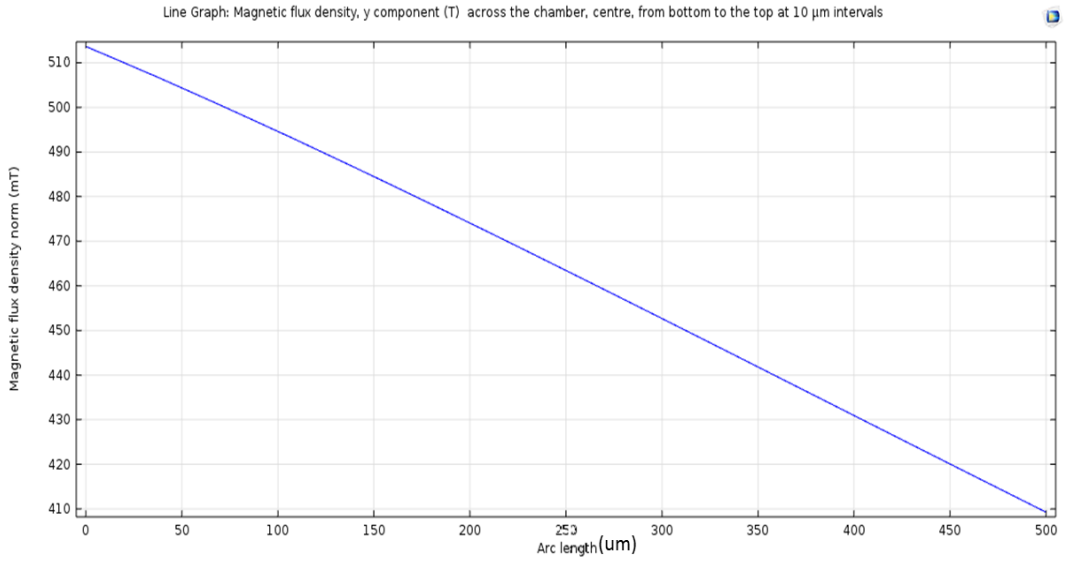


Figure 3.20: Magnetic flux distribution across the reaction chamber for using two permanent magnets.

From Figure 3.20, it can be seen that the magnetic flux density reduced from 510 mT (approximately) to 410 mT (approximately) from the nearest point of the magnet to the end point across the central line (shown in Figure 3.10). This re-

duction is also linear. Here the magnetic field gradient for this set-up is shown in Figure 3.21.

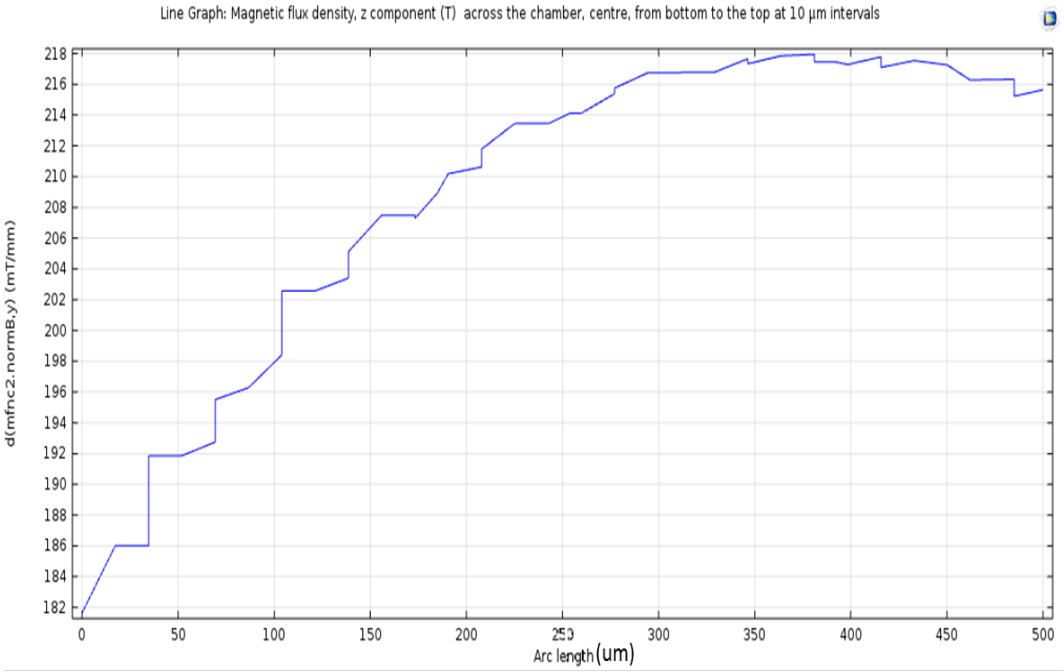


Figure 3.21: Magnetic flux density across the central line of the reaction chamber.

From Figure 3.21 it can be seen that the magnetic field gradient has a range of 218 mT/mm to 181 mT/mm. Though this set-up did not provide a purely homogeneous magnetic field gradient, the range of change is much less compared to both the previous set-ups. Moreover, this experimental set-up is possible to use in real-life magnetophoresis experiments, as both the magnets used are widely available.

3.2 Summary

This chapter and the previous chapter were concentrated on the physics of the LOC experimental set up of interest in this work. In this chapter we have demonstrated that simulation can be used to optimise the physical design, in particular with respect to the magnetic field configuration.

The magnetic field has a significant impact on the motion of the magnetic par-

ticles and this movement or more specifically its automated tracking and the measurement of the moving particles fluorescent intensity using image processing will form the majority of the remainder of this thesis.

The topics are linked in two ways. Firstly, success of the measurement process depends on the quality of movement of the particles. Problems such as particle aggregation which may degrade tracking and measurement are more likely the less homogeneous the field gradient. Secondly physics simulation may eventually be able to assist with the tracking process.

In the next chapter the coverage of image processing is addressed by providing background on the key issue of automatically detecting the presence of particles in images of the chip.

Chapter 4

BACKGROUND SUBTRACTION REVIEW

For every image (including single frame and sequence of frames), the pixel alignment with object feature (s) can be characterised in two ways pixels corresponding with interesting or significant visual information are regarded as "Foreground (FG)" and the rest of the image is "Background (BG)". Depending on the application, it is possible to have more than one foreground for an image. But it is quite impossible to provide any universal standard definition for foreground or background, so that comparing with that standard any part of the image can be marked as foreground or background. The definition of background and foreground will always be application specific and user dependent [64].

For most motion detection and tracking applications (like human action recognition [65], semantic indexing of video [66] , on-line discovering of unusual activities [67] and many more), it is required to detect all moving objects from the entire video. One of the most common approaches for such application(s) to understand the movement and changes in object properties is to remove the non-moving/nonintersecting part(s) from input image(s). To do so commonly used techniques are background subtraction [25, 68], temporal frame differencing [69] and optical flow [70]. Out of all these techniques Background subtraction is the most widely used approach for object

detection and foreground segmentation from static scenes, due to its low computational cost [71] and easy to implement nature. Through Background Subtraction it is possible to "remove" the background from a monitored scene by describing a suitable model for the background. This results in leaving only "interesting objects (foreground)" in the scene for tracking and further analysis [72].

A large body of literature exists for background subtraction. However, the effectiveness of any given technique is dependent on image characteristics, so implementation is not straight forward. This chapter reviews the literature on background subtraction and discusses the applicability of various techniques to the objective of this work.

The most basic background subtraction technique is to subtract every current frame from the previous one (subtraction of previous pixel value from current one on the same position at two consecutive frames) and compare the result with a predefined threshold. This basic background subtraction technique can be illustrated using the following figure shown in Figure 4.1:

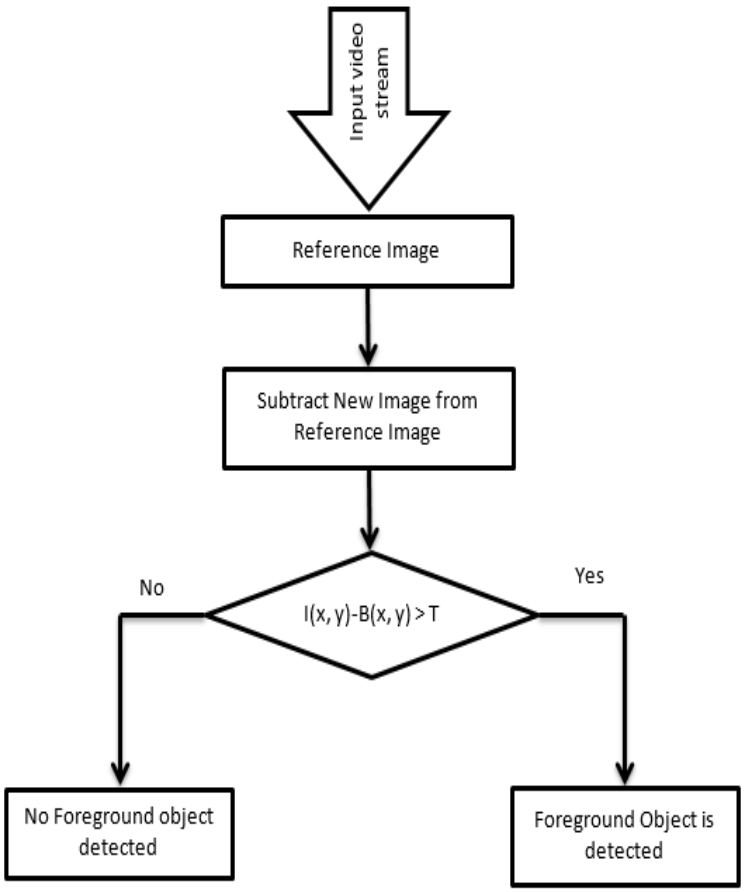


Figure 4.1: Basic Background subtraction Technique.

If the absolute difference of a pixel value $P(x, y)$ at (x, y) between the current frame $[I(x, y, t)]$ and a reference frame $[B(x, y, t-1)]$ is higher than the threshold value (T), then that pixel becomes a part of the foreground, otherwise it is considered as background. So mathematically:

$$mask(x, y) = \begin{cases} foreground & \text{for difference} \geq \text{threshold} \\ background & \text{otherwise} \end{cases}$$

Here "mask" is the output segmented binary image [73] highlighted with the foreground and dark background and "difference" is the absolute pixel value difference between current frame and previous reference frame. But this method cannot provide adequate results [74], because of the discontinuity produced at the edges of complex objects (objects without having regular shape or not having the same illu-

mination over the whole object or changes of illumination over the scene gradually). Moreover, pixel by pixel subtraction also produces high noise (example shown in Figure 4.2). It means for identifying or tracking any complex object(s), the pixel by pixel subtraction approach is not a good choice.

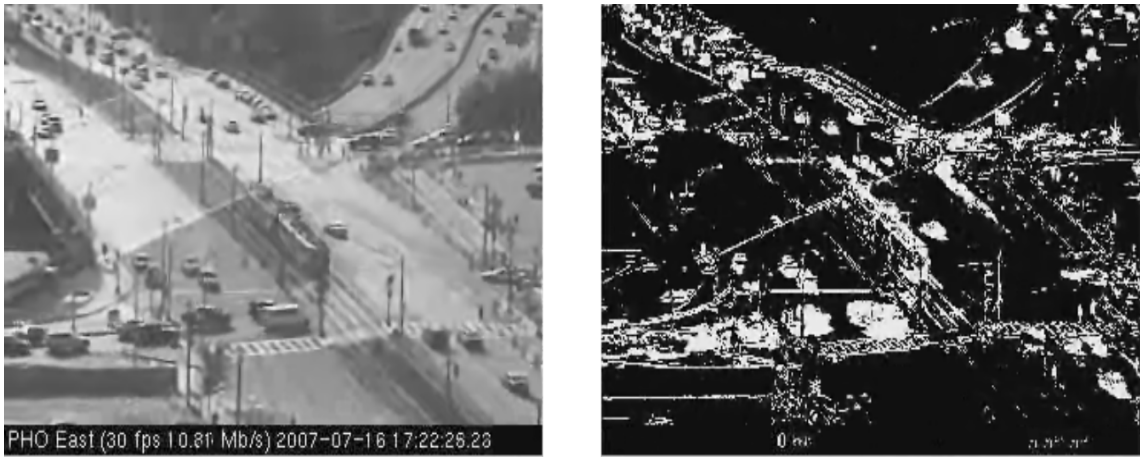


Figure 4.2: Basic Background subtraction using frame difference. Subtracted output contains lots of noises and discontinuities [75]. This discontinuities and noise made the situation impossible to detect any object or feature.

Background subtraction relates the positions of objects in different locations in different frames and it is always wanted to achieve a great rate of movement detection with a very low false detection. Also the outputs from background subtraction become an input for higher level processes (like tracking or recognition of people, vehicles in automated video surveillance systems or molecules in the (bio)chemistry laboratory). This means reliable and accurate background subtraction can affect the output of next level of processing heavily [76]. So background subtraction needs to be reliable and effective, which makes it possible to identify the non-moving objects, stationary or non-stationary background and different patterns of object motion accurately and efficiently [77]. The robustness, performance and accuracy of background subtraction mainly depend on how the subtraction process was modelled considering complex situations. So, rather than depending on a stable background environment, a robust background modelling is required that is flexible enough to handle all kinds of environmental changes including lighting changes, unwanted moving elements of the background scene, objects shadows, slow moving objects *etc.* For this reason, many background subtraction techniques had been developed to identify

the foreground and remove the background (including noise) in an effective way considering several complex situations. These techniques include several morphological and mathematical operations, which are required to be performed before, after and during the subtraction process. These types of background subtraction techniques are also known as Hybrid Background Subtraction techniques. One simple example of a Hybrid background subtraction technique is shown in Figure 4.3.

Figure 4.3: Background subtraction with application of several statistical and mathematical operations (Mixture of Gaussian) on 112th frame of the PETS sequence. Comparing Figure 4.2 and Figure 4.3 it can be seen that the amount of noise and discontinuity is less compared to the background subtraction using frame difference only shown in Figure 4.2. But still many discontinuity and noise can be seen in Figure 4.3, which can lead to erroneous detection (both positive and negative) of object(s).

To develop a system for automated micro particle detection and tracking for analysing Lab On Chip (LOC) experimental results, background subtraction is the first important approach, as subtracting background from each frame of the sequence allows the micro-particles to be detected. The micro-particle itself does not have any significant colour property, its grey scale value (widely called as particle intensity) changes when it passes through the fluorescent stream (see Chapter 1, section 1.2) for multi-laminar flow platform based LOC experiments. This grey scale value or intensity of the micro particle also has a direct relationship with analyte concentration. For analysing the captured images from experiments there was not much colour difference between background and particles, so that the colour cannot be used as a distinguishable property between foreground and background. The only property that makes the micro-particles differ from the background is their small

difference in grey scale value (intensity) between the particles and background. Also any diffusion of the florescence stream can change background properties which may lead to noise in subtracted image. Moreover, the ability of the user, the stability of camera and camera focus produce an erroneous detection (both positive and negative). So the implementation of background subtraction is not straightforward and lots of challenges are associated with it. Some difficulties and challenges related with detecting micro particles from multi-laminar flow microfluidic LOC experiments are discussed in the next section.

4.1 Problem Analysis and challenges

For a background subtraction algorithm to be robust and affective it is requiring to cope with many challenges and difficulties, which can arise during the capture or processing of input sequence(s). In general, three main assumptions were made for developing any background subtraction technique for detecting micro particles from florescence based magnetophoresis particle experiments, they are the capturing camera is fixed, object value (illumination or colour or any considerable property) is constant and the background is static (*e.g.*, the pixel model distribution is the same over the whole scene and no background object is changed, including insertion or moving out of any object). In most of the cases these conditions do not remain constant throughout the whole process, which makes the background subtraction difficult. Similar to many surveillances related approaches, images from multi-laminar microfluidic flow produce many challenges, which are not effectively solved using existing background subtraction algorithms. Before approaching any solution, it is required to analyse these challenges. With reference to the classification produced by Toyama *et al.* [78] and Bouwmans [79], challenges for subtracting background for multi-laminar flow based microfluidic experimental images are discussed in the following section:

4.1.1 Noisy image

Noise can arise both during image capture (*e.g.* sensor noise) and image processing (*e.g.* compression artefacts) which results in problems for object edge detection and image segmentation. Camera set-up and quality can affect the amount of noise as well.

One of the main focuses for this thesis is to detect objects in the image sequences that are taken during multi-laminar flow microfluidic LOC experiments. During the capture of these images some ideal conditions for production of high quality images are the capturing surroundings need to be dark (*e.g.*, no external light should be present), images should be taken from the part of the microfluidic chip where diffusion does not make an effect and camera focus and position are constant. Unfortunately, due to other practical considerations most of the time it is not possible to focus the camera on a position where diffusion do not occur and is not affected by external light. Noise and other artefacts exist for most of the multi-laminar flow platform LOC experimental results. Considering all such situations, one of the main concerns for the introduced background subtraction algorithm during this research was to deal with noises and artefacts.

4.1.2 Changes in illumination

The lighting environment of background can have gradual or sudden changes over time. Outdoor light is an example for gradual illumination changes, which varies during the day. Such gradual illumination change does not seriously affect background subtraction algorithms. However, sudden light changes (like lighting up a scene using an external light source) can affect the background model strongly and leads to false positive object detection [80].

During the multi-laminar flow experiment microparticles travel through various lamina flows including florescent stream(s), so the background illumination of the particles changes over time. Often these changes in illumination cause particles to be detected as noise (discussed as particle intensity issue in Chapter 1, Section

1.2). In this way changes in illumination make the detection of a particle complex. Moreover, depending on the location of the captured image in the microfluidic chip it is possible that the diffusion of a stream can affect the background illumination by changing the lighting condition of the overall capture (the stream itself exhibits florescence). Dealing with illumination effect(s) due to background illumination change and removing effect of diffusion of flow during the background subtraction was a challenging issue.

4.1.3 Bootstrapping

For modelling a background quickly and accurately, it is desired that the background subtraction model is initialized when no foreground object is present on the reference frame. But initial frame(s) without foreground objects are not available in many cases, *i.e.* before the background modelling starts foreground object(s) already exist in the reference frame. Hence, a bootstrapping strategy is often an important part if a background subtraction system is to quickly initialize object tracking [81].

For the best and most efficient micro particle detection results from multi-laminar flow experimental images, it is desired that during the capture of the images the first few frames do not contain any micro particles. In this way during the modelling of the background no foreground object will be included. But in typical situations and depending on the experimenter's experience in most of the cases it was found that micro particle(s) were present at the very beginning of frame capture. This made the modelling of the background during the training period challenging. So bootstrapping was a significant issue for analysing these microfluidic experimental results.

4.1.4 Camouflage

When a foreground object's pixels have similar brightness to the background model, it is difficult to differentiate them from background. In this situation camouflage occurs during removal of background from the image sequence. Camouflage prob-

lems can be divided into two types, they are- dark camouflage and light camouflage. When foreground object's pixels are darker than background model, then it is considered as dark camouflage. Otherwise, it is regarded as light camouflage [82].

When analyte concentration is low, the light from the particle may become too low to be detected (discussed as particle intensity issue in Chapter 1, Section 1.2). In this way the particle may become invisible or part of background at that time. This is a camouflage problem, which makes the particles absence from the foreground. Moreover, the analysed images were taken from a narrow focal plane close to the surface of the chamber, not throughout the depth of the chamber. This way the vertical position of micro particle within the chamber can also produce camouflage effects. Considering these situations, camouflage was a considerable issue for analysing these experimental images.

4.1.5 Foreground aperture

If the object does not have significant difference to the background, it becomes difficult to detect its movement, as the changes in pixel value become too small to be detected. Therefore, the entire object or part of the object can appear as background [78]. This problem can be overcome by using a background maintenance approach (discussed in Section 4.2.3 in this chapter), like used in the Gaussian Mixture Model (GMM) background subtraction algorithm [83].

Similar to the camouflage effect, foreground aperture also produces challenges for analysing the images from multi-laminar flow experiments. This problem becomes very challenging when the images were captured for an experiment with low analyte concentration.

4.1.6 Sleeping foreground object(s)

One of the initial assumptions for an object to be a part of foreground is it should keep moving. But in many cases an object which was initially part of foreground (*i.e.* moving object) can stay on the same position for a few consecutive frames,

or for all the rest of the frames. For such cases, many background models fail to distinguish this object from background, which causes false negative detection of objects. For such situations, these motionless foreground objects need to be handled using a different approach, rather than considering only their motion [84].

During magnetophoresis experiments it is common for some particle to become motionless typically sticking to the reaction chamber wall or a pillar *etc* (Discussed in Chapter 1 Section 1.2). In this way when a particle does not move for couple of frames it becomes a part of the background. Such a visible micro particle will stay undetected when using many conventional algorithms for background subtraction. In this way for developing a robust technique for detecting micro particles automatically, sleeping foreground object is a considerable issue.

4.1.7 Movement in background object(s)

An object which was initially considered as background can start moving during the processing. This movement can be very small or considerable. For a robust and effective background model this newly moved object would not be considered as foreground. But for many background subtraction techniques both the initial and new positions of the newly moved object(s) are detected separately as different object, which produces false positive detection. Such false positives are one of the big challenges for background modelling approaches. These types of detected objects are also known as "ghosts".

For microfluidic experimental result analysis, colour properties of the reagent stream are considered as part of the background and this can change with the change in reagent property or velocity (*i.e.* diffusion). During the capture of the result from multi-laminar flow based microfluidic experiments, the chemically activated microparticles affect the stability of the background, which in turn makes a "waiving tree" effect on the background. So it was required to consider the problems caused due to movement in background.

4.1.8 Occlusion and Clutter

Occlusion (both partial and full) can provide both false positives and false negatives observation, which can seriously affect the efficiency of detection capability of any object detection technique. In real life, occlusion is very common and it can happen at any number of frequencies.

Cluttered images or those images having too many foreground objects to detect sometimes make the task of image segmentation difficult. In this way it becomes challenging for a model to separate the foreground object from the background one.

Unequal forces to pull the magnetic particles toward a specific direction cause each particle to move at different velocities (Discussed in Chapter 1 Section 1.2). This difference in velocity is one of the main reasons for making clash between particles. These clashed particles may remain together permanently till the end of the process or may separate at a later time. Such a situation happens very frequently during the experiment. In this way occlusions between particles are a very common challenge for analysing and detecting images from LOC experiments.

4.2 Background subtraction steps (BG modelling steps)

Many background subtraction techniques have been proposed for different computer vision related applications. Most of these background subtraction techniques follow four major steps- pre-processing, background modelling, background maintenance and foreground detection [85]. Pre-processing consists of collecting the simple images and processing them into a format which can be easily read and processed by the subsequent steps. Background modelling then utilizes this new format input video frame for statistically calculating and updating the entire background scene. Subsequently, foreground detection identifies the pixels which are not included within the background scene and exports these pixels as a foreground mask. Background maintenance examines these foreground mask candidate pixels with the actual moving

objects to eliminate the false detection of pixels [85]. These background subtraction steps are shown using Figure 4.4.

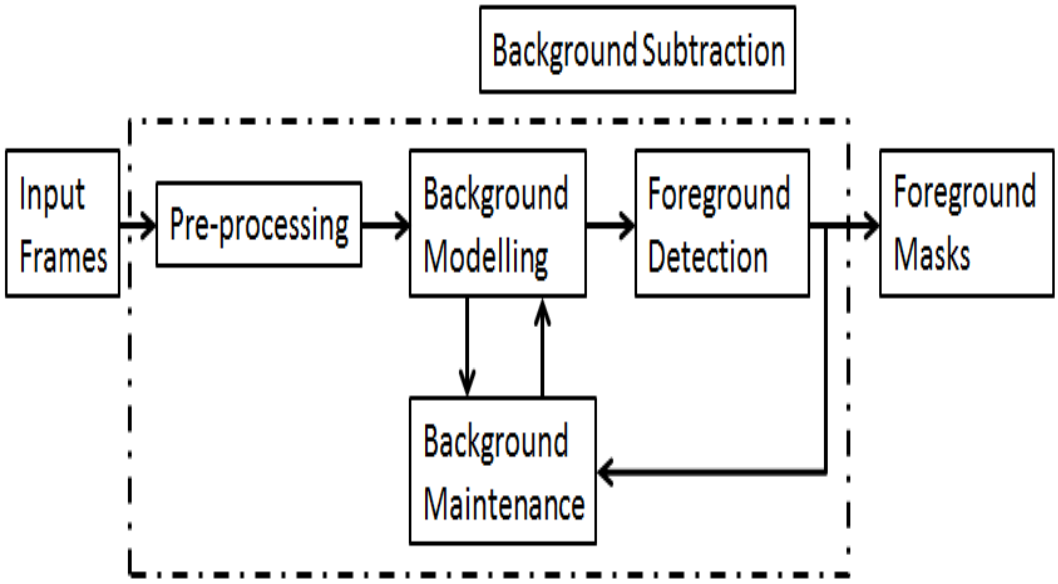


Figure 4.4: An overview of background subtraction steps.

These four steps are a continuous and repetitive process and they depend on each other. All these four major steps for background subtraction are discussed in detail as follows:

4.2.1 Pre-processing

Pre-processing of the raw input frames is the first task for background subtraction. During this step the image data format is converted or changed (like RGB to HSV or RGB to Grey image *etc*) to a format, which is suitable for the particular background subtraction technique. Pre-processing is an user dependent process, this means it depends on the input image characteristics and desired target, pre-processing is done. For most of the background subtraction algorithms, grey scale images are used as input. Grey scale image provide a scalar value for each pixel, so it requires less computational power for any approach to analyse them. Moreover different types of image smoothing (temporal and/or spatial smoothing) and/or other mathematical operations are also done on the image to reduce camera noise [85] at this stage.

4.2.2 Background modelling

Background modelling is the most important step of background subtraction. This is the algorithm or approach used for modelling and representing the background. This model continuously decides how the background representation will adapt under many complex conditions (described in the previous section of this chapter "Problem Analysis and challenges", in Section 4.1 in this Chapter).

Stable and simple backgrounds can be represented by simple methods, like via an average of pixels using grey scale or colour representation of the scene. However, for most practical videos, the background scenes are not stable. They contain several complexities like - waves, tree moving, light changes, busy roads *etc.*

Many studies have already been done related to background modelling for obtaining a robust method to deal with variety of complex environments, but not many different fundamental background models (most of them are mainly a modification or improvement or combination of fundamental models) has been developed. For most of these approaches, at the beginning of the process the background model is obtained by training it using a short sequence of frames which contain no foreground objects. But in many practical situations it is quite impossible to obtain sequences of frames without any foreground objects [78].

4.2.3 Background maintenance

Background subtractions are used for many long term observation applications. During this long term observation, the background can change (for example background illumination can severely change), foreground object(s) can become a part of background, a background object can have movement [67, 78] *etc.* In these situations, if the background model remains the same as the initial model, it will cause false positive or negative detection. Therefore, the background model needs to be updated and to adapt over a long period of operation. This update in background model for adapting with changes is known as background maintenance. Accurate and efficient background maintenance improves object tracking quality and reliability as

well. There are a few different types of update mechanisms available for updating background models with new frames, like conservative (selective) update [86], blind update [87].

The conservative update mechanism never includes a sample which belongs to a foreground region inside the background model. In this way a pixel sample can only be included in the background model if it has not been classified as a foreground sample. For this approach it is considered that background and foreground do not have similarity in colour. The most useful outcome from conservative update mechanism is it provides a sharp detection of the moving objects. But if a background sample is incorrectly classified as foreground, it prevents the background model from updating the pixel model. In this way the background pixel model can stay without being updated indefinitely and can cause a permanent misclassification. Many practical applications lead to such situations. As an everyday example a car park can be considered. If a parking location where a car was previously parked cannot be included within the background model by using a purely conservative update scheme. This update mechanism can lead to deadlock situations and everlasting "ghosts". But use of a separate update mechanism can handle such situations.

In the blind update mechanism pixel samples are added to the background model even when they have not been classified as background [86]. So this mechanism is free from pixel deadlocking problems. But the main problem for this update mechanism is for slow moving target(s), where the detection rate is low. In this method a slow moving target can be considered as background in the background model over a period of time.

4.2.4 Foreground detection

Foreground detection is the final step of background subtraction, which compares the input video frame with background model and identifies the foreground pixels. The most commonly used algorithm of foreground detection is to use the difference between the pixel values from the input video frame and the background model, as

Equation (4.1):

$$|I_t(x,y) - B_t(x,y)| > T \tag{4.1}$$

or the relative difference comparing input pixel and background model [88], as Equation (4.2):

$$\frac{|I_t(x,y) - B_t(x,y)|}{B_t(x,y)} > T \tag{4.2}$$

or determining the foreground threshold as Equation (4.3):

$$\frac{|I_t(x,y) - B_t(x,y) - \mu_d|}{\sigma_d} > T \tag{4.3}$$

Where $I_t(x,y)$ and $B_t(x,y)$ are the input and background pixel value respectively, T is the threshold, μ_d is the mean of $I_t(x,y) - B_t(x,y)$ and σ_d is standard deviation of $|I_t(x,y) - B_t(x,y)|$ [85].

In addition, some foreground detection methods introduce two thresholds [89, 90], one is the lower and the other is the upper threshold value. Two threshold foreground detection is useful when the noise in the sequence is regular and the objects contain very significant features.

4.3 Background Modelling Techniques

Out of several basic background subtraction techniques, Gaussian Mixture Model (GMM) proposed by Stauffer and Grimson [68] and single Gaussian model are considered as two of the most widely used ones. Gaussian Mixture Model is a pixel-based algorithm based on forming a statistical background model for each pixel separately. This model is found to be robust against gradual changes in illumination and slow movement in background regions. However, for any sudden illumination change in background and if a foreground object remains static over a long period of time this model fails. GMM also produces poor segmentation results for many complex

real world conditions having camera noise, complex edges and high frequency background objects (like tree branches, sea wave *etc*). Moreover, implementation of a fundamental GMM approach proposed by Stauffer and Grimson has a relatively high computational cost.

For monitoring many simple scenes the single Gaussian models is used for foreground segmentation. However, single Gaussian model is found effective when background is constant and no noise is available to consider. Also implementation of a single Gaussian model requires very low computational processing. But this model fails for any complex background model.

Two commonly used methods (*i.e.* single Gaussian model and Gaussian mixture model) for background modelling are described in detail in the following section.

4.3.1 Single Gaussian Model

For an input scene with static background, single Gaussian method models the changes of each pixel of a background scene independently by a single Gaussian distribution. For modelling the background (B) for a pixel (x_t) in a frame, this model uses the previous n pixel values of the same pixel as samples for describing a Gaussian probability density function using a mean (μ) and standard deviation (σ) as Equation (4.4):

$$(x_t|B) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x_t-\mu)^2}{2\sigma}} \tag{4.4}$$

For producing Gaussian probability distribution for each pixel it is required to store all n previous intensity values of them for every frame for calculating the mean (μ) and standard deviation (σ), which requires large computational memory. Requirement of large computational memory for single Gaussian model was later solved by Wren *et al.* [25] by introducing a running average approach for finding mean and standard deviation of the pixel intensities for adapting the background changes which requires less computational memory. The running average approach

for updating the background model can be described as follows Equation (4.5):

$$B_t = (1 - \alpha)B_{t-1} + \alpha I_t \tag{4.5}$$

Where $t \geq 1$, α is the pre-determined learning rate, B_t is the background image computed up to frame t, I_t is the current pixel intensity value and B_{t-1} is the previous running average for the computed background image.

The other parameter, *i.e.* standard deviation (σ) of the Gaussian model, can also be updated in a similar way using the same learning rate α, as shown in Equation (4.6):

$$\sigma_t = (1 - \alpha)\sigma_{t-1} + \alpha \left(I_t - \mu_t\right)^2 \tag{4.6}$$

Introduction of running average allows the model to compensate small changes in lighting condition and object movement, as gradual light change has minimum effect on the running average. Also with the use of running average approach, it requires only one mean and one variance to be stored for each pixel of the background model, which lead to lower memory requirement.

After modelling the background using Gaussian distribution, it requires a simple thresholding to classify a pixel as foreground or background. The classified result is then represented using a binary foreground mask $FG(x, y)$ as:

$$FG(x, y) = \begin{cases} 1 & \text{if } |I_t - B_t| > k\sigma_t \\ 0 & \text{otherwise} \end{cases}$$

where k is a constant and usually it is 2.5 [91], I_t is the pixel intensity of the current frame, B_t is the running average of the background image obtained using Equation (4.5), and σ_t can be obtained from Equation (4.6).

The single Gaussian method provides a robust result against small or gradual variations in the background dynamics of a scene, but this method fails when there are large or sudden changes in the background illumination to make effect on the running average.

4.3.2 Gaussian Mixture Model (GMM) Background Modelling

When there is large or sudden illumination changes in the background scene, a single Gaussian model is not enough to describe the scene dynamics, it requires a multi-model distribution to describe it. Stauffer and Grimson [92] proposed a Gaussian mixture model (GMM) to address this problem, which represents a background scene using multi-modal distributions. Multi-model distribution means multiple Gaussian distribution. Gaussian mixture model (GMM) is one of the most widely used and reliable basic background modelling techniques when input frames are from a static or non-moving camera (*i.e.*, this technique is suitable for modelling a static background and gradual light change in background). Gaussian Mixture Model (GMM) for background modelling is also widely known as Mixture of Gaussian Models (MOG). This algorithm consists of three basic assumptions (1) foreground is detected by excluding the background, not depending on foreground texture, colour or edge model (2) per pixel based processing is required rather than using region based processing and (3) the decision for background model comes from every frame, without requiring any feedback or tracking information [93]. Moreover, in this model it is considered that background is more visible than any foregrounds and it has less frequent change in background.

Each pixel value on an image changes over time. So if a pixel X is a part of a moving object (foreground) on an image at time t, it can be a part of shadow or background on other frame at time $(t + \Delta t)$. This way for a pixel (X) located at (x_0, y_0) can be considered as time series of pixel values, called 'Pixel Process'. So for a pixel X at (x_0, y_0) the pixel value history over time t can be represented as Equation (4.7):

$$\{X_1, X_2, X_3.....X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\} \tag{4.7}$$

Here I is the image sequence and i is representing time. Using this pixel history from recent frames (training frame), the pixel features can be modelled by a mixture

of K Gaussian distributions. In this way after modelling a mixture of K Gaussian distributions, the probability ($P(X_t)$) of observing the current pixel value X_t is Equation (4.8):

$$P\left(X_t\right) = \sum_{k=1}^{K} \omega_{i,t} * \left(X_t, \mu_{i,t}, \Sigma_{i,t}\right) \tag{4.8}$$

Here, K is the number of the Gaussian distribution, $\omega_{i,t}$ is an estimate of weight (portion of the data is accounted for by its Gaussian) of i^{th} Gaussian in mixture at time t ($\Sigma\omega_{i,t} = 1$), $\mu_{i,t}$ is the mean value of the i^{th} Gaussian in the mixture at time t, $\Sigma_{i,t}$ is a diagonal covariance matrix of the i^{th} Gaussian in the mixture at time t, η is a Gaussian probability density function as Equation (4.9):

$$\eta\left(X_t,\mu_t,\Sigma\right) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(X_t-\mu_t)\Sigma^{-1}(X_t-\mu_t)} \tag{4.9}$$

For making the computation easier it is considered that the pixel values are independent and have same variance of σ. So with identity matrix I the covariance matrix become as Equation (4.10):

$$\Sigma_{i,t} = {\sigma_{i,j}}^2 I \tag{4.10}$$

Using the above procedures each pixel is characterized by a mixture of K number Gaussians. An example of pixel characterization with $K = 3$ number of Gaussian can be represented using Figure 4.5:

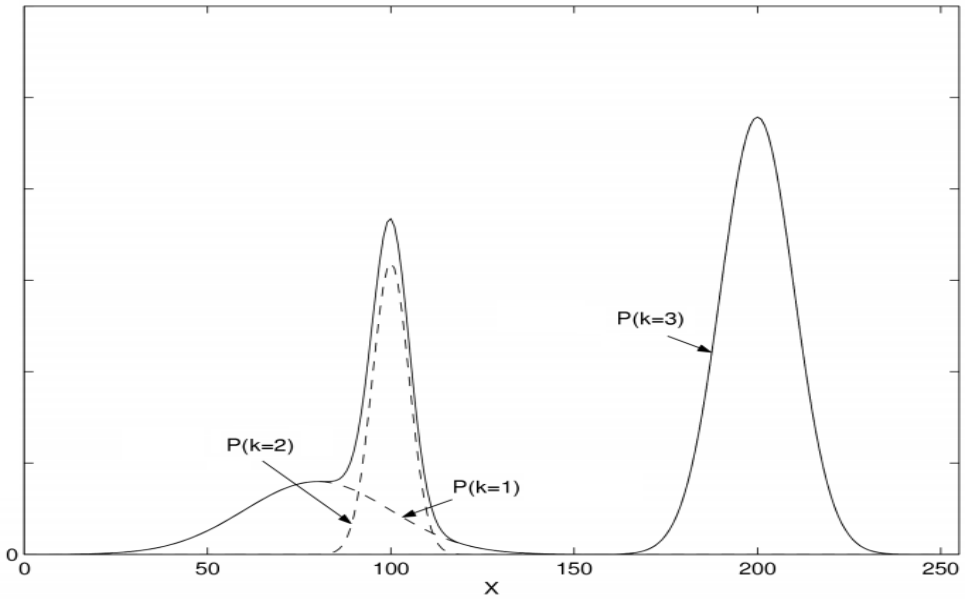


Figure 4.5: The pixel value probability $P(X_t)$ from Equation (4.9) is illustrated for 1D pixel values X ∈ 0,1,2,.255 with K=3.

In this stage it is required to initialize the parameters of the models, this means for each of the number of Gaussians K, the weight $\omega_{i,t}$, the mean $\mu_{i,t}$ and covariance matrix $\Sigma_{i,t}$. The value of K is chosen depending on the computer processing power and memory, normally it is between 3 and 5 [92]. For initializing the other values a computationally costly expectation maximization (EM) algorithm provides a better result (as it does soft clustering assignment using Bayesian classifier probabilities), but for real time implementation a K-mean algorithm (hard clustering assignment) provides faster results [92]. The more the number of modes are, the more the model performs well for detecting objects. With the increase in number of modes the complexity and usability reduce, as the whole model requires higher computational power. Also with a higher number of modes the process becomes slower. After the initialization is done these Gaussians are ranked in order and decided which Gaussian is belonging to foreground and which is background. For this the Gaussians are sorted according to order of decreasing number from a ratio $r_j = \frac{\omega_j}{\sigma j}$ for $j \in [1...K]$. Each of these Gaussians are known as 'mode'. As it is already mentioned in the assumption for GMM method that backgrounds are more frequent than foreground, so a background pixel will have higher weight and lower variance.

In this way higher importance goes to background components, which means background Gaussians will be on the top of the sequence. Now to decide which Gaussian is belonging to foreground, a threshold T_B is used. If a Gaussian Distribution B exceeds this threshold T_B then these $1B$ Gaussian components belongs to background, mathematically it can be expressed as Equation (4.11):

$$B = \arg\min_b \sum_{i=1}^{b} \omega_{i,t} > T_B \tag{4.11}$$

The rest of the distribution components $(B+1)K$ are considered foreground.

After the parameters are initialised and distributions are divided into foreground and background categories, the first detection of foreground is made by comparing a new pixel value (X_t) with the existing K Gaussian distributions. Normally it will match with one of the major components of the mixture model. For finding this match the absolute difference (known as Mahalanobis distance) between current pixel value and initialized mean (μ_j) is compared with the predefined threshold value (ς). This predefined threshold (ς) is 2.5 times of standard deviation [87], means mathematically it can be shown as Equation (4.12):

$$||X_t - \mu_j|| < 2.5 * \sigma_j \tag{4.12}$$

After this comparison the parameters for mixture of Gaussian are updated as in the following Equations (4.13), (4.14) and (4.15)

$$\omega_{i,t+1} = (1-\alpha)\omega_{i,t} + \alpha M_{k,t+1} \tag{4.13}$$

$$\mu_{i,t+1} = (1-\rho)\mu_{i,t} + \rho X_{t+1} \tag{4.14}$$

$${\sigma_{i,t+1}}^2 = (1-\rho){\sigma_{i,t}}^2 + \rho(X_{t+1} - \mu_{i,t+1})(X_{t+1} - \mu_{i,t+1})^T \tag{4.15}$$

Here, α is a constant learning rate for mode's weight and ρ is learning rate for

the Mean, Variance and Prior estimates Equation (4.16),

$$\rho = \alpha\eta\left(X_t|\mu_k, \sigma_k\right) \tag{4.16}$$

Here in Equation (4.13), $M_{k,t+1}$ is equal to 1 for matching component of j and 0 otherwise. If there is no matching found, the component with the lowest weight ω_k is re-initialized with $\omega_k = \omega_0$, $\mu_k = X_{t+1}$ and $\delta_k = \delta_0$.

Drawbacks

The implementation of the GMM by Stauffer and Grimson [92] was not efficient enough to deal with many complex situations, which requires improvement. Before any step(s) were taken for the improvement, it was necessary to understand the drawbacks of the system by Stauffer and Grimson [92]. So analysing the Equations (4.13), (4.14), (4.15) and (4.16) the observations can be summarised as follows:

1. The weights of the Gaussian modes change very slowly with time constant (constant learning rate) and it is roughly $\approx \frac{1}{\alpha}$ [93]. So the background modes do not change rapidly, as the weights change slowly. This (very) slow change in weights, lead the method proposed by Stauffer and Grimson to lack adaptation to any fast changes in environment.

2. The mean and variance updates in Equation (4.14) and (4.15) are independent of the weight values, so the changes in mean or variance do not make an effect on weight of 'modes'.

For solving these issues, the first improvement was done by Lee. Instead of a fixed learning factor, he introduced an adaptive update factor which can adapt with change in situation quicker, when any new change appears [94]. This learning rate can be shown as Equation (4.17).

$$\rho = \frac{1-\alpha}{c_k} + \alpha \tag{4.17}$$

Here, c_k is a counter which is maintained independently for each mode. Its value is initialized to 1 for a new mode and is incremented whenever a match with an incoming pixel occurs.

Though there is an introduction of variable learning factor by Lee [94], but still there were problems left. The GMM model modified by Lee is highly sensitive to the environment change, as the adaptive learning factor Lee introduced was the same for updating both the Mean (μ_k) and the Variance (${\sigma_k}^2$) of the model. Here the Mean value (μ_k) influences the sensitivity of detection and Variance (${\sigma_k}^2$) significantly influences the speed for the model to adapt with the change in environment. So having the same learning factor for updating Mean (μ_k) and variance (${\sigma_k}^2$) cannot be efficient. So, it was required to find suitable learning factors for Mean (μ_k) and variance (${\sigma_k}^2$).

Before the proposed approach for improvement is discussed, it was required to understand how this learning factor affect any system. For analysing the factors affected from the learning rate, at first the minimum time (t_{min}) for a Gaussian distribution from a part of K-number of Gaussian mixture to become a part of the background model was found. From Equation (4.13) it can be seen that the weight of any Gaussian changes depending on time and so for different times with a constant posterior probability q_k as it can be expressed as:

$$\begin{aligned}
\omega_k(t) &= (1-\alpha)\omega_k(t-1) + q_t \\
\omega_k(t-1) &= (1-\alpha)\omega_k(t-2) + q_{t-1} \\
\omega_k(t-2) &= (1-\alpha)\omega_k(t-3) + q_{t-2} \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\omega_k(1) &= (1-\alpha)\omega_k(0) + q_1
\end{aligned}$$

Using the above expressions, we can express $\omega_k(t)$ from Equation (4.13) as a function of $\omega_k(0)$ and for simplicity the new Equation (4.18), was written without the subscript k:

$$\omega(t) = (\omega(0) - q)(1-\alpha)^t + q \tag{4.18}$$

where $\omega(0)$ is the initial value of $\omega(t)$. If time t_min is the minimum time required for $\omega(t)$ to reach or exceed a particular value ω_{min}, then Equation (4.18) can be expressed as inequality shown in Equation (4.19):

$$\omega(t) = (\omega(0) - q)(1 - \alpha)^{t_{min}} + q \geq \omega_{min} \tag{4.19}$$

Let's consider a mixture of K Gaussians, with K_B Gaussians belonging to the background. Also assuming that their respective weights are ordered, *i.e.*, $\omega_1 \geq \omega_2 \geq ..... \geq \omega_{K_B} ..... \geq \omega_k$. If the weight $\omega_{K_B} = \omega_{min}$ is the minimum weight of the Gaussian components for belonging to the background. Then, using the definition of a Background Gaussian shown in Equation (4.20):

$$\sum_{i=1}^{K_B} \omega_i < T_B \Rightarrow \sum_{i=1}^{K} \omega_i - \sum_{i=K_B+1}^{K} \omega_i < T_B \tag{4.20}$$

Since, $\sum_{i=1}^{K} \omega_i = 1$, so Equation (4.20) can be written as Equation (4.21)

$$\sum_{i=K_B+1}^{K} \omega_i \geq 1 - T_B \tag{4.21}$$

Minimizing the weight ω_{K_B} imposes that $\omega_{K_B} = \omega_i \ , \forall i \geq K_B$. Therefore,

$$\sum_{i=K_B+1}^{K} \omega_i = (K - K_B)\omega_{K_B} \geq 1 - T_B \tag{4.22}$$

Now if $\omega_{K_B} = \omega_{min}$, then, Equation (4.22) become:

$$\omega_{min} = \omega_{K_B} \geq \frac{1 - T_B}{K - K_B} \tag{4.23}$$

Solving the inequality in Equation (4.19) yields

$$t_{min} \geq \frac{1}{\ln(1 - \alpha)} \ln \left[\frac{\omega_{min} - q}{\omega_0 - q} \right] \tag{4.24}$$

Now replacing the value of ω_{min} in Equation (4.24) from equation (4.23), we get,

$$t_{min} \geq \frac{1}{\ln(1 - \alpha)} \ln \left[\frac{(1 - qK) - T_B + qK_B}{(\omega_0 - q)(K - K_B)} \right] \tag{4.25}$$

For the Equation (4.25) assuming ω_0 have very small weight ($\omega_0 = \alpha$), for an exactly matched Gaussian $q = 1$ and bi-modal background $K_B = 2$ then the Equation (4.25) become:

$$t_{min} \geq \frac{1}{\ln(1-\alpha)} \ln \left[\frac{K + T_B - 3}{(K-2)(1-\alpha)} \right] \tag{4.26}$$

So, the minimum time required for any model to become part of background can be expressed by Equation (4.26).

Using the equation (4.26), it can be found that for a constant value of T_B equal to 0.7, learning rate $\alpha = 0.005$ and $K = 3$ (mixture of 3 Gaussian) the background will adapt after 70 frames. For $K = 5$, the same background will adapt after 31 frames. The change in learning rate makes a lot of difference to the model to adapt with the change in background. If the learning rate is changed to 0.003, then for the same T_B it will take 117 and 34 frames to adapt for $K = 3$ and 5 respectively. From the above discussions it can be clearly seen how the learning rate effect the total adaption process.

4.3.3 Frame difference

For the conventional two frame differencing background subtraction method, the method calculates the differences of two adjacent frames $f_(k-1)(x, y)$ and $f_k(x, y)$ for getting the region of moving object(s). This process can be shown using the Equation (4.27):

$$D_k(x, y) = |f_k(x, y) - f_{k-1}(x, y)| \tag{4.27}$$

After getting the absolute differences between two adjacent frames, it is required to set a threshold value (T) for getting a binary image ($B_k(x, y)$) from the difference image ($D_k(x, y)$). When pixel values of the difference images are more than the predefined threshold value, the pixel is considered as a possible part of the object pixels; otherwise as a background pixel.

$$B_k(x,y) = \begin{cases} 1 & D_k(x,y) \geq T \\ 0 & D_k(x,y) < T \end{cases}$$

This two frame difference moving object detection method can be shown using the following Figure 4.6.

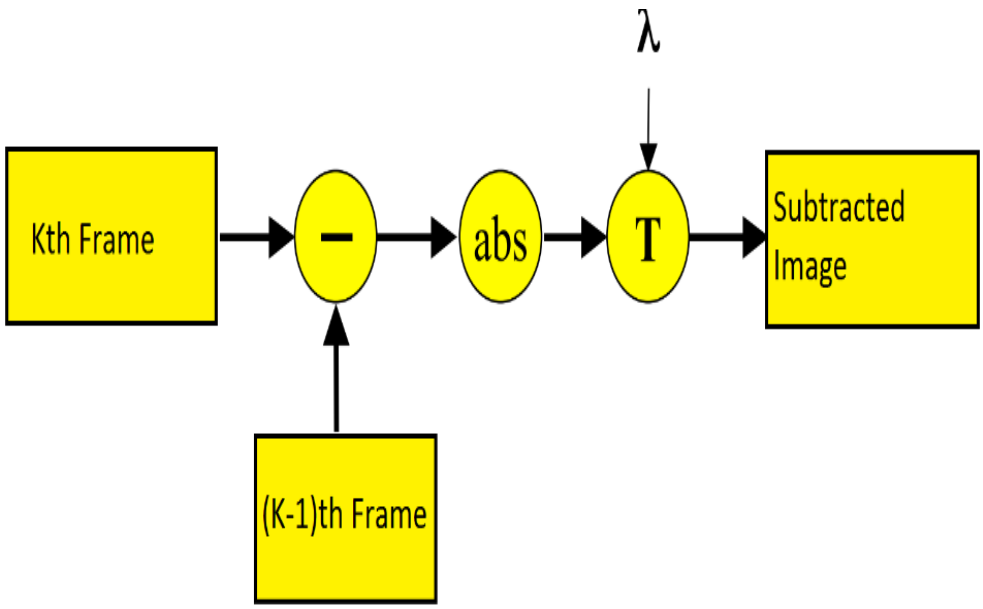


Figure 4.6: Two frame Difference background subtraction method.

At this point it is required to remove some object(s) smaller than the real object(s). It is done by checking the pixel connectivity on the binary image $B_k(x,y)$ and setting a threshold size for the object(s). While the size of the connected pixel area is more than the given threshold size, it is considered as a moving area for the object; otherwise it becomes a background area.

There are many disadvantages for this method, for example I) for an object moving at extremely low speed this method fails to detect it, II) if the inner grey part of the object is relatively homogeneous, then this method can not detect the overlapping part and this overlapping part in the moving object become an "empty" hole [95], III) the resultant difference image can show the range of movement due to the movement of objects in adjacent images, but it's movement extraction area is bigger than the actual movement and often can be "double" the real movement [96].

For real life applications, scholars improved the two-frame difference method. Instead of using two adjacent frames for detecting the object movement, they used three adjacent images for differencing and then making several logical and morphological operation(s) [97]. Also some scholars proposed taking an image block as unit of difference and many more approaches [98] for optimizing the results of detecting moving object.

4.4 Summary

Background subtraction is one of the main initial approaches for developing an automated microparticle detection, tracking and intensity measurement system. Background subtraction provides the information about the particle's position and other movement characteristics which helps the tracker to relate their current position with the previous one. For developing a robust system to detect the objects accurately, several challenges are associated from the input images from the LOC experiments. These challenges were discussed in this chapter. Moreover, lots of approaches have already been made for developing robust background modelling techniques. A few of the basic techniques for background modelling which are related with this project were discussed in this chapter. The following chapter will provide a brief description of the improved combined implementation of these techniques.

Chapter 5

PROPOSED BACKGROUND MODELLING APPROACH

Background subtraction is one of the most widely used methods for detecting moving objects. The main target for background subtraction is to detect foreground objects by subtracting the background from every current frame. This way it is possible to find the target object(s). When a reference background frame without any moving objects is available, it is easy to subtract the background from the current frame comparing it with the reference frame. However, in practice, in many cases it is not possible to obtain a pure background image. So the basic models for background subtraction discussed in Chapter 4 cannot be implemented for most practical scenarios. A few of the most widely used viable background-modelling techniques (Single Gaussian Model, Gaussian Mixture Model, Frame Difference) and some very common challenges related to background subtraction have already been discussed in Chapter 4.

Gaussian Mixture model (GMM) first proposed by Stauffer and Grimson in 1999 [92] uses K-number of Gaussians for modelling each pixel. The basic GMM can cope with some variation in image background caused by movement of background objects and gradual changes in background illumination [92]. However in order to adapt to such changes GMM requires a long period of time for updating the background

(*i.e.* GMMs do not get updated frequently), which is one of the major challenges for any background subtraction method [99]. For solving the update rate problem for background models an improved Gaussian Mixture model was proposed by Lee [94]. However, this improvement requires large computational power for complex scenarios, as a significant amount of calculation is required to initialize K (3 to 5) Gaussian for each pixel. Also for any scene containing slow moving objects, it was found that GMM improved by Lee also fails to differentiate such objects from the background. Moreover, for any object detection situation having camouflage and bootstrapping effects, GMM fails to detect the objects successfully.

The experimenter uses tiff or seq format for saving their experimental results from the LOC magnetophoresis experiments, which produces more information to process compared to widely used avi or mpeg files. Moreover, when the concentration of the analyte is low, the microparticles used in magnetophoresis experiments do not show much colour difference from the background. This mades it quite difficult to use a threshold value to detect any object within the frame, as an inappropriately low threshold can produce high noise and an inappropriately high threshold will cause the object to be detected as background.

The objects detected during magnetophoresis are chemically activated to produce light proportional to analyte concentration, so a moving lighted objects are generally elliptical in shape (due to motion) and becomes circular when not moving. Moving bright objects produces higher intensity light at the centre of the object and reducing in an area surrounding the object. These light intensity changes produce a gradient in grey scale value changes at the edge of the object, which cannot be considered as shadow but which produce a secondary illuminated object layer.

The magnetic particles may adhere temporarily or permanently to the internal surface of the LOC device. Particles may collide and become stuck together. Such joined particles may also become stationary. The analysis systems should be able to identify situations where particles are joined or have become stuck as these may provide misleading results if measured. Large numbers of stuck particles may indicate a system malfunction. Using the conventional GMM method it was not possible to

deal with such non-moving particles. So it was required to have a method which can deal with the problem related to non-moving particles.

In this section, an overview and details of the proposed method for detecting micro particles in the LOC experiments will be provided. The image sequence obtained from the LOC experiments has micro particles as objects that are required to be tracked. They however are very similar in colour to the background, which make the separation of foreground object from background more difficult. The results of object detection of micro particles in the input sequence obtained from LOC experiment will be presented using the proposed methods. Also the results using the proposed methods will be compared with other standard techniques of object detection in terms of object detection accuracy, noise effects and robustness.

5.1 Algorithm Overview

The process of object detection and background subtraction is performed in several steps, specifically there are four major steps I) pre-processing, II) background modelling, III) extracting foreground object features and IV) object detection. Each of these steps also contains several sub-steps. The overview of the proposed algorithm is shown in Figure 5.1.

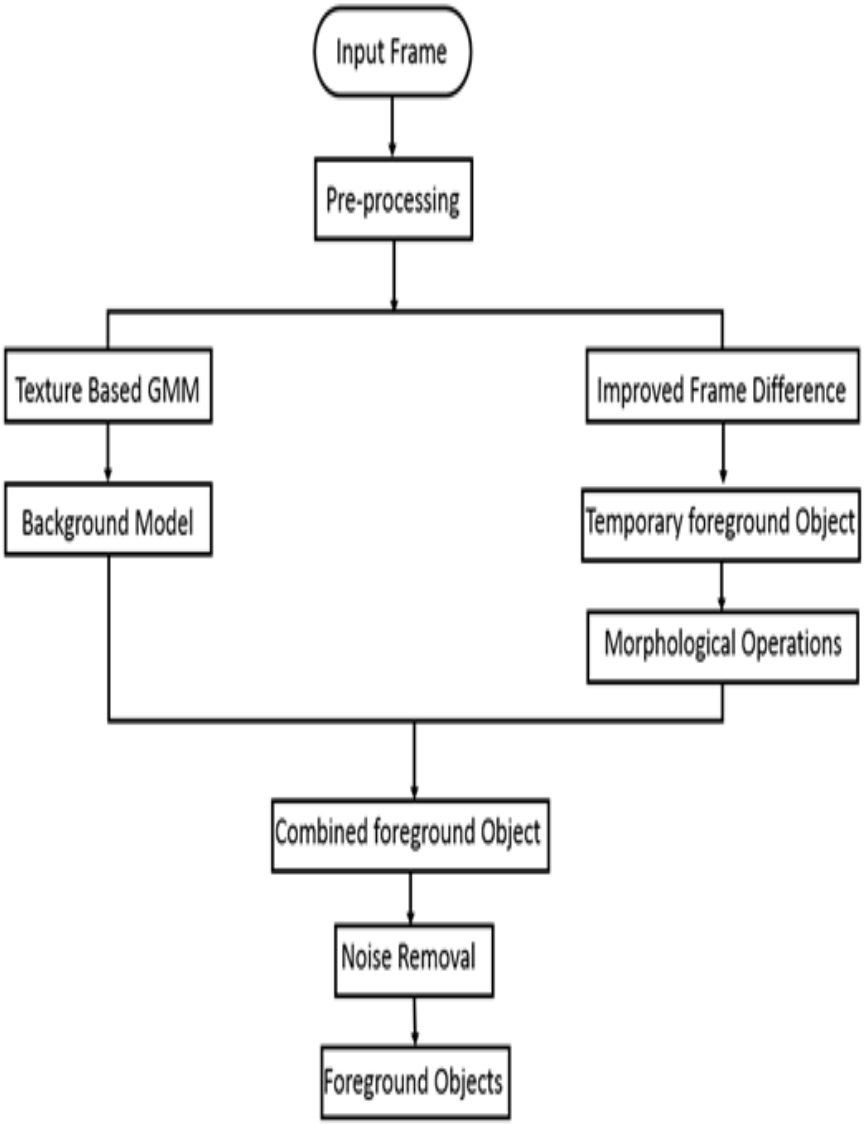


Figure 5.1: Proposed algorithm for foreground object detection.

Out of these four major steps the most significant and important step is background modelling, which in this case is a combination of two different very widely used techniques Gaussian Mixture Model (GMM) and Frame Differencing. Individually each of these techniques have drawbacks, like inability to observe slow moving object(s), finding a whole object without any discontinuity either on the edge or inside the object, adaption with changes in the environment *etc.* For these reasons, to overcome many of these problems, more than one technique was used in combination to produce a robust background modelling and foreground detection

technique. The following sections will discuss in more detail each of the steps and their outcomes.

5.1.1 Pre-processing

Foreground detection from any input sequence starts with pre-processing or preparing the input frames ready for further processing. Pre-processing is a user dependent step for any background subtraction technique, *i.e.,* the specific process depends on the properties of the input sequence and requirements. As the main target for the background modelling in this thesis is to analyse the images from magnetophoresis experiments, so it was found that three step pre-processing was required for analysing those results. These steps are - image conversion, image intensity normalization and removal of noise generated due to image intensity normalization. These pre-processing steps are discussed in details in following sections:

Convert image to Suitable grey scale

The original input image sequences obtained from different chemistry experiments are RGB colour images. Colour image means each pixel of the image contains more than one value (values for red, green and blue). However, colour provides lots of information which is not necessary for detecting the object movement. Moreover, it requires large memory capacity for storing this information and it also requires more processing resources. In this way colour images are not suitable for fast processing. So the first step for this proposed algorithm was to convert the input images to grey scale. In a greyscale image, each pixel contains a single value only and this value contains the information about the brightness of the pixel and no colour information is available. Grey images are composed with shades of gray, varying from black at the weakest intensity to white at the strongest. These shades of gray or brightness of the pixel are usually divided into 256 (0-255) parts, where 0 represents the darkest (black), 255 is for the brightest (white) intensity. As for a grey scale images there is only one value to consider for each pixel, so it was found much easier and

computationally less expensive to manipulate them.

There are a couple of mathematical expressions available to convert RGB input images to grey scale ones. The weighted average method is the most commonly used one for grey image conversion. The weighted average method gives 'Red', 'Green' and 'Blue' different weights according to their importance. This weighted average method for calculating Grey value for any pixel can be formulated as Equation (5.1):

$$Grey_{(x,y)} = w_r \times R_{(x,y)} + w_g \times G_{(x,y)} + w_b \times B_{(x,y)} \tag{5.1}$$

Here, w_r, w_g and w_b are the weight of '*Red*', '*Green*' and '*Blue*' component of the pixel respectively. Experimentally it was found that human eye is highly sensitive to green, then red and finally by the blue component. The same consideration was found for widely known image processing tools like Photoshop, ImageJ *etc.* Moreover, for this project lots of functions were used from a widely used image processing library called 'OpenCV'. In this library, the values for w_r, w_g and w_b were used as $w_r = 0.2989$, $w_g = 0.5870$, $w_b = 0.1140$ and so for this project as well.

Image quality identification

Chemically activated micro-particles get its fluorescent intensity depending on the concentration of analyte and is proportional to the concentrations. As many magnetophoresis LOC experiments were carried out with very low concentration of analyte, so the micro-particles used in those experiments have very low contrast with the background (Figure 5.2).

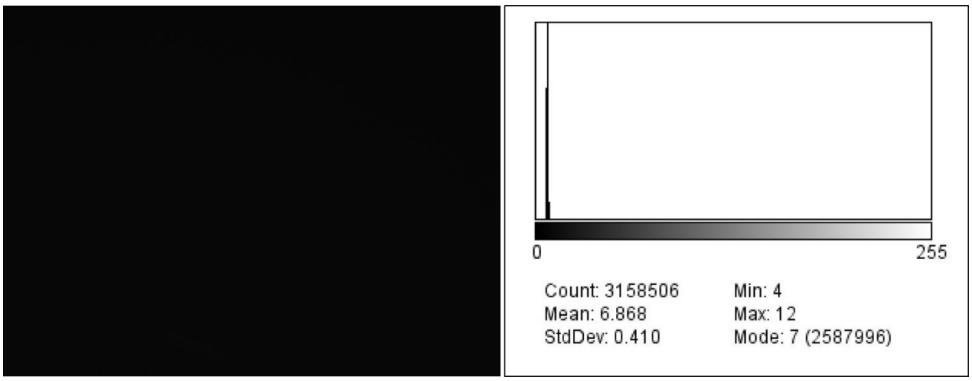


Figure 5.2: Histogram of image with poor contrast. This image frame was taken from one of the DNA Hybridisation experiment done by Dr Martin Vojtek using zero concentration of Analyte (0nM).

Under these conditions it is required to increase the grey scale difference between the foreground (objects) and background. Out of the several methods available such as controlled pixel value multiplier, image linear and non linear histogram normalization, image entropy calculation *etc*, the most efficient one found was linear image histogram normalisation. Linear image histogram normalization was computationally less expensive and also was easy to use. But normalising the image histogram produces salt and pepper noise, which was then removed using a Gaussian filter.

Both these pre-processing processes (histogram normalization and Gaussian filter) make the background modelling computationally expensive. So these two steps are only implemented when essential, *i.e.,* for magnetophoresis LOC experiments with lower concentration. For determining the requirements of pre-processing, it is assumed that there are some microfluidic particles (objects) available within the first three frames (chosen as the number of training frame). Then the grey scale value differences between the pixels were determined. Using experimental result from DNA Hybridization done by Dr Martin Vojtisek [17], it was found that if there is a minimum grey scale value difference of 20 ($\tau_{min} = 20$) between the objects and background then the objects were possible to be detected without pre-processing.

$$\text{Preprocessing requirement} = \begin{cases} 1 & \text{pixel value difference,} \tau \geq 20 \\ 0 & \text{pixel value difference,} \tau < 20 \end{cases}$$

In Figure 5.3, the input sequence (first three frame) from experimental result

with $0nM$ analyte concentration and their 2-D grayscale histogram is shown.

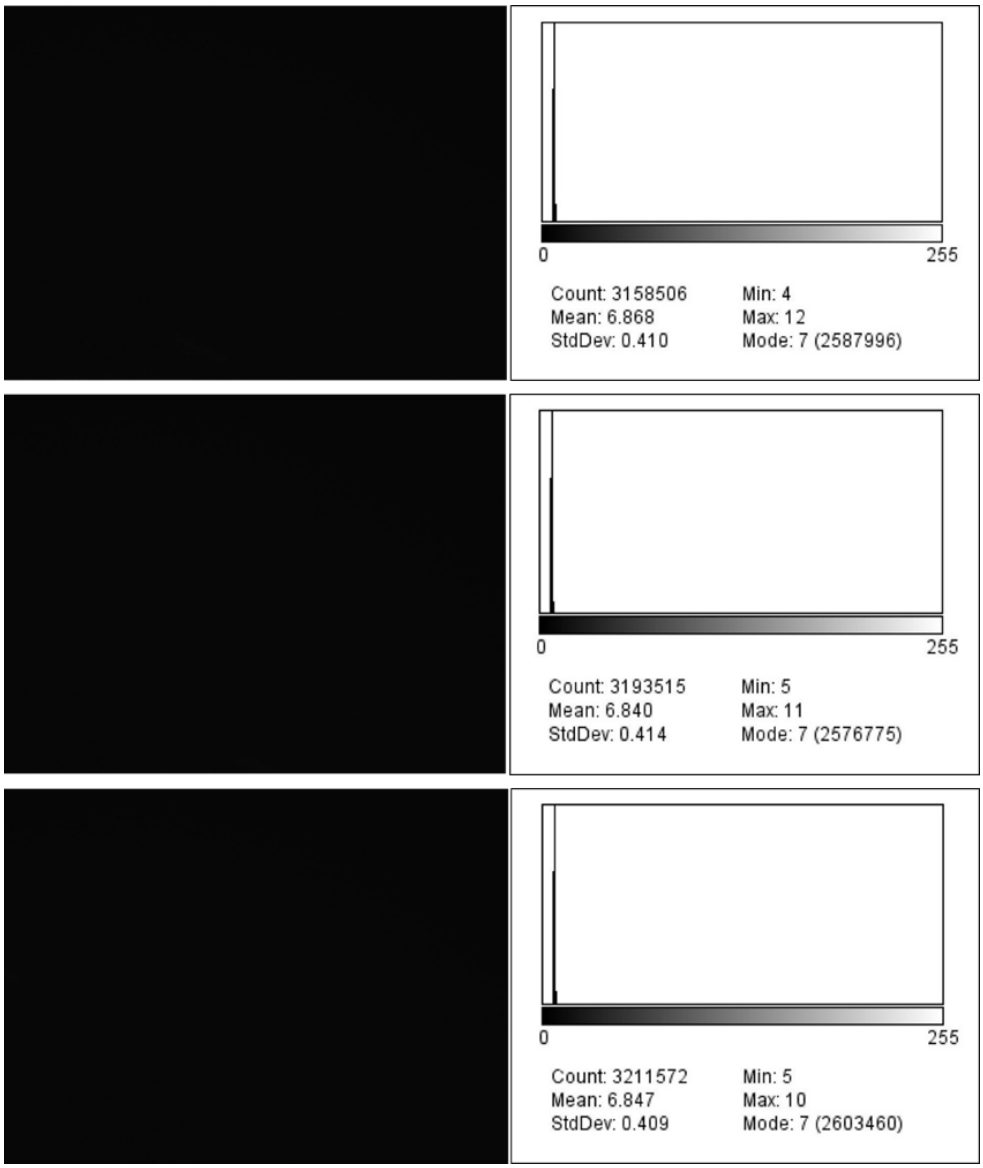


Figure 5.3: The left side images are showing the first three frames from experiment with 0 nM concentration and the corresponding 2D grey scale histograms are shown in the right side graphs respectively.

In Figure 5.3 the objects has very low grey scale value and it is very close to the background pixel's value. After measuring grey values for each pixel in these frames it was found that, most of these pixels' values were varied from 4 to 14. For input frames with such histogram distributions pre-processing steps, like histogram normalization and Gaussian filtering are required.

On the other hand, for the same experiment done with higher concentration of

$20nM$ of analyte (DNA), the grey scale value distribution is shown in Figure 5.4.

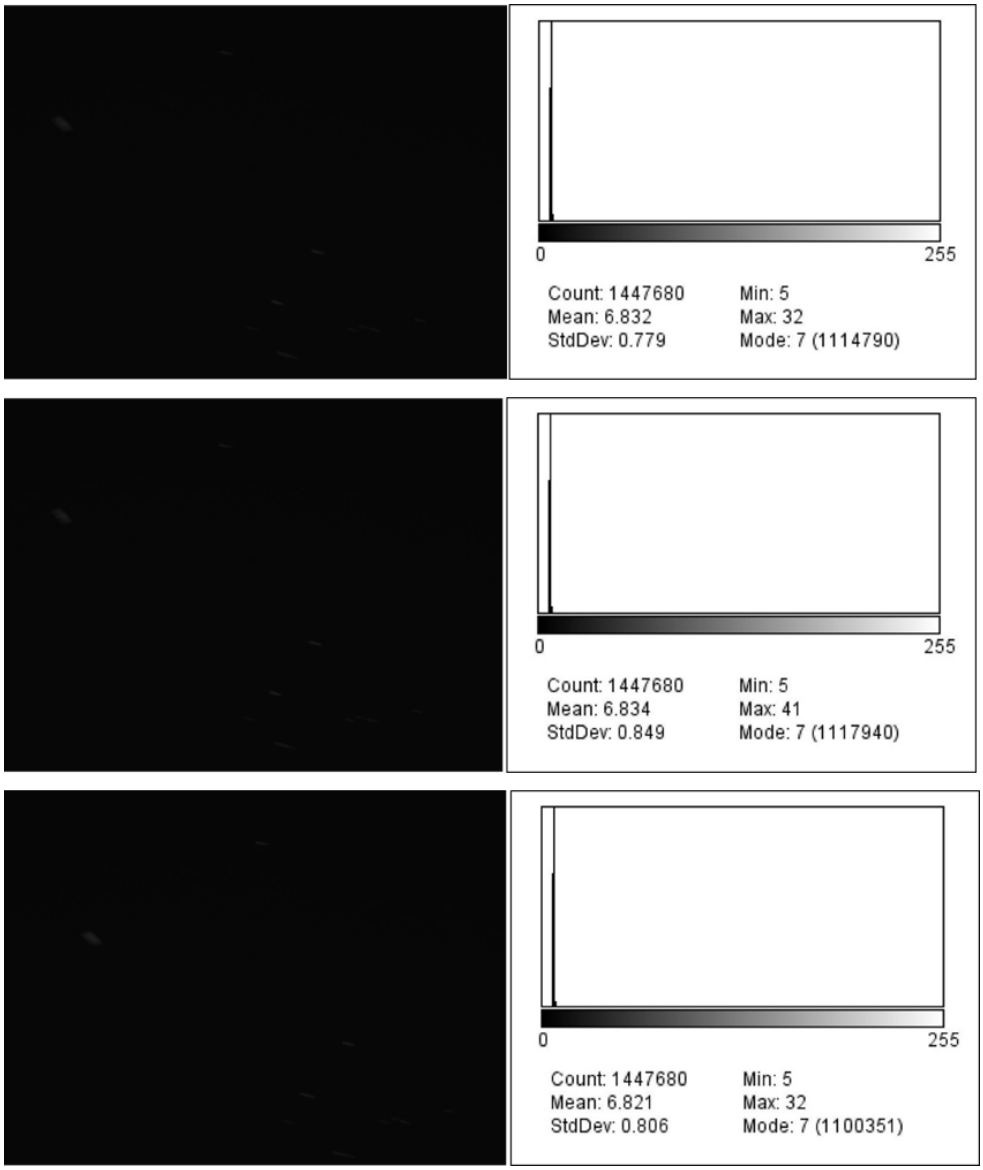


Figure 5.4: The left side images are showing the first three frames from experiment with 20 nM concentration and the corresponding 2D grey scale histograms are shown in the right side graphs respectively. Objects were found in 1st frame.

In Figure 5.4 it can be seen in the histogram distribution, that the object intensities are higher than background and this difference between them is more than "20". Also the objects in this case are more easily visible than those in Figure 5.3. For the input sequence shown in Figure 5.4, the objects are easily discriminable, so it does not require pre-processing. The process of pre-processing can be described using the flow chart shown in Figure 5.5.

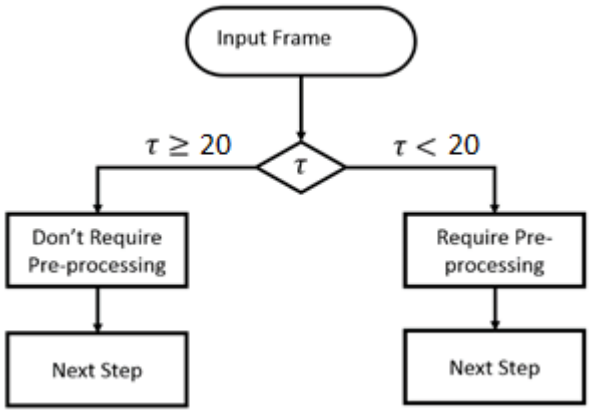


Figure 5.5: Requirement checking process for pre-processing.

Histogram Normalization

Histogram normalization on any image is done when it is required to change the pixel intensity range. Often it is helpful to normalize the image histogram for improving the object and background grey scale value difference(s). Histogram normalization helps improve images with poor contrast.

For LOC experimental result images, foreground object pixels have comparatively higher intensity than the background. Increasing the contrast difference between background and foreground object was helpful for later background subtraction and object detection. Linear histogram normalization was performed on the experimental result images in the pre-processing stage. Through histogram normalization an n-dimensional grayscale image $I : \{X \subseteq R^n\} \rightarrow \{Min...Max\}$ with intensity value of range (Min, Max) is transformed to $I_{new} : \{X \subseteq R^n\} \rightarrow \{newMin...newMax\}$ with intensity value of range $(newMin, newMax)$, using the following Equation (5.2):

$$I_{New} = (I - Min)\frac{newMax - newMin}{Max - Min} + newMin \tag{5.2}$$

Here, I_{New} is the new pixel value after normalization. I is the current pixel value, Min and Max are current minimum and maximum pixel values. $newMin$ and $newMax$ are minimum and maximum pixel values for the new image.

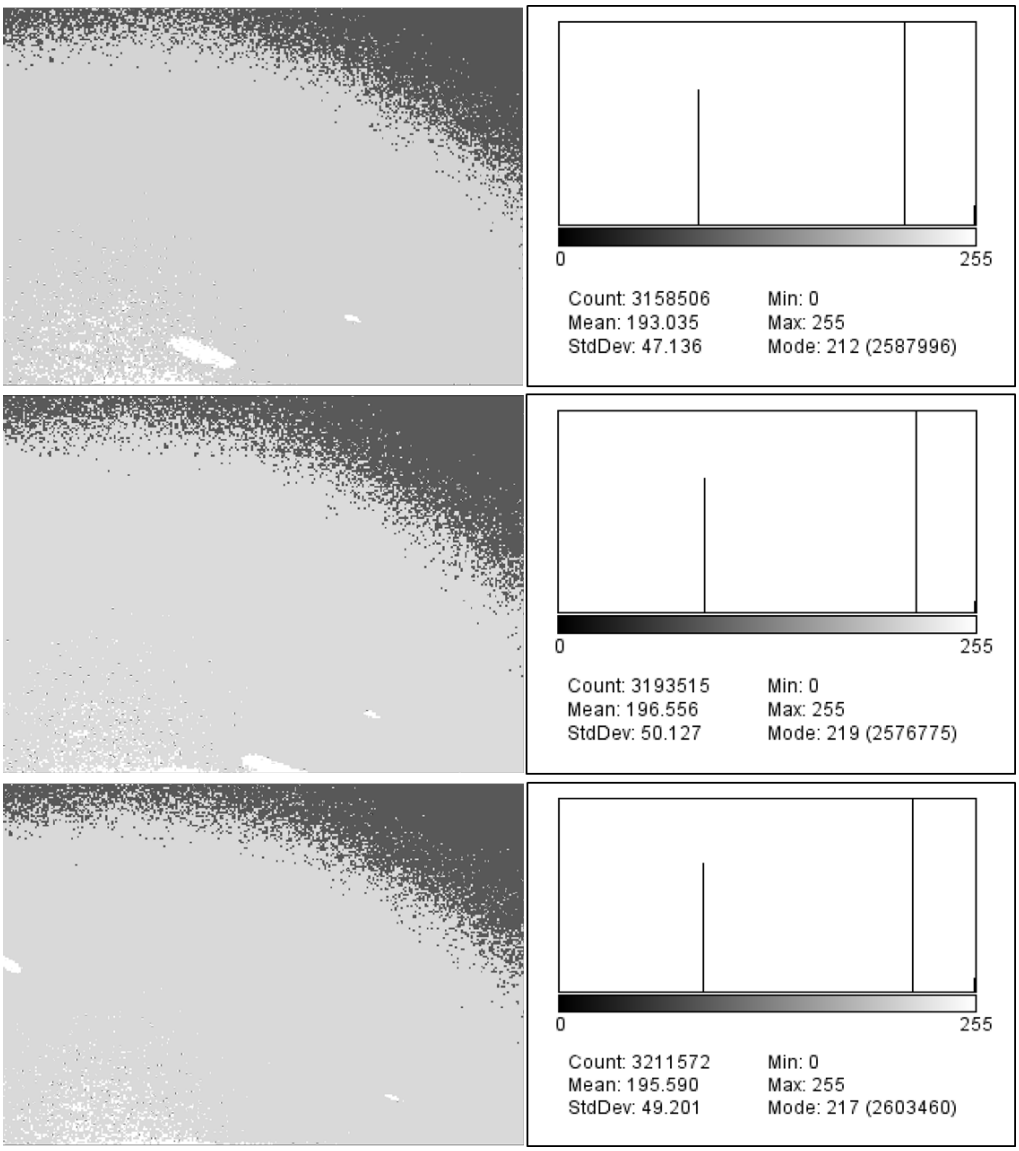


Figure 5.6: Normalised histogram image for the same image in Figure 5.3 with clear object and histogram distribution.

From a poor contrast image (shown in Figure 5.3), it was not possible to identify any object, while, an image with normalised histogram (shown in Figure 5.6) contains obvious objects. After normalization, poor contrast images obtain more contrast and pixel intensity difference is enhanced. But it was useful to check the effect of histogram normalization on an image which was wrongly identified as an image requiring normalization. For this reason, a manual histogram normalization was applied to an image sequence captured from DNA hybridization experiment with analyte concentration of $40nM$, $80nM$ and $100nM$. Figure 5.7 shows the effect of

histogram normalization on an image, where objects were clearly visible.

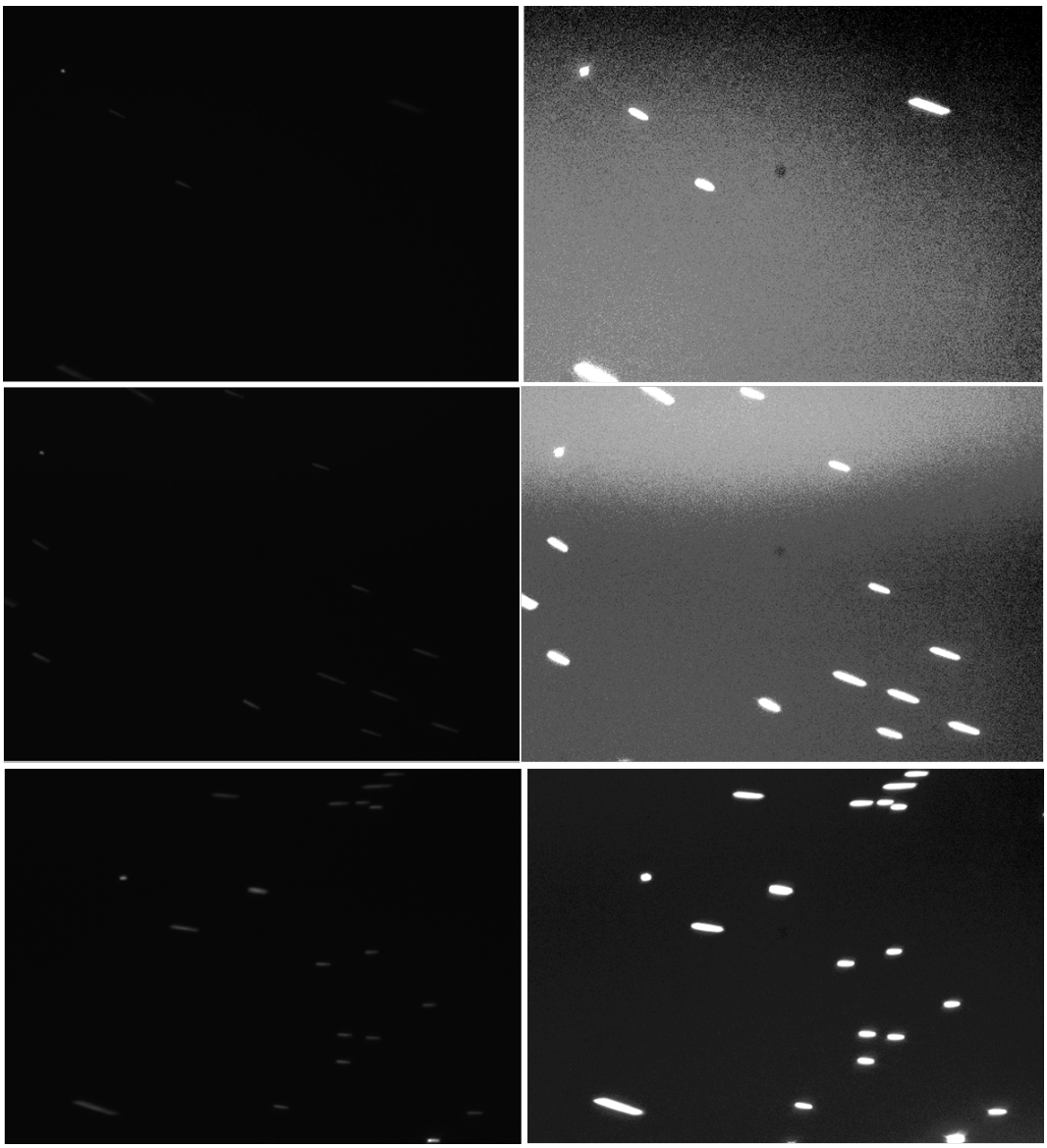


Figure 5.7: Normalization applied to input sequence image from experiment with concentration of 40 nM, 80 nM and 100 nM respectively. Here the left side images are showing the original input and the right sides are normalized images of them.

From Figure 5.7 it can be clearly seen that after applying histogram normalization on an image with clearly visible objects, it did not have any negative effect on it. The images can be used for object detection but effort is wasted pre-processing these images. It should be noted that normalization introduces noise to the image, especially for the poor quality images. This noise issue will be solved in a later stage.

Normalization was also applied in other complex situations (daily life situations, like car movement, human walking) to check whether it caused any negative effect on detecting objects or not. It can be seen in Appendix 1 that after normalization there were no significant problems which affected object detection. Thus we can conclude that normalization was suitable to use in our results to obtain better contrast for later analysis.

Gaussian filter Kernel

Images with normalized histogram typically exhibit high levels of noise, especially for original images with poor contrast ratio. This noise can be categorised as salt and paper noise. Therefore, it is required to filter it to smooth the images for the purpose of avoiding erroneous detection. It was found that implementation of a Gaussian filter is one of the ways to remove most of this noise and smooth the image. Though mean filter was another option for doing so, but the Gaussian filter [100] was used as a filter for the subsequent detection process for its efficiency.

The Gaussian filter operates as follows - if a $(2m+1)\times(2m+1)$ mask is defined with a centre of $(0,0)$ and other (x,y) ranges are from $(-m,-m)$ to (m,m), the 2-D Gaussian filter function element for this mask can be defined as Equation (5.3):

$$(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.3}$$

where $x, y = -m, ..., 0, ...m$ and σ is the standard deviation of the Gaussian distribution with mean is zero. For larger value of σ, the details of the input image get reduced. Moreover, Gaussian blur with a large σ can introduce artifacts to the resultant image. As for the implementation used, it was required to keep most of the detail from the input image, so a small value for σ was chosen. Also a larger mask size for the Gaussian filter makes the process computationally expensive, but a small mask size did not remove most of the noise. So a mask size of (5×5) was chosen. If v is the intensity of the noisy input, then the resultant image ($G_R(x,y)$)

after the Gaussian filter can be shown as Equation (5.4):

$$G_R(x, y) = G(x, y) * v \tag{5.4}$$

Here, $*$ denotes the convolution operation.

After the Gaussian filter was applied on the noisy input image from histogram normalization step, the blurred image acts as the input image for foreground detection. An example image after applying Gaussian filter is shown Figure 5.8.

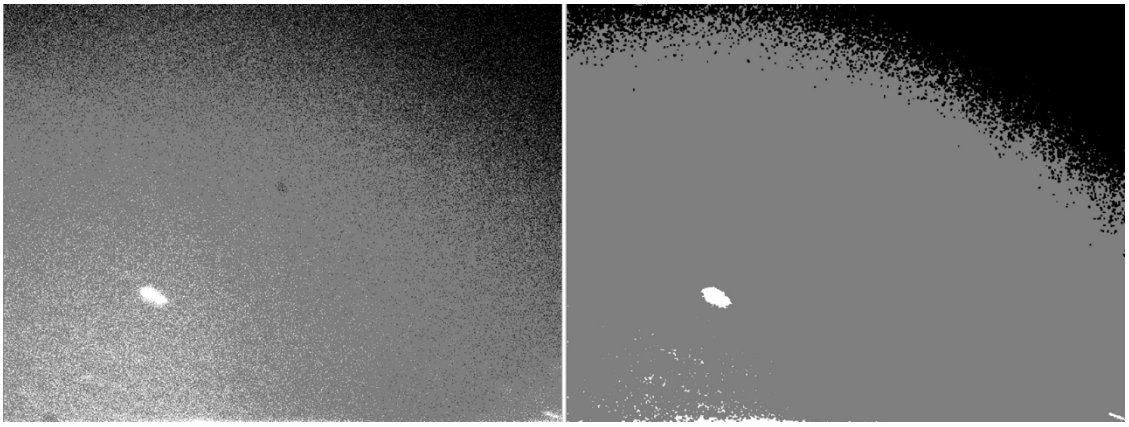


Figure 5.8: Gaussian filter of 5×5 mask was applied.

5.1.2 Background modelling and foreground detection

After the pre-processing is done, the image frames are considered ready to be processed via two individual and separate background modelling techniques (Texture based BGModel and improved frame difference). Here each of these process is implemented on every input frame separately and in parallel, at the end the output of these parallel processes are combined together using several logical operations. In the following sections, each of these model improvements with their individual outputs will be discussed first, which will be followed by an explanation and examples of the output from their logical and manual combinations.

Three frame difference

Three frame difference method for detecting moving objects is developed on the basis of the conventional two frame differencing (Discussed in Chapter 4). It takes the three adjacent image frames for detecting the shape of the moving object(s) contours.

Considering three consecutive image frames $f_{k-1}(x, y)$, $f_k(x, y)$ and $f_{k+1}(x, y)$, the differences between the current frame and both the previous and next frames is found. These two subtractions produce two grey scale images $S_{k,k-1}(x, y)$ and $S_{k+1,k}(x, y)$. In both the cases the absolute difference is considered. These two subtraction processes can be expressed as follows in Equation (5.5) and (5.6):

$$S_{k,k-1}(x, y) = |I_k(x, y) - I_{k-1}(x, y)| \tag{5.5}$$

$$S_{k+1,k}(x, y) = |I_{k+1}(x, y) - I_k(x, y)| \tag{5.6}$$

As the image frames have been subtracted, so some information about the moving objects has been lost or added. Hence the union of the two subtracted grey images is used to obtain a resultant grey-scale image ($R(x, y)$ to recover some lost information to help the detection process. This combination is defined by Equation (5.7):

$$R(x, y) = S_{k,k-1}(x, y) + S_{k+1,k}(x, y) \tag{5.7}$$

As long as there is any moving object in both of the subtracted images, then it is considered that there is a possible object in the combined image $R(x, y)$. This situation can be mathematically expressed using an "AND" operation giving a function for the possible existence of an object within the combined frame $R(x, y)$ is shown as.

$$\text{Possible Object} = \begin{cases} 1 & S_{k,k-1} \cap S_{k+1,k} = 1 \\ 0 & S_{k,k-1} \cap S_{k+1,k} \neq 1 \end{cases}$$

At this point it is required to select a threshold value for getting a binary image from $R(x, y)$. The choice of threshold has a large effect on the performance of the final detection, so it is important to choose the correct one. There are two types of thresholding available - one applying the same threshold to the whole image and the other uses local threshold values at different locations in the image (also known as adaptive thresholding). For our approach, it was found that use of local threshold values for doing the image segmentation provided the best results. There are three primary algorithms for finding these local threshold values for adaptive thresholding, they are - I) finding the best value, II) calculating the biggest entropy and III) finding biggest square variance using inter-classes algorithm [101]. Moments automatic thresholding - based on finding the best value method, provided the best result for segmenting the resultant grey scale image ($(R(x, y))$. Moreover, the implementation of "moments automatic thresholding" approach is comparatively simple. Comparison of the combined frame differences image ($(R(x, y))$ with the adaptive threshold (T_A) can be expressed using the following Equation 5.10:

$$B(x, y) = \begin{cases} 1 & R(x, y) \geq T_A \\ 0 & R(x, y) < T_A \end{cases}$$

The resultant binary image ($B(x, y)$) contains noise and holes in the identified objects, so it is required to remove them. This can be done by implementing several morphological filtering operations on it, like 'opening' and 'closing' operations, also known as 'Erosion' and 'Dilation' respectively. So the overall process of this three frame differencing technique is shown in Figure 5.9.

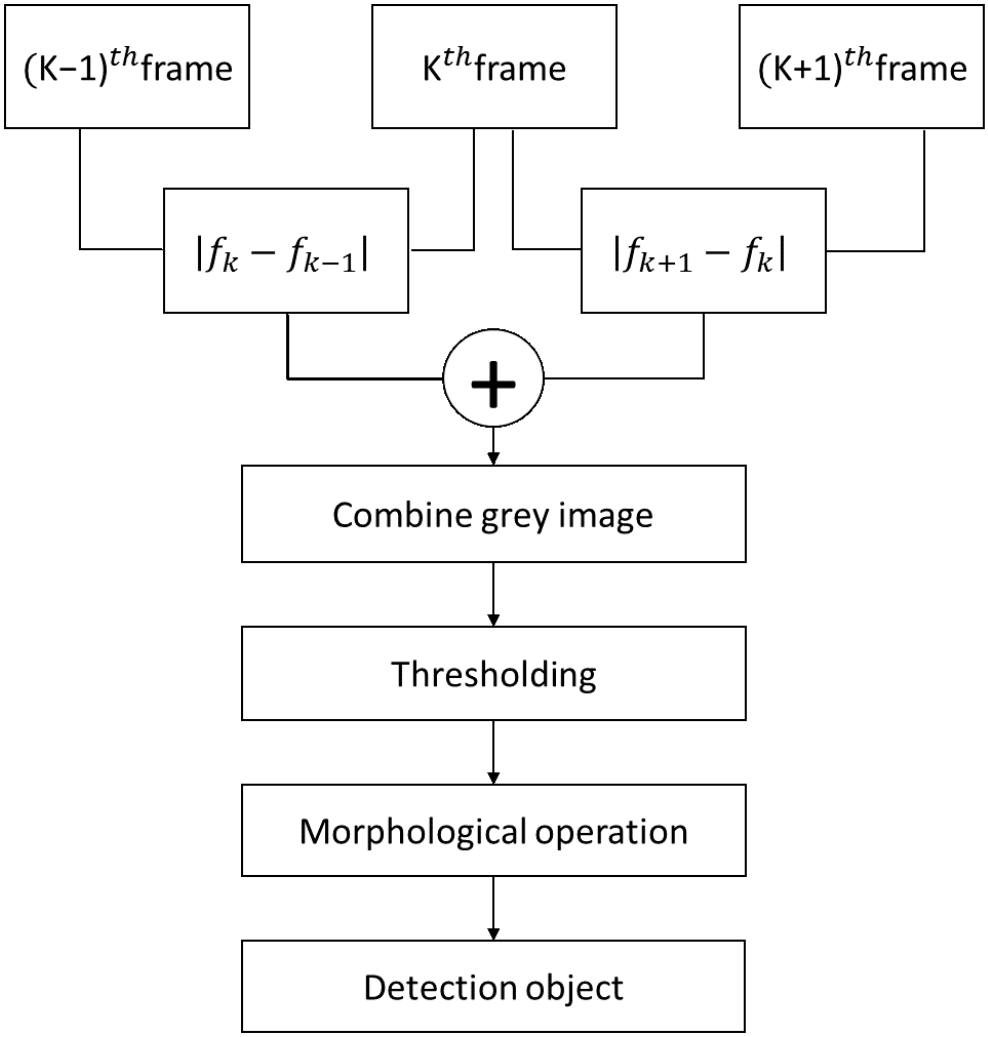


Figure 5.9: The principle for detecting new moving target algorithm using three frame difference.

Morphological operation 'opening' (Erosion) can eliminate very small objects, separate fine joints between objects and smooth the boundary of large objects within a binary image. On the other hand, 'closing' morphological operation (Dilation) is able to fill very small gaps and clearances between objects and smooth boundaries of small objects. Therefore, after these morphological operations on the binary image, small noise dots and regions are eliminated, including the filling of small gaps and clearances. In the proposed algorithm an 'opening' operation is performed to remove different small noises. Then a 'closing' operation is performed to fill different 'holes' inside the objects. So if $B(x, y)$ is the image and O is the structural element for the 'opening' and 'closing' operation, then the whole operation can be expressed as

Equation (5.8):

$$(B \circ O) \blacksquare O = \{[(B \ominus O) \oplus O] \oplus O\} \ominus O \tag{5.8}$$

In the expression shown in Equation (5.8), "$\oplus$" is Dilation operator, "$\ominus$" is Erosion operator "$\circ$" is Opening operator and "$\blacksquare$" is Closing operator. Here the structural element "O" is a 3×3 matrix and can be expressed as

$$O = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

After performing both 'opening' and 'closing' morphological operations it is found that some small white region which do not relate to object occur in the resultant binary image. For eliminating these errors, the areas of all the connected white pixels regions are computed and compared with a given threshold area. If a white region is larger than this threshold, then it is considered as a moving object.

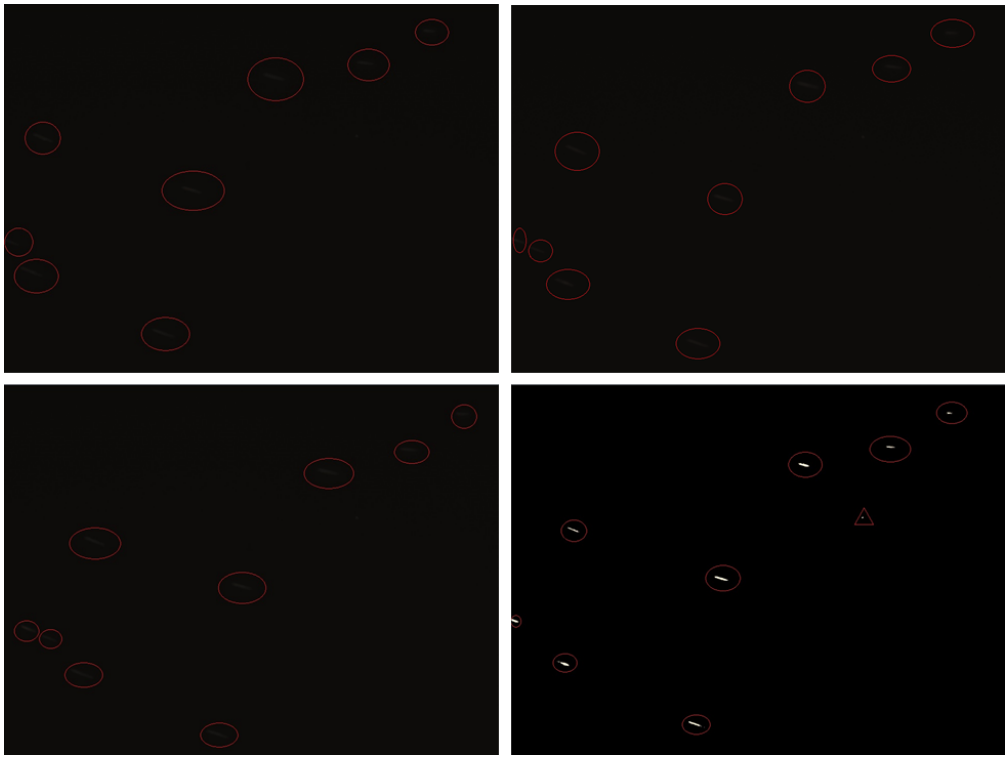


Figure 5.10: Three frame difference algorithm is applied on image sequence from 20 nM analyte concentration experiment (61st, 62nd and 63rd frame).

Implementation of this three frame difference technique provides information of non-moving (previously moving) or slowly moving object (s), which is relevant for location analysis. Many micro-particles get stuck for several frames and also in many cases their motion is too slow to be detected via GMM. So for our purpose the three frame difference was only used to identify the non-moving and slowly moving object(s), which was later reinserted using data from a previous frame within the detected foreground object(s) via GMM.

Improved GMM

For any background subtraction model to be considered as robust one of the main criteria is that- it must be able to adapt to environmental changes quickly. This can not be achieved using same the learning factor for updating Mean (μ_k) and variance ($\sigma_k{}^2$) used in the basic model GMM. To solve this problem, one approach is to use two different learning factors (β_1 and β_2) for updating the Mean and Variance [102].

With two learning factors Equations (4.14) and (4.15) can be adapted as shown in Equation (5.9) and (5.10):

$$\mu_{i,t+1} = (1 - \beta_1)\mu_{i,t} + \beta_1 X_{t+1} \tag{5.9}$$

$$\sigma_{i,t+1}{}^2 = (1 - \beta_2)\sigma_{i,t}{}^2 + \beta_2 (X_{t+1} - \mu_{i,t+1})(X_{t+1} - \mu_{i,t+1})^T \tag{5.10}$$

In which $\beta_1 > \beta_2$, here larger β_1 is required to improve the detection sensitivity and smaller β_2 is required to prevent the model from detecting an object as a part of background. However, use of two constant values for β_1 and β_2 over the whole process of background modelling was not a viable solution, as it was found that a constant learning rate would not be able to fully solve the problem of adapting with environment change. For this reason, it was required to find a mathematical expression for both these values (β_1 and β_2). Various publications (for example [93, 103, 104, 105]) showed several approaches for finding a relational equation(s) for finding adaptive values for β_1 and β_2, but most authors indicate that they have found them experimentally. β_1 and β_2 experimental values are quite difficult to prove mathematically. Some authors found the values of β_1 and β_2 using several statistical methods, they are also difficult to understand and implement, as these approaches are very much situation dependent, so these values are only applicable for certain environment(s).

In this situation, an approach using texture based segmentation for further improving the GMM background modelling technique was used. For texture based segmentation a newly developed technique called "Temporal Adaptive Median Binary Patterns (T-AMBP)" was used used for updating the weights of background models. For implementing T-AMBP with GMM, the GMM originally developed by Stauffer and Grimson [68] was not used, its improved version by Lee [94] was used. The implementation of T-AMBP with GMM is described in a later section (Under Section 5.1.2), but the description of T-AMBP is described first in following section.

Temporal- Adaptive Median Binary Pattern (T-AMBP)

Texture provides information about spatial arrangement of pixels grey levels or intensities in an image, which can be used as important characteristics to segment images into areas of interest, to classify and identify objects or regions [106]. Texture could give more information about an image compared to its histogram. For example, an image having a histogram with 50% white and 50% black pixels can have different textures as shown in Figure 5.11.

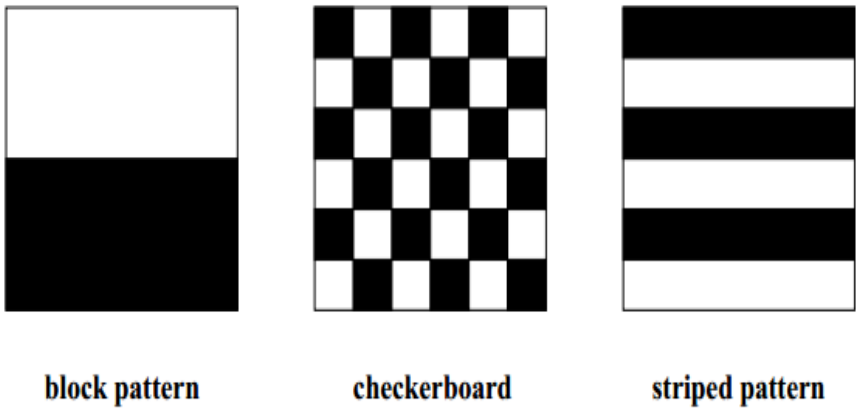


Figure 5.11: Three images with different texture characteristics and same distribution of histogram [100].

Many techniques have been developed for identifying objects based on recognizing different texture features, such as local binary pattern (LBP) [107] and median binary pattern (MBP) [108]. Texture can also be used for capturing background statistics. Recently Adaptive Median Binary Pattern (AMBP) [109] showed some excellent performance for many different texture segmentation applications which makes it suitable to use in background modelling. Moreover, one of the most important properties of the AMBP operator is its tolerance of sudden illumination changes in the background. In order to make AMBP more suitable to use with an input sequence and for other datasets, a modification to the AMBP operator was proposed, which can be called Temporal Adaptive Median Binary Pattern (T-AMBP).

Before the T-AMBP is discussed it was required to understand the initial approaches for developing T-AMBP, specially LBP and MBP. For this reason, the following sections are going to describe the basics of LBP and MBP.

Local Binary Pattern (LBP)

LBP is considered as one of the best texture descriptors. It describes textures or elementary structures of the local region and enhances the local properties by analysing the spatial distribution within the same area [107]. The basic LBP compares the neighbourhood pixel to the central pixels in a 3×3 block and transforms to an 8-bit binary number. Within this 8-bit binary number '1' is for intensity of the neighbourhood pixel over or equal to the central pixel value and 0 for otherwise. This resultant 8-bit binary pattern represents one of the 256 (0 to 255) distinct known patterns. The process of getting 8-bit binary pattern from a 3×3 block can be shown using Figure 5.12.

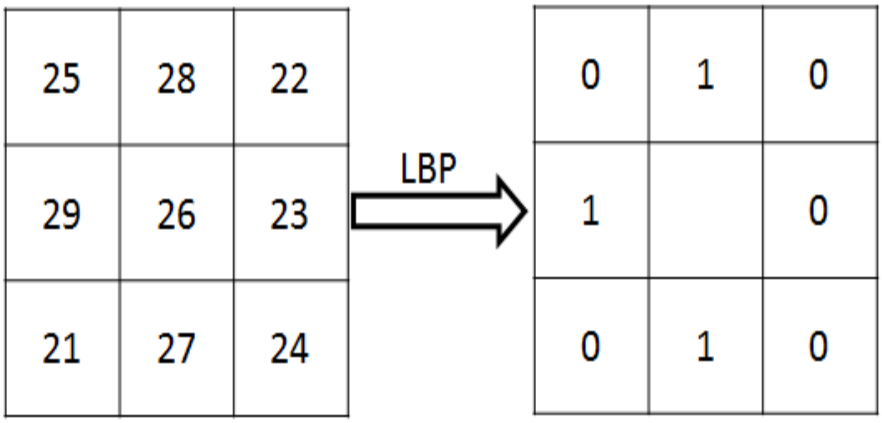


Figure 5.12: LBP technique performed a 3×3 neighbourhood pixel into an 8-bit binary. The central pixel 26 is used as threshold and ignored in the output for LBP.

In Figure 5.12, the right side of the figure is showing the binary pattern for the left side 3×3 block. Here all the neighbour pixels were compared with the central pixel value (26). So the resultant local binary pattern is 01000101, which is 69 (within 0 to 255). This means the texture value for the central pixel within this 3×3 block is 69.

The LBP operator can be expanded to use a larger neighbourhood and a different number of sampling points (**??**). However, the generic formulation of LBP operator can be presented without limitations to the neighbourhood size or sampling points [100]. So, in a (P, R) neighbourhood, where P is the number of sampling points and

R is the radius of the circle, the value of LBP with respect to a pixel (x_c, y_c) can be found using Equation (5.11):

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p; \qquad s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

In Equation (5.11), g_p denotes the gray level pixel value of all of the neighbourhood sampling points (indexed from 0 to $(P-1)$) used in LBP operator and g_c denotes the grey level of the pixel value in the centre.

LBP is very effective with a noiseless image, but it becomes difficult for LBP to segment the textures when there is any impulse noise within the image. To increase the robustness of this technique, Hafiane *et al.* replaced the threshold value using the median approach and developed Median Binary Pattern (MBP) operator [108].

Median Binary Pattern (MBP) and Adaptive Median Binary Pattern (AMBP)

Instead of comparing with the central pixel value, Median Binary Pattern (MBP) takes the median among the entire neighbourhood of the sampled point as the threshold value, and then produces a 9-bit binary pattern after comparing with this median value (Figure 5.13).

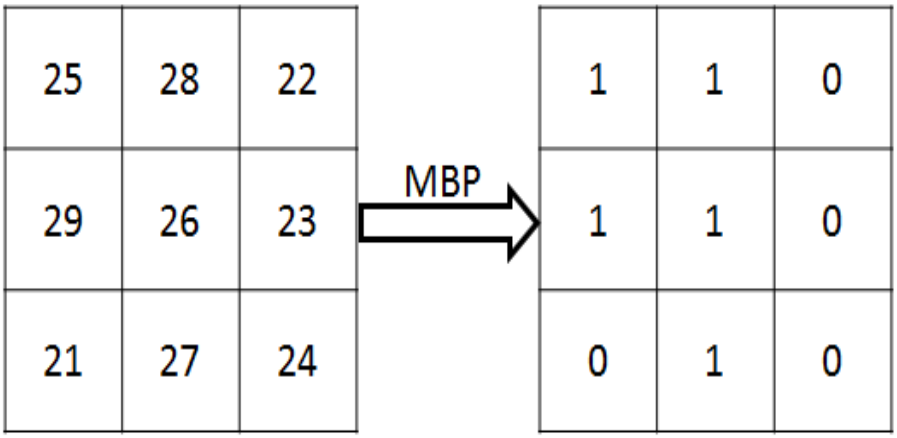


Figure 5.13: MBP technique performed a 3×3 neighbourhood pixel into an 9-bit binary. The median 25 is used as threshold and central pixel is also compared with median and output as one of the binary results.

Then using MBP operator, Equation (5.11) can be rewritten as Equation (5.12)

$$MBP_{P,R} = \sum_{p=0}^{P-1} s(x)2^p; \qquad s(x) = \begin{cases} 1 & \text{if } x \geq \text{Med} \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

MBP has shown to be a better non-parametric texture pattern operator compared with LBP [110], it also decreases the effect of impulse noise. But Hafiane *et al.* recently improved MBP technique further by combing LBP and MBP together [109]. This improvement is called as Adaptive Median Binary Pattern (AMBP).

AMBP algorithm is an adaptive median filter, modified from improving Median Binary Pattern. The basic idea of AMBP is to expand the search window for finding the median. The AMBP approach can be achieved within two steps. In a grey scale image with maximum analysis window k_{max}, assuming that S is the square region around a given pixel $I(i, j)$, whose size can be changed, so AMBP can be described as:

First step

For given $k = 1$ initially, then the pixel values within S block will be

$S = I(i - k : i + k,\ j - k : j + k) = I(i - 1 : i + 1,\ j - 1 : j + 1)$

Which is actually the 3×3 neighbourhood with its centre at $I(i, j)$. Now from the S region it is required to find the values of maximum(S_{max}), minimum(S_{min})

and median (S_{med}). In this situation it is required to check if these values follow the relation shown in Equation (5.13):

$$S_{min} < S_{med} < S_{max} \tag{5.13}$$

If the squared window satisfies the relation shown in Equation (5.13), it is required to check another relation as Equation (5.14):

$$S_{min} < I(x, y) < S_{max} \tag{5.14}$$

If the pixels within the squared region follows the relation shown in Equation (5.14) as well, then the threshold can be expressed as, $\tau = I(i, j)$ and the whole process act as LBP. If the squared region follows the relation in Equation (5.14), but not the relation in Equation 5.18 then the threshold $\tau = S_{med}$ and in this case the whole process act as MBP.

Second step

If within the 3×3 neighbourhood, $S_{min} < S_{med} < S_{max}$ cannot be satisfies, then the size of square region S will increase by 1 and becomes

$$S = I(i - (k + 1) : i + k + 1, j - (k + 1) : j + k + 1)$$

Then the new S_{max}, S_{min} and S_{med} are found. This process is repeated until the relation shown in Equation (5.13) is satisfied. Thus the value of k will increase until maximum analysis window k_{max} is reached, as long as a square window S can be found. After the window satisfying the Equation (5.13) is found then the process of finding the threshold follows as in step 1. After reaching $k = k_{max}$ Equation (5.13) is not satisfied, then the threshold will be $\tau = S_{med}$ for the smallest 3×3 block.

The process of AMBP is shown in Figure 5.14 which demonstrates changing the block size of the Adaptive Median Binary Pattern operator.

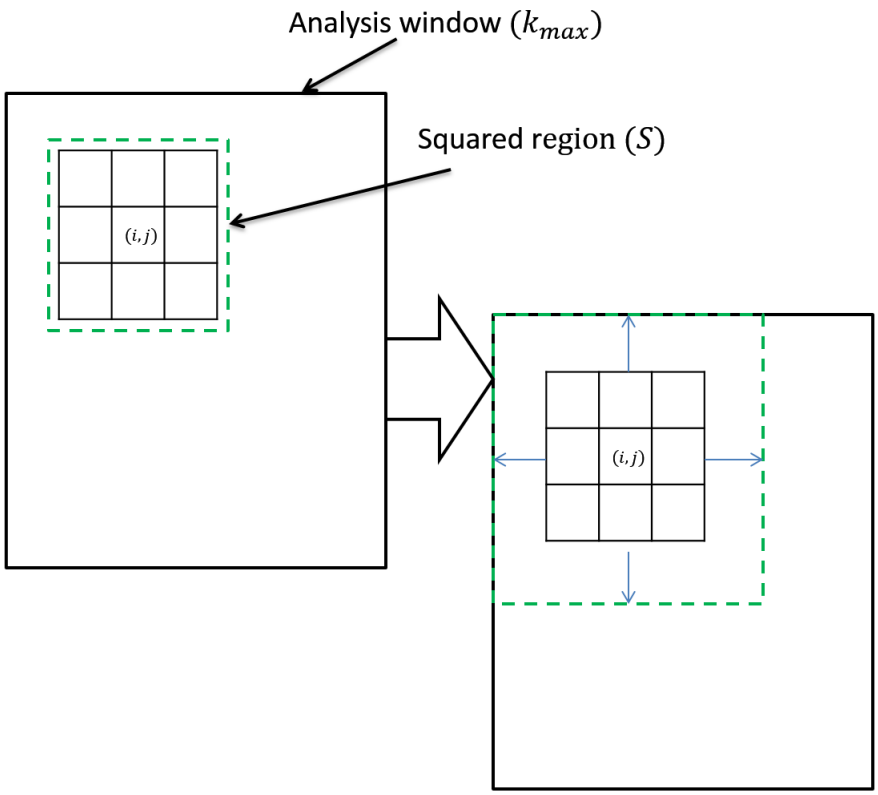


Figure 5.14: Demonstration of changing the block size for Adaptive Median Binary Pattern operator.

AMBP generates more binary information and shows better classification accuracy comparing with LBP and MBP. AMBP performs well for texture classification, but for using this technique with background modelling and subtraction it is required to have temporal knowledge about the pixel. So this improved AMBP includes knowledge about temporal information and is called Temporal-Adaptive Median Binary Pattern (T-AMBP).

Temporal-Adaptive Median Binary Pattern (T-AMBP)

This operator combines AMBP information from current frame and pixel information from the previous frames. The T-AMBP operator can be expressed using Equation (5.15):

$$T - AMBP = \left(\sum_{p=0}^{P-1} s(g_p - \tau) + G_t(p(i,j,t) - \overline{\mu}_{H,k}(t-1) \right) 2^P \qquad (5.15)$$

Here

$$G_t(x) = \begin{cases} 0 & \text{for } p(i,j,t) - \overline{\mu}_{H,k}(t-1) < 2.5 * \sigma^2_{H,k,t-1} \;\; k = 1, 2...k \\ 1 & \text{otherwise} \end{cases}$$

Where τ is threshold as defined the AMBP operator, $\overline{\mu}_{H,k}(t-1)$ and $\sigma^2_{H,k,t-1}$ are the mean value and standard deviation respectively for each group of histogram at time point $(t-1)$.

Using this T-AMBP operator, it is possible to increase the intensity of the brighter texture more brighter compare to the background.

T-AMBP with GMM

The objects to be detected from Lab On Chip experiments are microparticles and they differ from the background through their intensity level only, that is there are grey-level differences between the objects and background. The background is also subject to intensity gradients due to the nature of the lighting present. This non-uniformity in grey scale values within the frame of the LOC video sequences can be represented as texture features for the background. In the approach proposed T-AMBP operator is used for describing texture information for modelling the background. Using the texture information measured by T-AMBP instead of using pixel value directly for modelling background is more robust, adaptive and effective against noise.

In the following section the procedure for background modelling, updating and foreground detection based on T-AMBP and GMM is explained. T-AMBP method was used here as a feature vector for modelling background. For estimating the Mean(μ_k) and Variance($\sigma_k{}^2$) of the pixel model, the improved GMM method proposed by Lee [94] shown in Equations (5.16) and (5.17) was used.

$$\mu_{i,t+1} = (1-\rho)\mu_{i,t} + \rho X_{t+1} \tag{5.16}$$

$$\sigma_{i,t+1}{}^2 = (1-\rho)\sigma_{i,t}{}^2 + \rho(X_{t+1} - \mu_{i,t+1}).(X_{t+1} - \mu_{i,t+1})^T \tag{5.17}$$

Where $\rho = \frac{1-\alpha}{c_k} + \alpha$

In the first step, the T-AMBP histogram was used to model background by computing the binary value for each pixel within a square region of window size $N \times N$, where the initial value of N is equal to 3, but the size is adjusted according to the situation (Figure 5.14). K histogram groups $\{\overrightarrow{m}_0, \overrightarrow{m}_1, \overrightarrow{m}_2 ..... \overrightarrow{m}_{K-1}\}$ are used for representing background models, where K is a user defined number (different from the number of Gaussians in GMM). According to the background types, K is usually set between 2 and 5 [111]. Bigger K will provide more detailed information, however the process will be slower. Each of these histograms has a weight between 0 and 1, the summation of all these weights is 1. So, if the weight of k^{th} histogram can be represented as ω_k, then

$$\sum_{k=0}^{K-1} \omega_k = 1 \tag{5.18}$$

Now if the histograms are sorted in a descending order according to their weight (ω_k) and considering the first B histogram models as the background histograms, then we get:

$$\omega_1 + \omega_2 + ... + \omega_1 > T_B \tag{5.19}$$

Here, T_B is a user defined threshold and its value can vary from 0 to 1 depending on the background scene and closely related with K.

Now if $\overrightarrow{h}$ is the current T-AMBP histogram for a given pixel in the new input frame for the video sequence, then it is first compared with the existing K model histograms using a proximity measure. This proximity measure histogram intersection was used according to Heikkila and Pietikainen [76], as it was easier to implement and has very low complexity. Also it has the advantage that it neglects features which occur only once in one of the histograms. So according to Heikkila and Pietikainen [76] if $\overrightarrow{h_e}$ and $\overrightarrow{h_c}$ are the existing and current histograms respectively, N is the number of histogram bins then the histogram intersection between

$\overrightarrow{h_e}$ and $\overrightarrow{h_c}$ histogram is given by Equation (5.20):

$$\cap \left(\sum_{k=0}^{N-1} min(h_{ek}, h_{ck}) \right) \tag{5.20}$$

Here k is the histogram bin index.

A similarity threshold T_s is chosen in this stage and all the histogram intersection results are compared with T_s. The value of T_s is user defined and has been found that it's value between 0.6 and 0.7 generate good results [111]. If the similarity is lower than T_s for all histogram models, then the pixel is considering as foreground and the model histogram with the lowest weight is replaced with current histogram $\overrightarrow{h}$. Also a low initial weight (user defined) is assigned to it and no further processing is done.

On the other hand, if the similarity is higher than the threshold T_s for at least one histogram model, then the pixel is labelled as background. In this case the model histogram is updated with the highest proximity value. If the best matching model histogram is denoted by $\overrightarrow{m}_k$, then it is adapted with the new data by updating its bins according to Equation (5.21):

$$\overrightarrow{m}_k(t) = (1 - \alpha_b)\overrightarrow{m}_k(t-1) + \alpha_b \overrightarrow{h} \tag{5.21}$$

where α_b is a user-settable learning rate. At the same time, the weights of all model histograms are also updated according to the rules of basic GMM, which can be expressed as Equation (5.22):

$$\omega_k(t) = (1 - \alpha_\omega)\omega_k(t-1) + \alpha_\omega M_k(t) \tag{5.22}$$

where M_k is 1 for the best matching histogram and 0 for the others and α_w is another user defined learning rate. For both the learning rates (α_b and α_w) the value must be within 0 and 1. The bigger the learning rate, the faster the adaption.

Combining Three frame differencing with output from GMM

Before the process of combining these widely used techniques is discussed, the reason for combining them together must first be discussed. First consider the characteristics of the captured image sequences:

1. The objects to be detected for this project are chemically activated microparticles, whose intensity is related to the concentration of the analyte.

2. The micro-particle are circular in shape, but due to their velocity and their relatively long time exposure looks oval in shape on the image. These ovals have a brightest point at their centre (almost) and with the intensity being lowest in both sides of their tail.

3. Another characteristic of these micro-particles is their motions are not uniform, specially some particles stick or more very slowly.

4. Each of the image sequences from the experiments contains around 100 frames.

5. Depending upon the experience of the experimenter the background condition of the sequence can change dramatically (if the flow rate is not controlled properly).

Now considering the above situations, it was found that the use of only Gaussian Mixture Model (GMM) resulted in losing the non-moving or slow moving objects. It also takes quite a large number of frames for GMM to adapt to any change in the background, it was not possible with the input sequence(s) available to allow such a number of frames for adapting to the change. So it was decided that the combination of improved GMM with the three frame difference approach would be used. Here, three-frame differencing is used to identify the slow moving or non-moving objects. Then the output from the improved GMM is compared with the image output from three frame difference ($TFD(x, y)$) as following expression for identifying the foreground object(s):

$$R_{fore}(x,y) = \begin{cases} 1 & IGMM(x,y) \cap TFD(x,y) = 1 \\ 1 & IGMM(x,y) \cap TFD(x,y) \neq 1 \end{cases}$$

The combination of these method resulted in detected objects, both moving and non-moving. But still the output contains some noises, like salt and paper noise and random noise. This noise was then removed in the post processing stage.

5.1.3 Post Processing: Median

The combined binary output from the background model results contain some unavoidable isolated points which are not actually objects. It is necessary to filter these points, as they may interfere with the later object tracking process. For removing these isolated points the method of median filtering [112] was used. The median filter is a nonlinear smoothing technology. The basic idea of this technique is to replace the grey value of the current pixel with the median grey value of the neighbouring pixels. Considering a neighbourhood of S, the median filtering can be expressed using Equation (5.23):

$$g(x,y) = medf(i,j); \quad (i.j) \in S \tag{5.23}$$

Here $g(x,y)$ is the new pixel value, $f(i,j)$ is the pixel value in (i,j) position.

For implementing this filtering, it is required to consider a moving window with $N \times N$ points for scanning the image. Each pixel is then replaced by a pixel value equal to the median value of the pixels within the window.

For doing this the pixels in the window are sorted according to their grey value (highest to lowest). If $N \times N$ is an odd number, then middle (central) value is used; otherwise the average of the two middle values is used to replace the current value of the pixel.

It is always required to use an optimum size for the moving window. The bigger the window is the greater the loss of information. Moreover larger windows require more computational power. So considering both the situations, a 3×3 sized moving

window was used as a median filter. After this the result shows that the isolated points were reduced with little smoothing effect on the output image, which can be seen in Figure 5.15

Figure 5.15: Application of 3x3 nonlinear median filter on the binary output image.

5.2 Results

In this section results after using proposed approach are presented. All the required programs were written using the C++ programming language (it was tested using python as well) with the Opencv 2.4.10 image processing library (www.opencv.org). Opencv supported this work with many basic functions like erosion, dilation, median *etc.* As input sequence, most of the analysis was done using magnetophoresis experimental results from DNA hybridization carried out by Dr Martin Vojtisek. Also for checking the usability in different scenarios, three benchmark data sets were used. The performance of the approach mainly depends on number of Gaussian (K) and which was chosen $K = 3$.

Here the improvements of the approach can be divided into four parts, 1) robustness to low illumination object detection, 2) robustness to non-moving (which was previously moving) object, 3) Robustness to noise and 4) robustness to shadow for general benchmark inputs. So the results are divided into four sections. For comparison, the results will be compared with four widely known methods - Gaussian Mixture Model (GMM) [92], Vibe [73], KDE [87] and Adaptive Background subtrac-

tion learning [113]. For comparing the approach with those methods (except Vibe), codes from the widely used bgslibrary coded by Andrews Sobral (www.github.com/andrewssobral) was used. For the Vibe method the code was provided by O. Barnich and M. Van Droogenbroeck (www.telecom.ulg.ac.be/research/vibe/doc/).

5.2.1 Improvement for detecting low illumination object

This improvement was one of the key requirements, as in the magnetophoresis experiment it is required to detect objects from very low concentration analytes, meaning the objects have very low intensity. For demonstrating the results for improvement in detecting very low illumination objects, we have chosen input sequences from 0 nM and 20 nM concentration of analyte. The result can be seen in Figure 5.16 and Figure 5.17.

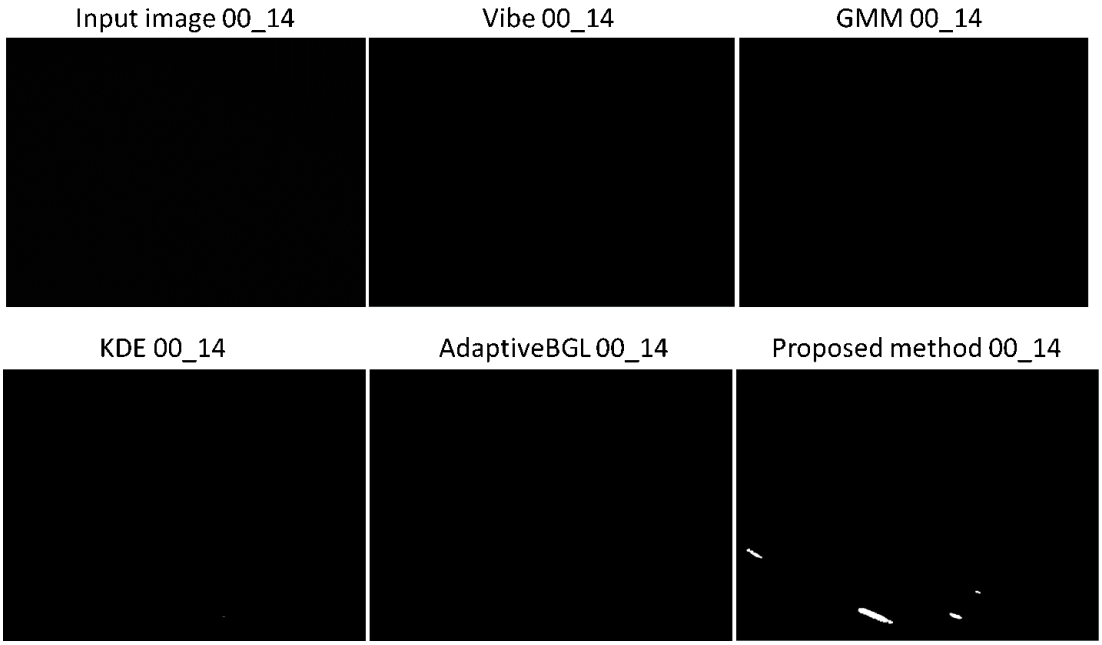


Figure 5.16: In this figure is for result image with input sequence achieved from 00 nM analyte concentration (also known as blank). In this figure the first image is the input from 00 nM image sequence and 14th frame, the others are output from detected objects using Vibe, GMM, KDE, Adaptive Background Learning and proposed method respectively.

Micro-particles are present within these image sequences, regardless of the analyte concentration. But they are not detectable using the human eye in many

cases. From Figure 5.16 for which the analyte concentration is zero, it can be seen that while all four well established method failed to detect any object, the proposed method detected all the three objects.

Figure 5.17 shows results with concentration of 20 nM analyte. In this image sequence, though the microparticles are not clearly visible, with high attention they could be seen in the input data.

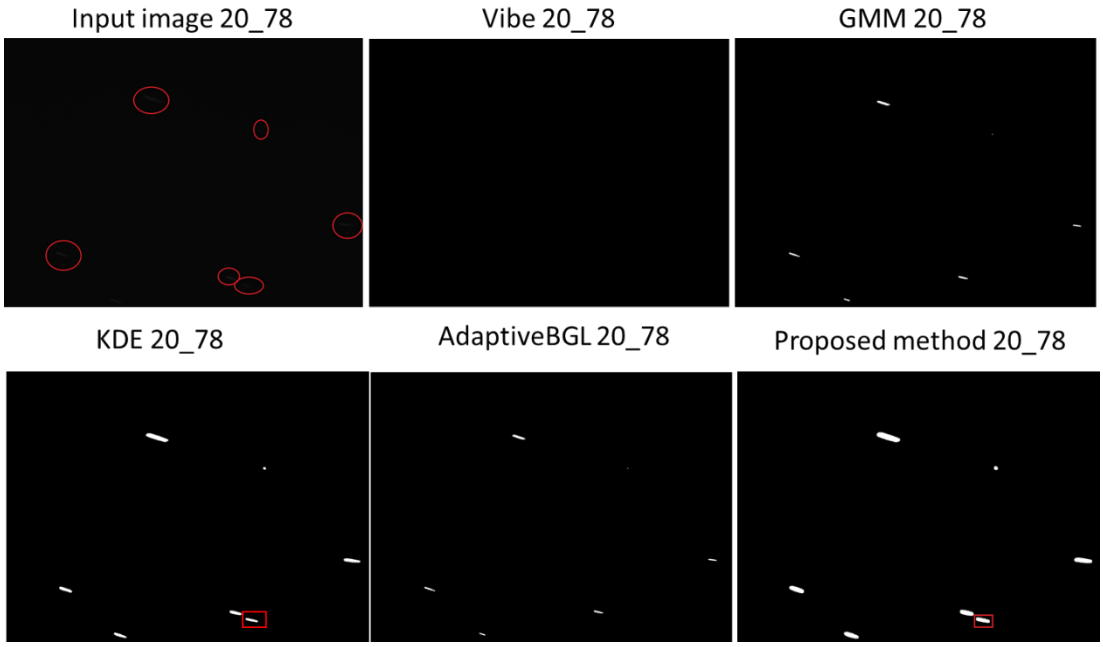


Figure 5.17: Object detection result for input sequence from magnetophoresis experiment with analyte concentration of 20 nM. The first image is the input from 20 nM image sequence and 78th frame, the others are output from detected objects using Vibe, GMM, KDE, Adaptive Background Learning and the proposed method respectively.

In Figure 5.17, 'Red' circles in the first image are showing the actual positions of the objects within the frame. Similar to the result in Figure 5.16, vibe was not able to detect any objects, GMM detected 5 objects out of 6, KDE detected all the objects, and the adaptive background learning method detected 5 objects. Though GMM detected 5 objects in this case, the object shape was not captured correctly. This is also true for adaptive Background learning method. So for this case, out of all the four established methods, KDE performed the best. Therefore, only objects having high contrast with background could be detected by most of the methods in the test system. But when considering the objects shape,it can be

clearly seen that the proposed method provided better representation of the micro-particles. Even for the objects which were very close to each other (marked by red rectangle for both KDE and proposed method) the proposed method detected both the objects separately with a good represented shape and with much clearer view. One very important thing to be mentioned here is that GMM and Adaptive Background learning required a minimum of 20 frames for detecting these objects, KDE required a minimum 10 frames, whereas our method only required the first 3 frames to start the detection process. Also the objects detected by KDE had discontinuities at the middle of the objects.

So the proposed method successfully detects particles from input sequence from analyte concentration of 0nM, where others failed. Also it performed well over other methods for the input sequence from analyte concentration of 20nM. The problem of detecting object with extremely low contrast was solved. Thus this system can be implemented into many other fields where object signal is extremely faint or low contrast comparing with background.

5.2.2 Detecting Non-moving (previously moving) Objects

One of the very common problems that the experimenter in Chemistry Department faced was many micro-particles which were moving previously suddenly stopped moving either for a few frames or for the rest of the experiments. It was very important to detect these objects (stopped moving), as it was found that non-moving objects produce much higher intensity compared to a moving object potentially leading to erroneous results. So such objects are required not to be considered for the calculations related to finding analyte concentration. In this case the requirement was to detect such non-moving objects and keep a record of them during the whole tracking process. In the case of many other methods, where they detect foreground object(s) via subtracting the current pixel value from the previous one, they failed for detecting suddenly stopped objects, because when a moving object suddenly stopped in one frame, the pixel value of the object position in all the frames while

the object is non-moving are same. Therefore, there will be no difference in pixel value between the two consecutive frames in the object's stuck position. However, GMM and other methods derived using statistical approach, subtract the current pixel value with the previous background model's mean instead of individual pixel. In this way there will be difference between the suddenly stopped object's pixel value and background model. Thus it is recognised as a foreground object. Therefore, this suddenly stopped object will take a while to be part of the background model. But the main difference between these statistical approaches is how long a model takes to convert a non-moving object to background. Comparative results for keeping a record of non-moving objects are shown in Figure 5.18.

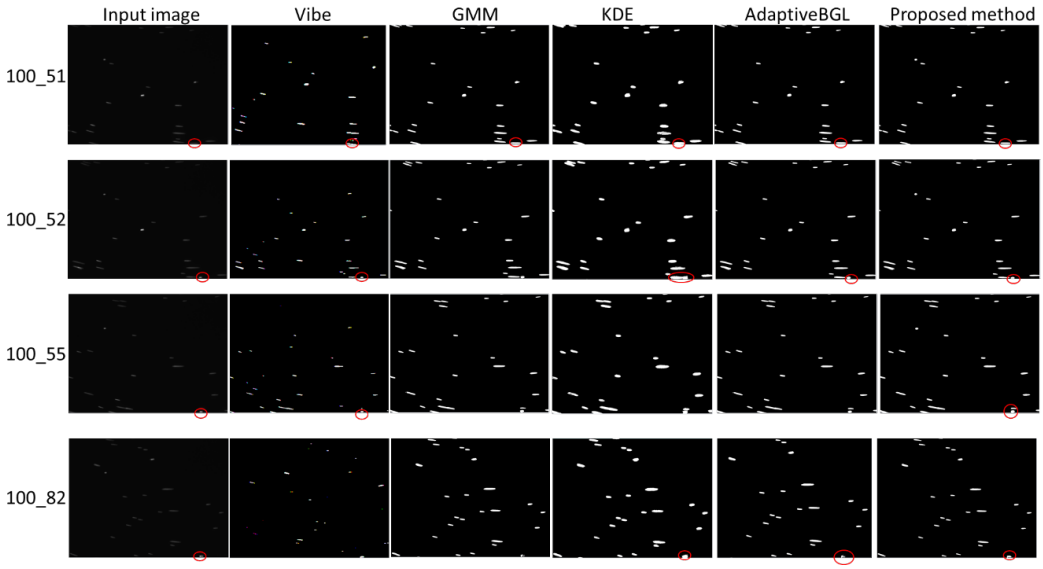


Figure 5.18: Most left side images in every row is showing the input frame. Other five rows of images are the detected object by from Vibe, GMM, KDE, Adaptive Background Learning and proposed method respectively. 'Red' circle is showing the position of the interested object location. The sequence used here from experiment with analyte concentration of 100 nM.

From Figure 5.18 it can be seen that all of the test methods including the proposed method could deal with the situation of suddenly stopped objects. In comparison, GMM, Adaptive Background learning and Vibe turn the stopped object to background faster than KDE and the proposed method, which can be seen in Figure 5.18 frame 82. From this test it was found that it took about 25 frames for GMM and Vibe, 27 frames for Adaptive background learning method and about 30

frames for KDE, and about 50 frames for the proposed method before the object was classed as part of the background. The reason for the proposed method keeping a record of a non-moving objects longer than other is- because a histogram based model to compare the binary value of the pixel with background model was used. The binary value of the pixel is derived using T-AMBP, which includes temporal and spatial information of the pixel and enhanced the signal of the foreground object. As the proposed method always uses enhanced signal to compare, so it takes longer (about 50 frames) to detect a non-moving (previously) object as background.

5.2.3 Robustness to Noise and Artefact

In addition to the noise, the image from the magnetophoresis experiments contain artefacts, which include air bubble(s) and non-moving particles (within capturing period). Also it was found several existing background models require lots of frames to remove the effect of any (background) object which was available during the training period. Figure 5.19 shows the results of a comparison of how various algorithm handle such an artefact.

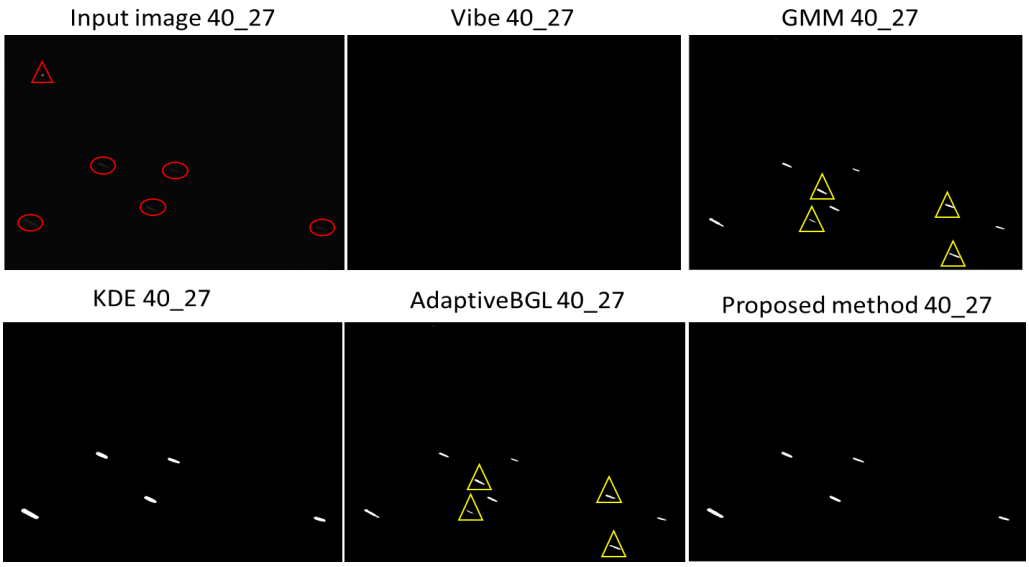


Figure 5.19: The top left image is showing the input frame from experiment by 40 nM concentration of analyte. The Red triangle shows an object which never moved in the input frame. Other images are showing the detected objects form Vibe, GMM, KDE, Adaptive background Learning and proposed method respectively. Yellow triangles are artefacts, but detected as object.

In Figure 5.19 it can be seen that the objects are falsely detected by GMM and the adaptive background learning method as indicated by the 'Yellow' triangles in the result images. The object marked by 'Red' triangle is an object which exists in the input frame from the beginning of the process and it never moved. All these methods including our proposed method detected it as background. Also the size of the detected moving objects by KDE were larger than the real size of the objects (checked by counting number of pixel manually within the input object and detected object) by 3 to 5%, on the other hand, our method provided result by detecting objects about 0.5 % to 1% smaller than the real size of it.

5.2.4 Robustness to general benchmark inputs

Machine vision developments are often target oriented, so that most cases when an approach is suitable for specific type of task, it may will not provide similar performance for other type of condition. For example, considering Vibe method, it usually provides good performance for human and vehicle detection tasks. But for input images from microfluidic experiment results, it provided the worst results amongst the methods evaluated.

Though the main target was to develop an approach which could detect and measure micro-particles over a wide range of analyte concentrations, it was always desirable to develop an object detection technique which can perform in other situations. For this reason, the proposed approach was checked for detecting humans and cars from the PETS'09 data set images. The comparison results can be seen in Figure 5.20 and Figure 5.21.

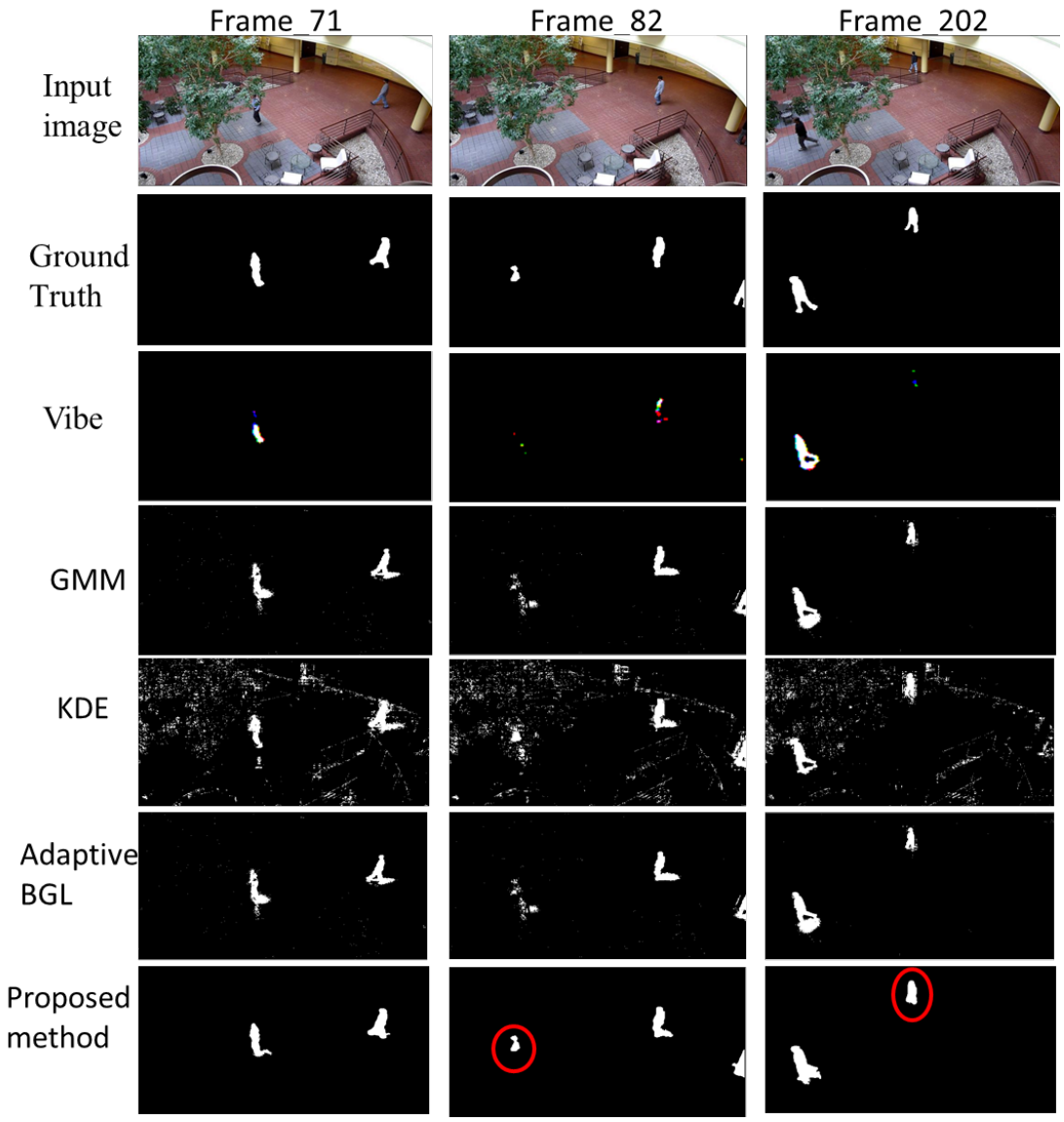


Figure 5.20: The images in the first row is showing the input frames from PETS'09 dataset for detecting human. Here the second row is showing the images for ground truth. Other rows of images are showing the detected human form Vibe, GMM, KDE, Adaptive background Learning and proposed method respectively.

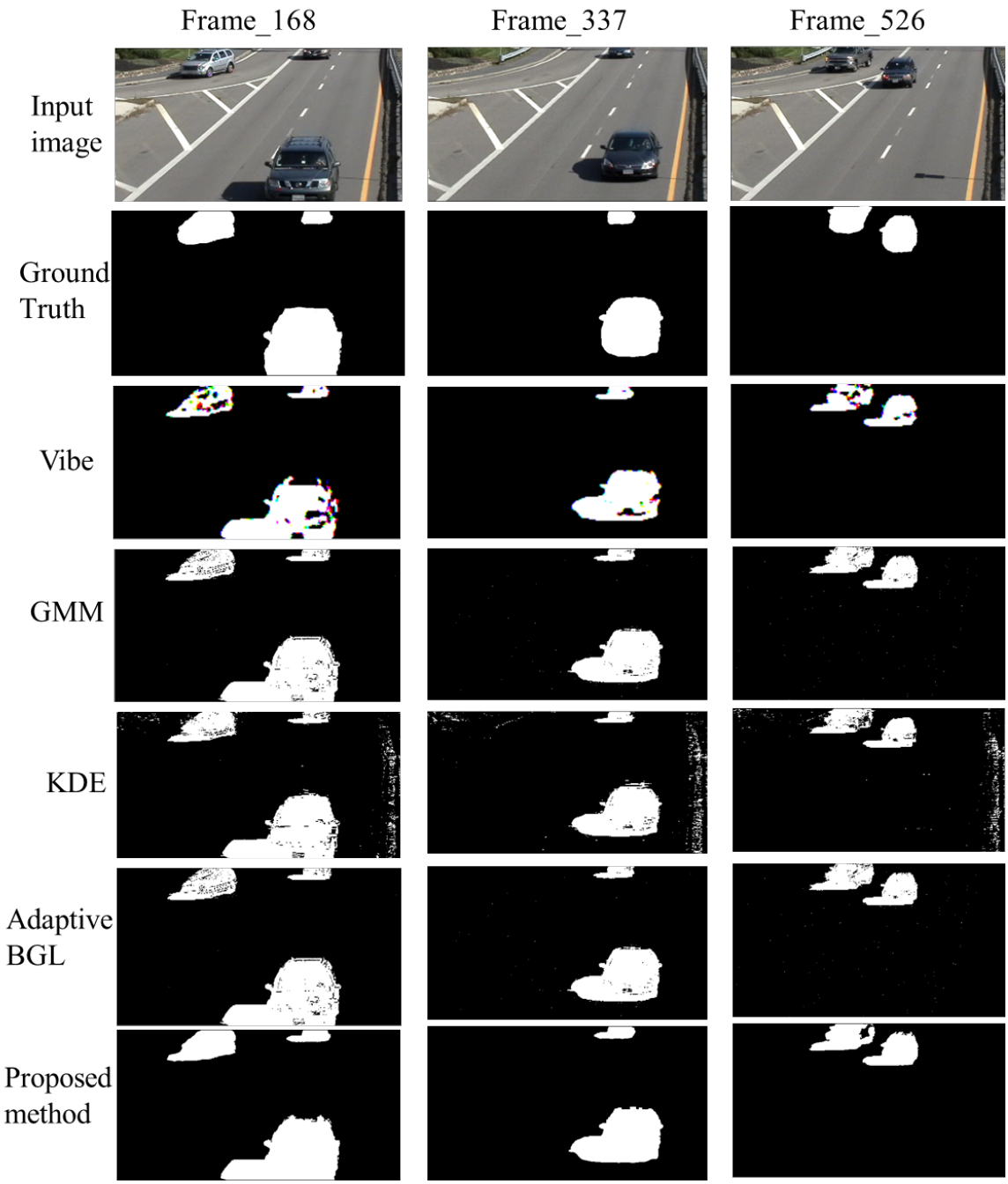


Figure 5.21: The images in the first row is showing the input frames from PETS'09 dataset for detecting car. Here the second row is showing the images for ground truth. Other rows of images are showing the detected car form Vibe, GMM, KDE, Adaptive background Learning and proposed method respectively..

By analysing the images in Figure 5.20 and Figure 5.21, improvement for the proposed method over other methods in terms of the detected shape of the objects and noise can be seen. For the shape of the object, the proposed method detected both the human and car as its full shape without having any discontinuity within the objects. Moreover, in Figure 5.20 it can be seen that when other methods had difficulties of detecting an object (human) far from the camera, the proposed approach detected the object as a whole body. On the other hand, in the same figure, it also can be seen that an object with is barely visible comparing with background, can be detected (red circle) as an object without noise.

For the detected objects (cars) in Figure 5.21, the objects are clearly visible without any discontinuity. But comparing Figure 5.20 and Figure 5.21, it can be seen that the shadow for the detected objects in Figure 5.21 is much obvious then that of Figure 5.20. This is because that the difference between object colour components and shadow components is different in two images (Figure 5.20 and Figure 5.21).

5.3 Evaluation Metric

In this section, the proposed approach is compared with several well-established algorithms like GMM [92] , ViBE [73] , KDE [87] and Codebook [114]. For comparing the performance, image sequences from benchmark datasets "Li" and "wallflower" were used. Five videos were taken with different settings. These videos have several challenging features like moving leaves and waves, moving objects becoming stationary for a long duration *etc.* For the evaluation metric, the widely used and well-defined formula called $F - Measure$ [115] was used, which considers precision and recall factors.

In the following TP and FP denote correctly and incorrectly classified foreground pixels respectively, FN denotes foreground pixels that are incorrectly classified as background pixels. The Precision and Recall are defined as:

$$Precision = \frac{TP}{TP+FP}, \qquad Recall = \frac{TP}{TP+FN}$$

For the input videos from the standard benchmark data set, videos having 300 to 1500 frames in length were used, though the widths and heights of the input frames were not same. The *F-measure* indicates the accuracy [116] of the image segmentation, the higher the *F-measure* is the better the approach is. The result metric is shown in Table 5.1. It was not possible to calculate the *F-measure* for LOC experimental data, as no ground truth was available.

Input Data	Vibe	GMM	Codebook	PKDE(Siltp)	Proposed Method
Li Dataset (Lobby)	72.63	68.42	63.38	78.80	86.87
Li Dataset (Camouflage)	47.79	67.07	63.38	71.92	85.32
Wallflower dataset (Waving Trees)	54.33	60.54	47.27	64.28	72.58
Wallflower dataset (Time of day)	39.67	40.01	51.32	73.67	83.18

Table 5.1: F-measure comparison between various existing algorithms on Li and wallflower dataset. Here the results for Li and wallflower dataset using Vibe, GMM, Codebook, PKDE were reported by [116], [117] and [113] respectively. The other results were calculated.

5.4 Summary

The proposed approach to background modelling and subtraction overcomes the problem of detecting slow moving objects and objects which were moving initially and then stopped, it is also able to handle objects which are barely visible and cope with sudden illumination changes. In most cases when GMM, Vibe, KDE, Adaptive background learning *etc* failed to detect micro-particles from results with low concentration (specially 0 nM and 20 nM), proposed approach performed very well.

The results in Figure 5.16, Figure 5.17, Figure 5.18, Figure 5.19, Figure 5.20 and Table 5.1 showed the performance of the proposed method over others for

more general image processing benchmarks. Though the images from microfluidic experiments do not have shadows, it was possible to reduce the effect of shadow to some extend for the benchmark input sequences. Moreover, the proposed approach was able to better detect objects which are far from the camera.

The work in this chapter enables the automated data analysis process to locate the micro-particles in the experiments throughout the required analyte concentration range. It also enables stuck particles to be identified. This is potentially useful beyond simply identifying the correct particles for analysis; if particle behaviour becomes abnormal (*e.g.* a much greater level of "stucking" than usual) it may be possible to detect faults with the system. The identified "good" particles are passed to a tracking algorithm which will follow their path through the video sequence to further facilitate selection of the best data set from which to extract measurements. The following chapter reviews relevant published material on object tracking on which our object tracking will be developed.

Chapter 6

OBJECT TRACKING REVIEW

Object tracking or visual tracking is a repeated estimation process of the current and future state (position) of an object by comparing with the previous state from any given scene. Object tracking is already being used for many different applications like surveillance, medical imaging *etc*. Depending on the application, tracking systems can be manual, autonomous (fully) and semi/partially autonomous. For this project the objects, which needed to be detected and tracked automatically were micro particles from multi-laminar flow microfluidic LOC experiments. Automatic object tracking is one of the main tasks for this thesis.

Automatic object tracking is considered as very important and critical task for many computer vision systems, as implementation of automatic object detection and tracking systems could monitor complex activities and prevent many potential problems. So it has been applied in many surveillance systems for automatically understanding the tracked objects' activities, movement and behaviour, such as human identification [25], traffic monitoring [118] and environment surveillance [119]. Advanced object tracking analysis systems require robust technique(s) which can identify and detect the objects taking into account complex environmental changes, differentiate them from multiple other objects and define the tracks for individual object.

Automatic object tracking can be divided in three steps: 1) representing the object with distinguishable feature(s), 2) detecting the object(s) and 3) tracking

the detected object(s) through a sequence of image frames (also includes analysing the objects tracks to understand its movement). Object features includes colour, edge/shape, optical flow, texture *etc*. Object representation may be based on object shape (geometric shapes, silhouette and contour, articulated shape, object skeleton) or object appearances (such as probability object density, trained templates, active appearance models *etc*). Background subtraction (described in Chapter 4) is the main approach used for detecting the objects of interest, as background subtraction differentiates the wanted object. There is a very strong relationship between object representation, identification and tracking. As tracking is performed on differentiable features and data for the objects provided by first two tasks, robust and efficient tracking is highly depending on the performance of object representation and detection.

Lots of approaches were already being proposed for robust and efficient tracking. Each of them has their own advantages and limitations and not all the approaches are suitable for every tracking application. Tracking approaches can be divided with respect to two points of considerations. Firstly considering the input sources and secondly depending on the use of search windows and object matching strategies. In terms of input sources, object tracking can be divided into on-line and off-line tracking. On-line tracking techniques are also known as real-time tracking. In some cases, real time tracking and real time object detection are considered as similar approaches, but in real time object detection the object(s) is continuously detected without keeping any record of its trace, whereas for real-time object tracking the objects are detected continuously and their traces are also kept on record. In terms of the search window and object matching strategies used, tracking approaches can be classified into two main types deterministic and probabilistic tracking. The following part of this chapter will mainly review widely used object tracking techniques.

6.1 Object Detection vs Object Tracking

Object detection and recognition is the step prior to tracking, but sometimes object detection and recognition works closely and in such a similar way to tracking, thus tracking and object detection seem to be the same process. However, tracking is a totally different process compared with detection and recognition. For any object detection, the location and configuration of some classified object is required to be found, such as face detection, vehicle detection *etc*. For detection, the relationships of the detected objects within different video frames are not defined, also the identity of the detected object(s) is not maintained. Moreover, object recognition is the task of identifying or classifying objects according to a given set of known object(s) (like car, human, micro-particle *etc*). For object detection and recognition it is not necessary to find the location from the corresponding object(s) in a subsequent frame with the same identity. For example, for identifying vehicle registration plate location and recognition the registration number recognizing process takes place, but no tracking is required.

6.2 Object tracking challenges for LOC experiments

Object (multiple) tracking is a visual tracking process where the tracker dynamically configures the tracking process for one or more objects for each frame of a video sequence. Through this dynamic configuration, the tracker identifies each object separately and recognises the same object with the same ID over the whole sequence. For making the tracking process successful, ideally three main assumptions are made (1) object(s) change their position(s) in a regular pattern and speed, (2) the shape and features of the object do not change dramatically within consecutive frames and (3) the objects move slowly [120] so that there is minimum overlap between the areas occupied by object in the current frame and previous frame.

Unfortunately, these assumptions do not always occur in practice. In many cases

object(s) stop moving and changes in formation and shape occur. Also, depending on the camera quality and set-up (including user's experience), in many cases the capture of object movement is not uniform and the areas occupied by an object in two consecutive frames do not have any overlap or has a very small overlap. Both these situations were found to be very common during the analysis of LOC microfluidic experimental data (see Figure 6.1).

Figure 6.1: Micro-particle's position changes over the frames without overlapping with previous position. Images from experiment with analyte concentration of 100 nM and frame numbers are 7th and 8th respectively.

For robust tracking in some cases more than one camera is used, but before those images are processed these multiple video inputs need to be combined to fuse the

information from multiple sources [68, 121, 122, 123]. But for the Lab On a Chip experiment we were restricted to video from a single source only.

For the best object tracking result, it is required that the objects' properties are different from the background. But for analysing LOC experimental data it was found that for the experiments with low concentration reagent, the object does not differ much from the background. Moreover, due to several experimental conditions it is found that previously detected objects can become invisible to a detector or tracker. To solve such problems, it seems that the application of a Kalman filter (Discussed in section 6.4.2) approach could provide the future position of any lost object for continuous tracking. But there are several limitations for the implementation of those approaches, in particular if many particles are close to the predicted position to be, it is difficult for Kalman filter and Particle filter to find the correct object. It was found very common in LOC images that many objects stay close to each and also they follow similar path(s).

6.3 Tracking Steps

Lots of approaches and algorithms have been published for tracking multiple objects and almost all of them consist of four major phases: 1) data acquisition, 2) processing (object detection and analysis), 3) motion analysis and 4) tracking.

In the first phase, data acquisition, the camera collects the sequence of data (video) and passes it to the second phase for processing and modelling. The processing phase includes data pre-processing, objects detection and recognition, where Background (BG) modelling or Foreground (FG) detection, noise removal, some morphological operations are done. In general, object detection is the way to differentiate the target object(s) from the whole input scenario. Object detection is implemented continuously on every frame for finding the new position of old objects and also for detecting new object(s). Details of these steps were discussed in Section 4.2 in Chapter 4.

Output(s) from the processing step are then used for analysing motion. In this

phase some tasks from the processing phase, with some new strategies, are required for detecting and understanding the movement and other activities (such as shape deformation, angular movement *etc*). After analysing the behaviour and activities of the detected object, the tracker tracks them using different strategies. Multiple object tracking keeps repeating the tasks of object modelling, environment modelling and searching. Appearance changes, variation in illumination, differentiating multiple objects from each other, finding new object *etc* are considered in object modelling. This object modelling is performed using some unique features of the objects like shape, colour, motion, background, orientation *etc*. Sometimes a combination of two or more distinguishable features, possibly with some statistical approach, is considered for object modelling.

Next, in the tracking phase, the input environment is modelled first, that is the point and direction of object entry and exit from the frame *etc*. This environment modelling is necessary to provide the tracker with the advantage of having prior knowledge about the search window for tracking.

6.4 Widely Used Basic Tracking Algorithms

6.4.1 MeanShift Tracking

Meanshift (MS) [124] is a non-parametric (not based on initial parameter to create probability distribution, the parameters are generated from training) approach for tracking, based on region matching. Therefore, it does not require declaring any fixed parameter (like any initial value, mean *etc*) for starting the tracking. It is an iterative procedure of modelling object based on kernel weighted colour histograms [125]. Object tracking by meanshift is very efficient for tracking objects, which are suitable for expressing by their histogram and not limited to any individual colour.

The objects (micro-particles) imaged in the microfluidic experiments are not rigid, although the particles (*e.g.* magnetic micro-particles) themselves may be rigid, this is not always the case in LOC based imaging (*e.g.* bubbles, which may deform).

Furthermore, in this case the captured image is subject to motion blur resulting in object shapes which are dependent on their velocity and the frame capture rate. This variability means the development of a 2D parametric model was not possible. Therefore, Meanshift, with some modifications, and combined with other models was identified as a suitable approach.

According to the basic Meanshift algorithm, after detecting an object for the first time, the total area of the object is defined using a rectangular bounding box (this rectangle covers most of the object whilst covering the lowest amount of background). Then the histogram of the object within the bounded rectangle is found. Later attempts are made to match the histogram of the target object from the current frame with the histogram of candidate regions from the successive frame until a sufficiently good match is found. In this way the histograms of the target and candidate models are compared. In many cases it is not possible to match the whole histogram for an object within a bounded rectangle (as the object or background surrounding the object within the bounded rectangle get changed). In this situation, maximum correlation between the histogram of the object from previous frame and current frame is calculated using Bhattacharyya coefficient [126]. If the comparison (Bhattacharyya coefficient) has a value less than a threshold, then it is considered that it is the same object.

Figure 6.2 shows the basic approach for Meanshift tracking:

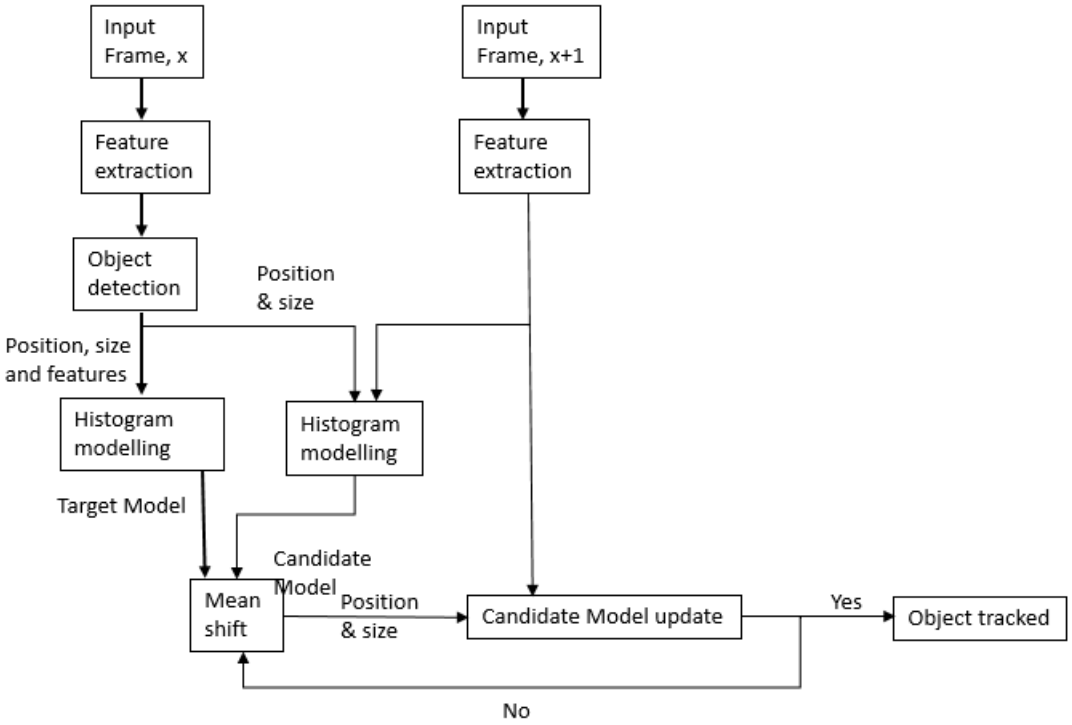


Figure 6.2: Generalized block-diagram of standard mean shift tracking algorithm. The MS-tracking-algorithm requires a histogram object model. Commonly colour intensities are used as input for the histogram modelling, but also more powerful features can be used. The feature extraction block can also be combined with the histogram model blocks, to reduce the number of features that are calculated [125].

The basic mean shift operation iteratively moves the centre of the region of interest to the centre of mass of that region, until the centre of the region of interest is sufficiently close to the centre of mass. This whole process of matching the centres can be shown using a flow diagram shown in Figure 6.3

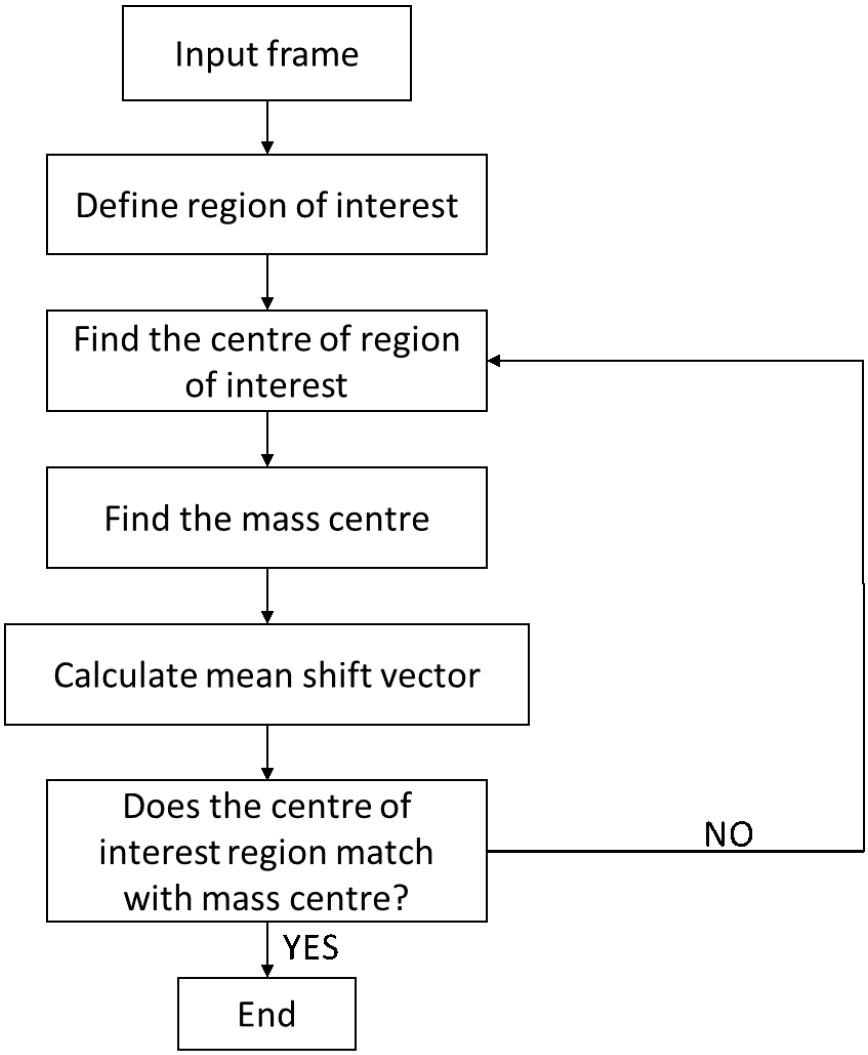


Figure 6.3: Matching of the objects during comparing the target and candidate region in Meanshift.

Meanshift uses a "density gradient estimator" iteratively for computing the local maximum for achieving the most similarity (using Bhattacharyya coefficient) to the sample distribution. This iterative process can be divided into four basic steps as shown in Figure 6.4. In Step 1, the area under the Blue circle is the region of interest with A as centre of the region and B is the mass centre. According to the Meanshift, the regional centre A will move toward B to match both centres (Step 2). Then B becomes the new centre for the region of interest (Step 3). The new region of interest will then have C as mass centre (Step 3). Then the centre for the region interest will move from B to C. Finally, C will be the point of both region interest and mass centre (Step 4).

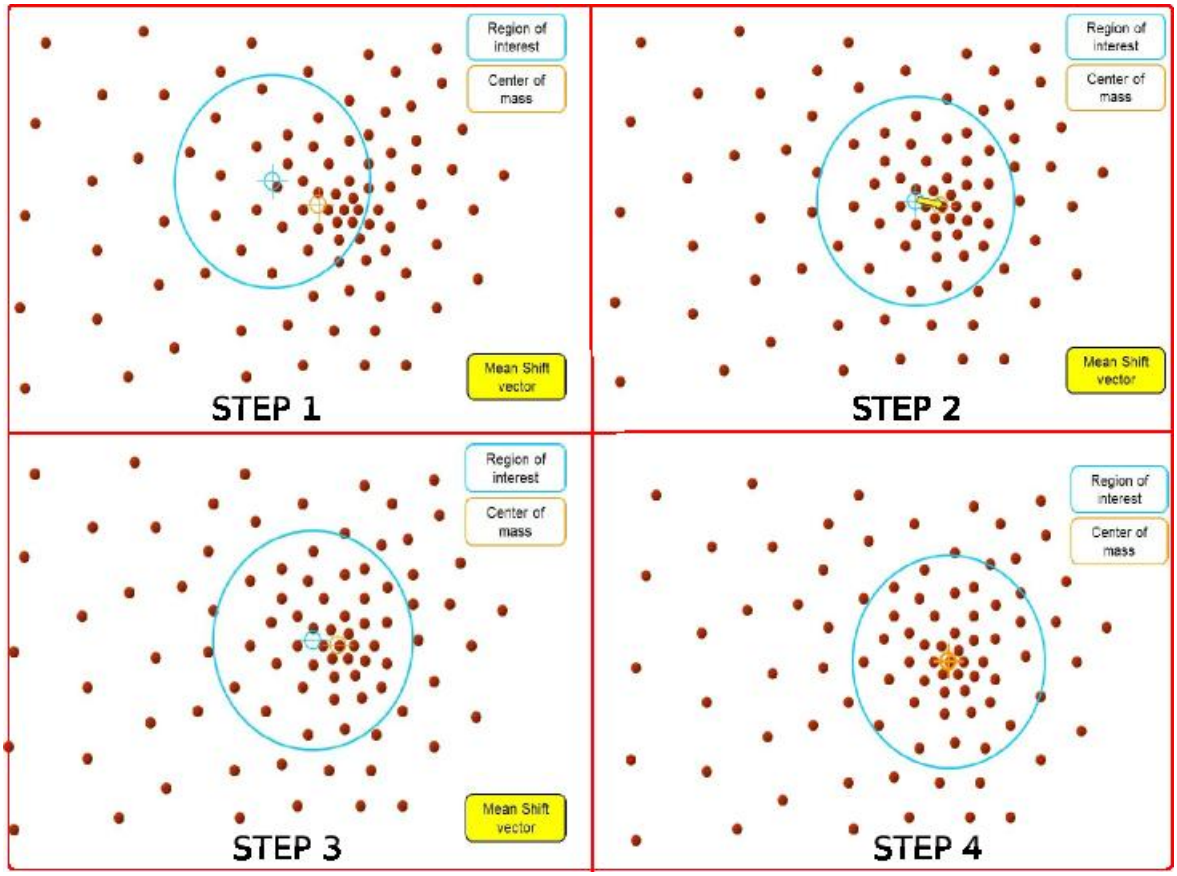


Figure 6.4: Steps for Meanshift to match the centre of mass and centre for region of interest [127].

Therefore, if $m(x)$ is the center of mass and x is the regional centre, then mathematically mean shift $(M(x))$ can be expressed as Equation (6.1):

$$M(x) = m(x) - x \tag{6.1}$$

There are two factors, which are required for the correct movement of the centre of the region of interest: 1) the average distance between centres (x) for region of interest and mass centre $(m(x))$ and 2) the movement direction.

For within a region of interest containing n data points $(x_i, i = 1, 2, 3n)$ within a d-dimensional Euclidean space (R^d), the basic average distance with respect to a point x is shown using Equation (6.2):

$$m(x) = \frac{1}{n} \sum_{x_i \in S_h} (x_i - x) \tag{6.2}$$

Where the region S_h is a hypersphere of radius h, centred on x and containing n data points (Figure 6.5).

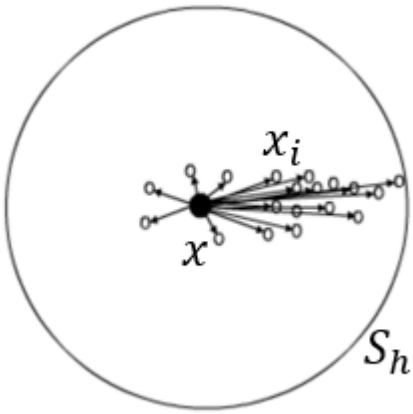


Figure 6.5: Illustration of the meanshift movement possible directions [128].

In Equation (6.2), $(x_i - x)$ provides the distance of x_i from x and mean gives the average of those distances for n samples within the S_h sphere.

From the basic equation of mean and mean shift, it is easy to get the average (mean) of the distances between sample point x_i and center x. However, for finding the direction for the movement of centre, it requires application of a Kernel estimation function. It (Kernel estimation function) considers different sample points with different weights, the specific weighting depending on the Kernel used.

Now multivariate Kernel density estimator ($\widehat{f_k}(x)$) with kernel profile $K(x)$ for n data points ($x_i, i = 1, 2, 3.....n$) within a d- dimensional Euclidean space(R^d) can be defined as Equation (6.3):

$$\widehat{f_k}(x) = \frac{1}{n}\sum_{i=1}^{n} K(x - x_i) \tag{6.3}$$

In this equation the kernel K gives more weight to pixels whose locations x_i are closer to the centre of the target. As the considered sphere (S_h) has radius h and it is situated within d- dimensional space, using these considerations, Equation (6.3)

can be expressed as Equation (6.4):

$$\widehat{f}_k(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{6.4}$$

In Equation (6.4), $K(x)$ is a radially symmetric kernel satisfying Equation (6.5):

$$K(x) = k\left(||x||^2\right) \tag{6.5}$$

Here, $k(x)$ is the profile of the kernel, which is

1. non-increasing $k(a) \geq k(b)$ for $a < b$) and
2. is piecewise continuous $\int_0^\infty k(r)dr < \infty$

Out of several known kernel profile (like Flat kernel, Gaussian kernel), the Epanechnikov kernel profile is the most commonly used in the mean-shift tracker. It can be expressed as:

$$k(x) = \begin{cases} \frac{3}{4}(1 - x^2) & \text{if } |u| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Graphically the Epanechnikov kernel profile can be represented using Figure 6.6.

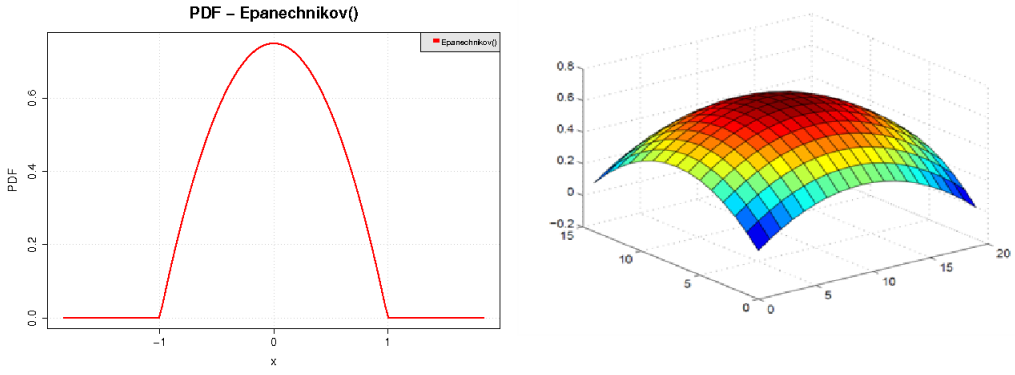


Figure 6.6: Profile for the Epanechnikov kernel (left) and its corresponding three-dimensional spatial representation for a circular image region (right). This Epanechnikov kernel mask is for a 100 × 80 image [129].

Combining Equation (6.4) with Equation (6.5), it (Equation (6.4)) can be rewritten as Equation (6.6):

$$\widehat{f}_k(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left|\left|\frac{x-x_i}{h}\right|\right|^2 \tag{6.6}$$

The use of a differentiable kernel allows us to define the estimate of the density gradient as the gradient of the kernel density estimate. So after differentiating Equation (6.6), it comes:

$$\widehat{\nabla} f_k(x) = \nabla \widehat{f_k}(x) = \tfrac{2}{nh^{d+2}} \textstyle\sum_{i=1}^{n}(x-x_i)k'\left|\left|\frac{x-x_i}{h}\right|\right|^2 =$$

$$\frac{2}{nh^{d+2}}\left[\sum_{i=1}^{n} k'\left|\left|\frac{x-x_i}{h}\right|\right|^2\right]\frac{\sum_{i=1}^{n} x_i k'\left|\left|\frac{x-x_i}{h}\right|\right|^2}{\sum_{i=1}^{n} k'\left|\left|\frac{x-x_i}{h}\right|\right|^2} - x \tag{6.7}$$

Now by considering a kernel function $g(x) = -k'(x)$ for estimating the kernel density $k(x)$ (kernel $G(x) = gx^2$). Introducing $g(x)$ in Equation (6.7) it becomes:

$$\widehat{\nabla} f_k(x) = \frac{2}{nh^{d+2}}\left[\sum_{i=1}^{n} g\left|\left|\frac{x-x_i}{h}\right|\right|^2\right]\left[\frac{\sum_{i=1}^{n} x_i g\left|\left|\frac{x-x_i}{h}\right|\right|^2}{\sum_{i=1}^{n} g\left|\left|\frac{x-x_i}{h}\right|\right|^2} - x\right] \tag{6.8}$$

In Equation (6.8), the first part is the density estimate at x with kernel G and the second part is the basic mean shift equation. So the density estimation function is:

$$\widehat{f}_g(x) = \frac{1}{nh^d}\left[\sum_{i=1}^{n} g\left|\left|\frac{x-x_i}{h}\right|\right|^2\right] \tag{6.9}$$

Using Equation (6.8), the equation for mean shift can be written as:

$$M_g(x) = \left[\frac{\sum_{i=1}^{n} x_i g\left|\left|\frac{x-x_i}{h}\right|\right|^2}{\sum_{i=1}^{n} g\left|\left|\frac{x-x_i}{h}\right|\right|^2} - x\right] \tag{6.10}$$

Equation (6.8), (6.9) and (6.10), can be rewritten as:

$$\widehat{\nabla} f_k(x) = \widehat{f}_g(x)\frac{2}{h^2}M_g(x) \tag{6.11}$$

So, the final meanshift equation can be represented as Equation (6.12):

$$M_g(x) = \frac{2}{h^2} \frac{\widehat{\nabla} f_k(x)}{\widehat{f}_g(x)} \tag{6.12}$$

Thus, the mean shift vector always points toward the direction of maximum increasing density.

For meanshift tracking, selecting the right kernel is considered as one of the most crucial parts. It determines the size of the search window and within this the sample weights are determined. This is also proportional to the expected image area of the target. Normally, this kernel scale is initialised by the first tracking window and its fixed during the rest of the tracking process. So when the scale of the target changes significantly, it results in failure of the tracking.

6.4.2 Bayesian theorem based Tracker

Prediction and update based object trackers (like Kalman Filter, Particle filter *etc*) are widely used to overcome problems of occlusion and temporary lost object(s). These trackers use three steps prediction, measurement and update. Though there are few prediction and update based trackers available, for this work only the Kalman filter will be discussed. For prediction and update the Kalman filter uses the widely used and well established Bayesian theorem. In the following sections, the very basics of Bayesian estimations and then the Kalman filter will be discussed.

Bayesian estimation

The well-known Bayes theorem was originally developed by the British statistician back in 1763 [130], who developed the fundamental probability law. French mathematician Pierre-Simon de Laplace later rediscovered the modern form of Bayesian theory [131], which has then been successfully implemented in statistical estimation and decision, modern machine learning and pattern recognition. Bayesian theory is considered as a branch of mathematical probability theory which models uncertainty via incorporating prior knowledge and observational measurements [130, 131, 132].

In 1960, the Kalman filter [133] [134] was first published, derived from Bayesian theorem it is considered as one of the most successful approaches in various engineering and scientific topics [134, 135]. The explanation of Kalman filter will be introduced in the next section.

In a dynamic system, estimation is normally required for optimal accurate results when a new measurement is received. For estimating the dynamic state, one way is to store the entire data set and analyse it to predict the posterior state. However, the calculation for the huge data set will be enormous. A recursive filter is a method developed for solving this problem. The Bayesian theorem can be used to make the prediction and provide an update for a recursive filter [136]. In the prediction step, a system model will be used to predict the posterior state. In the update step, the new measurement is received to update and modify the predicted result.

1. **System state model:** In a given dynamic system, the system state model can be expressed as Equation (6.13):

$$x_t = f_t(x_{t-1}, v_{t-1}) \tag{6.13}$$

   where f_t is the function which characterises how the state of the system will unfold over time. This function is also called a transition function or dynamic function. v_{t-1} denotes the system noise (also known as process noise).

2. **Prediction:** The Bayesian prediction of probability density function for the moment at time t will be as Equation (6.14):

$$\widehat{p}(x_t) = p(x_t|z_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1} \tag{6.14}$$

   where $p(x_t|z_{1:t-1})$ is the prediction of posterior distribution, and $z_{1:t1}$ is the prior measurements from time 1 to $(t1)$.

3. **Update:** The next step is to update the result according to the prediction and integrate the new measurement. A new measurement received at time t,

can be expressed as Equation (6.15):

$$z_t = h_t(x_t, n_t) \tag{6.15}$$

Where h_t is the function that measures the state (also known as the observation function). n_t denotes the measurement noise. Then the predicted state model Equation (6.14) will be updated as Equation (6.16).

$$p(x_t) = p(x_t|z_{1:t}) = \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} \tag{6.16}$$

Where

$$p(z_t|z_{1:t-1}) = \int p(z_t|x_t)p(x_t|z_{1:t-1})dx_t \tag{6.17}$$

In realistic work, it is not possible and necessary to keep all of the prior measurements, which generates huge amounts of calculation. Therefore, to reduce the complexity of the calculation, the estimation of current state normally just according to a complete summary of the past via the Chapman-Kolmogorov equation [137] which is derived from Markov Chain. Markov Chain [138] is defined as a sequence of data, where data at time point $(t + 1)$ only depends on data at time point t. However, using Chapman-Kolmogorov equation considers multiple steps of Markov Chains [138]. Meaning that estimating the probability at $(t + 3)$, can be estimate from time point t directly. So, the prediction (Equation (6.14)) and update (Equation (6.16)) can be updated as:

Prediction:

$$\widehat{p}(x_t) = p(x_t|z_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|z_{1:t-1})dx_{t-1} = \int p(x_t|x_{t-1})\widehat{p}(x_{t-1})dx_{t-1} \tag{6.18}$$

Update:

$$p(x_t) = p(x_t|z_{1:t}) = \frac{p(z_t|x_t)p(x_t|z_{1:t-1})}{p(z_t|z_{1:t-1})} = \frac{p(z_t|x_t)\widehat{p}(x_t)}{p(z_t|z_{t-1})} \tag{6.19}$$

Where

$$p(z_t|z_{t-1}) = \int p(z_t|x_t)p(x_t|z_{t-1})dx_t = \int p(z_t|x_t)\widehat{p}(x_{t-1})dx_t \tag{6.20}$$

Therefore, prediction (Equation (6.18)) and update (Equation (6.19)) are the fundamental steps of Bayes filter. In the real work, the difficulty is how to represent the probabilities. Lots of methods have been developed based on Bayes theorem, one of which is Kalman filter discussed below. This applies to the linear and Gaussian distribution situation.

Kalman Filter

The Kalman filter is a recursive filter used for solving prediction of a state, estimating optimal state and filtering noise problems. It is widely in use due to its simplicity, optimality, ability to track and robustness. This theory was named after Rudolf E. Klmn, one of the primary developers of its theory. The algorithm works in a two-step process: the prediction step and the measurement update (correction) step. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (which includes some error and random noise) is observed, the estimates are updated using a weighted average, with more weight being given to estimates with higher certainty.

For a dynamic and linear system, the models of motion (x_k) and actual measurement/observation (z_k) at time k can be expressed using following linear state Equations (6.21) and (6.22):

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + Gw_{k-1} \tag{6.21}$$

$$z_k = H_k x_k + v_k \tag{6.22}$$

Here

A is the state matrix ($n \times n$ dimension),

x_{k-1} is the process state vector at time $(k-1)$ (unknown and ideal),

B is the input matrix,

u_{k-1} is the input vector,

w_{k-1} is the process uncertainty,

G is process uncertainty matrix,

H is the measurement matrix, it is a noiseless connection between the state vector and measurement vector, which is stationary over time ($m \times n$ dimension)

v_k is the measurement uncertainty with zero mean ($\overline{v} = 0$).

Models of motion and actual measurement are matrices of $n \times 1$ and $m \times 1$ dimension respectively.

Now if Q_k and R_k are the process and measurement uncertainty covariance matrices respectively, then they can be related to the motion noise and measurement noise as Equation (6.23) and Equation (6.24):

$$Q_k = E\left[w_k w_k^T\right] \tag{6.23}$$

$$R_k = E\left[v_k v_k^T\right] \tag{6.24}$$

Here $E[.]$ means expected value.

The state vector changes from time $(k-1)$ to k, keeping a relationship with the previous state. So, with $w_{k-1} = 0$ the posteriori state estimate ($\widehat{x}_k^-$) vector at time k can be found using Equation (6.21) as:

$$\widehat{x}_k^- = A_{k-1}\widehat{x}_{k-1}^a + B_{k-1}u_{k-1} \tag{6.25}$$

When the input u_{k-1} is well known, then prediction error ($\widetilde{x}_k^-$) can be easily found by subtracting Equation (6.21) from Equation (6.25), as

$$\widetilde{x}_k^- = \widehat{x}_k - \widehat{x}_k^a \tag{6.26}$$

Replacing the values of x_k and $\widehat{x}_k^-$ in Equation (6.26) from Equation (6.21) and

Equation (6.25) it becomes:

$$\widetilde{x}_k^- = A_{k-1}(\widehat{x}_{k-1} - \widehat{x}_{k-1}^a) + Gw_{k-1} \tag{6.27}$$

Using Equation (6.26) in Equation (6.27), it comes:

$$\widetilde{x}_k^- = A_{k-1}\widetilde{x}_{k-1}^- + Gw_{k-1} \tag{6.28}$$

$\widetilde{x}_k^-$ is filtering error.

Using the definition of a priori covariance, error covariance matrix P_k at time k (before the update state) is:

$$P_k^- = E[\widetilde{x}_k^-(\widetilde{x}_k^-)^T] \tag{6.29}$$

After replacing the values in Equation (6.29) from Equation (6.27) and Equation (6.23), it comes

$$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + GQ_{k-1} \tag{6.30}$$

Now the measurement error or measurement innovation is the difference between the actual measurement obtained at time instant k and the measurement prediction obtained from the predicted value of the state, which is:

$$\text{Measurement Error} = z_k - H_k\widehat{x}_k^- \tag{6.31}$$

Now, with a Kalman gain of K_k at time k, the unbiased state estimate $\widehat{x}_k^-$ can be related to the Measurement Error as

$$\widehat{x}_k^a = \widehat{x}_k^- + K_k[\widehat{z}_k - H_k\widehat{x}_k^-] = (I - K_kH_k)\widehat{x}_k^- + K_kz_k \tag{6.32}$$

Substitution of Equation (6.25), (6.22) and using Equation (6.21) in to Equation (6.32) gives:

$$\widehat{x}_k^a = \widehat{x}_k^- + K_k[\widehat{z}_k - H_k\widehat{x}_k^-] = (I - K_kH_k)\widehat{x}_k^- + K_kz_k \tag{6.33}$$

Substitution of Equation (6.25) and (6.22) in to Equation (6.33) gives:

$$\widehat{x}_k^a = A_{k-1}\widehat{x}_{k-1}^a + B_{k-1}u_{k-1} + K_k[H_k A_{k-1}(\widehat{x}_{k-1} - \widehat{x}_{k-1}^a) + H_k G w_{k-1} + v_k] \quad (6.34)$$

Now, the posteriori error covariance matrix P_k^a at time k is

$$P_k^a = E[(\widehat{x}_k - \widehat{x}_k^a)(\widehat{x}_k - \widehat{x}_k^a)^T] \quad (6.35)$$

But

$$\widehat{x}_k - \widehat{x}_k^a = (I - K_k H_k)(A_{k-1}\widetilde{x}_{k-1}^- + G w_{k-1}) - K_k v_k = L_k(A_{k-1}\widetilde{x}_{k-1}^- + G w_{k-1}) \quad (6.36)$$

Considering, $L_k = (I - K_k H_k)$

So using the values from Equation (6.36) to Equation (6.35) and after several mathematical manipulation it comes:

$$P_k^a = L_k P_k^- L_k^T + K_k R_k K_k^T = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \quad (6.37)$$

Again the trace of a matrix is equal to the trace of its transpose, so the trace of the matrix P_k can be written as:

$$T[P_k^a] = T[P_k^-] - 2T[P_k^- H_k K_k] + T[K_k(H_k P_k^- H_k^T + R_k)K_k^T] \quad (6.38)$$

Differentiating Equation (6.38) with respect to K_k gives:

$$\frac{\mathrm{d}T[P_k^a]}{\mathrm{d}K_k} = -2(P_k^- H_k)^T + 2K_k(H_k P_k^- H_k^T + R_k) \quad (6.39)$$

Setting to zero and re-arranging Equation (6.39) gives:

$$(P_k^- H_k)^T = K_k(H_k P_k^- H_k^T + R_k) \quad (6.40)$$

Now solving for K_k gives:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad (6.41)$$

Equation (6.41) is the Kalman gain equation.

Finally, substitution of Equation (6.41) in to Equation (6.36) gives:

$$P_k = (I - K_k H_k) P_k^- \tag{6.42}$$

Now the overall Kalman filter equations can be divided into three sections initialization, prediction and filter as:

Prediction:

$$\widehat{x_k^-} = A_{k-1}\widehat{x}_{k-1} + B_{k-1}u_{k-1} \tag{6.43}$$

$$P_k^- = A_k P_{k-1} A_k^T + G Q_{k-1} G^T \tag{6.44}$$

Filtering:

$$\widehat{x}_k = \widehat{x_k^-} + K_k[z_k - H_k \widehat{x_k^-}] \tag{6.45}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{6.46}$$

$$P_k = (I - K_k H_k) P_k^- \tag{6.47}$$

With initial Conditions:

$$\widehat{x_0^-} = \overline{x} \tag{6.48}$$

$$P_o^- = P_o \tag{6.49}$$

The process for Kalman equations can be updated in five steps and these five steps can be shown using Figure 6.7.

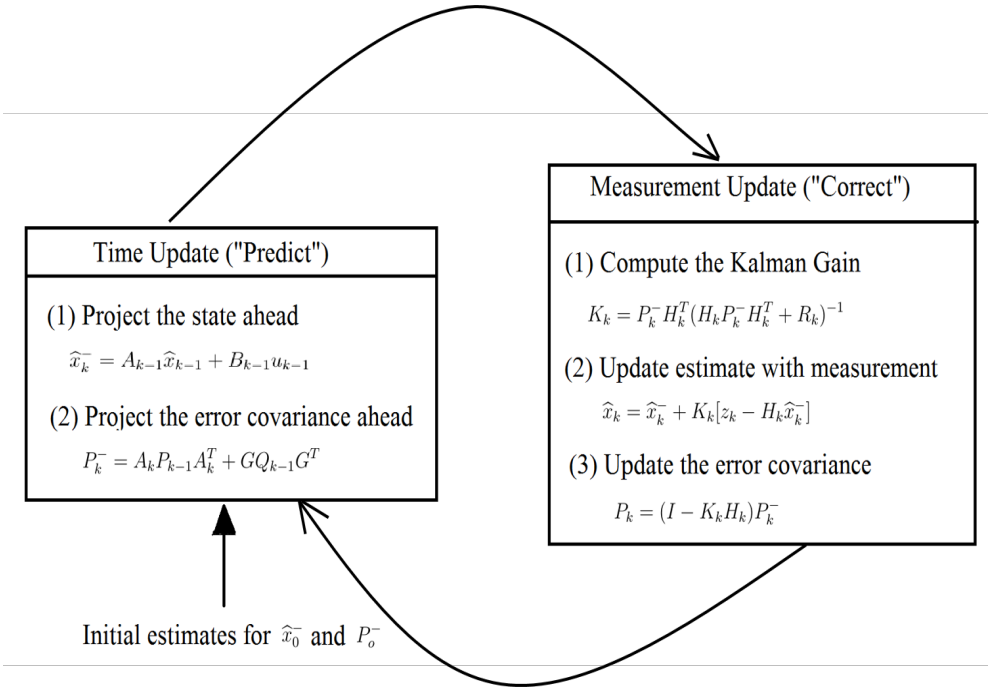


Figure 6.7: A complete picture of the operation of the Kalman filter, combining with the Equations (6.43), (6.44), (6.45), (6.46) and (6.47) [139].

6.4.3 Optical Flow

In any image sequence, the distribution of apparent velocities of brightness patterns is called optical flow. Optical flow can provide information of spatial arrangement and the rate change of this arrangement for each pixel of an object. If there is no moving object in the image sequence, optical flow within the image will be continuous. Otherwise, discontinuities optical flow can help to determine moving objects from images [140, 141]. However, optical flow methods have some difficulties, for example, the performances of optical flow techniques are low in poorly textured images which do not define the objects clearly. Another difficulty is handling the lighting conditions because optical flow assumes the brightness is always constant [142]. Therefore, optical flow is normally combine with other techniques for foreground segmentation.

Measurement of optical flow is required to be accurate and reliable. Many optical flow methods have been proposed during the last four decades. Among them, pop-

ular techniques could be categorized into four types, differential methods, region-based (block) matching, energy-based techniques and phase-based methods [142]. Differential based methods estimate the flow velocity and calculate the motion between two frames taken at two time points.

The most popular differential methods include Horn and Schunck [140], Lucas and Kanade [143, 144], Nagel [145] and Uras *et al* [146]. The Lucas and Kanade method is a good algorithm for determining small motion. Therefore, in this project, Lucas and Kanade optical flow method was implemented with improved Gaussian Mixture Model for background modelling and object detection.

Explanation of Lucas and Kanade method with equation

The Lucas and Kanade method is based on three assumptions: that the brightness of moving object is constant, motion is very slow compared with frame changes, and nearby pixels have the same velocity.

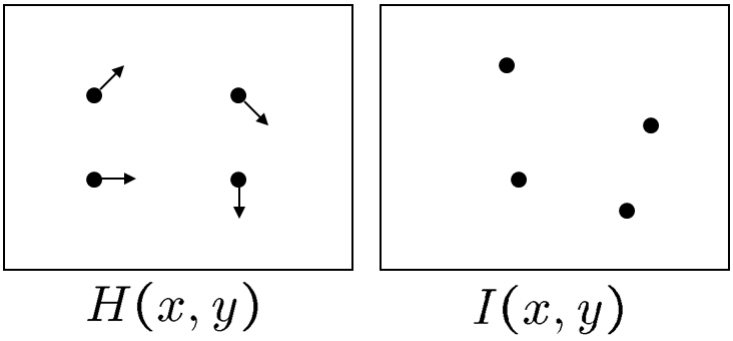


Figure 6.8: Illustration of pixels movement from image frame H(x,y) to frame I(x,y).

Therefore, to estimate pixel motion from image frame $H(x, y)$ to image $I(x, y)$ (Figure 6.8), first assumption is that the brightness of moving object is constant. If the displacement is (u, v), it gives:

$$H(x, y) = I(x + u, y + v) \tag{6.50}$$

Then assuming that object is moving very slowly, in this condition, the displacement (u, v) will be less than one pixel. Taking the Taylor series expansion, Equation

(6.50) becomes as Equation (6.51):

$$I(x+u, y+v) = I(x,y) + \tfrac{\partial I}{\partial x}u + \tfrac{\partial I}{\partial y}v + \text{higher order terms}$$

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \tag{6.51}$$

Combing Equation (6.50) and (6.51), it comes

$$\begin{aligned} 0 &= I(x+u, y+v) - H(x,y) \\ &\approx I(x,y) + \tfrac{\partial I}{\partial x}u + \tfrac{\partial I}{\partial y}v - H(x,y) \\ &\approx I(x,y) + I_x u + I_y v - H(x,y) \\ &\approx (I(x,y) - H(x,y)) + I_x u + I_y v \\ &\approx I_t + I_x u + I_y v \end{aligned}$$

$$I_t + \nabla I \cdot [uv] \approx 0 \tag{6.52}$$

The third assumption is that of locally constant and smooth motion, which means the pixels neighbours having same displacement (u, v). When using 5×5 windows, which gives 25 equations per pixel. hen Equation (6.52) will become

$$0 = I_t(P_i) + \nabla I(P_i) \cdot [uv] \tag{6.53}$$

Equation (6.53) will be expanded with $n \times n$ windows to

$$\begin{bmatrix} I_x(P_1) & I_y(P_1) \\ I_x(P_2) & I_y(P_2) \\ . & . \\ . & . \\ . & . \\ I_x(P_n) & I_y(P_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(P_1) \\ I_t(P_2) \\ . \\ . \\ . \\ I_t(P_n) \end{bmatrix} \tag{6.54}$$

Considering $A = \begin{bmatrix} I_x(P_1) & I_y(P_1) \\ I_x(P_2) & I_y(P_2) \\ . & . \\ . & . \\ . & . \\ I_x(P_n) & I_y(P_n) \end{bmatrix}$, $V = \begin{bmatrix} u \\ v \end{bmatrix}$ and $b = -\begin{bmatrix} I_t(P_1) \\ I_t(P_2) \\ . \\ . \\ . \\ I_t(P_n) \end{bmatrix}$

Then, Equation (6.54) can be written in matrix form as,

$$A^T AV = A^T b \tag{6.55}$$

Therefore,

$$V = (A^T A)^{-1} A^T b \tag{6.56}$$

Where A^T is the transpose of A.

Then Equation (6.57) can be written to

$$\begin{bmatrix} V_u \\ V_v \end{bmatrix} = \begin{bmatrix} \sum_i I_x(P_i)^2 & \sum_i I_x(P_i) I_y(P_i) \\ \sum_i I_y(P_i) I_x(P_i) & \sum_i I_y(P_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(P_i) I_t(P_i) \\ -\sum_i I_y(P_i) I_t(P_i) \end{bmatrix} \tag{6.57}$$

However, all n pixels P_i in the window are not same importance for the calculation. So the weights W is assigned to the equation which give more importance to the pixels closer to the central pixel point. Then Equation (6.56) becomes

$$V = (A^T W A)^{-1} A^T W b \tag{6.58}$$

Combining Equation (6.57) and (6.58) gives,

$$\begin{bmatrix} V_u \\ V_v \end{bmatrix} = \begin{bmatrix} \sum_i w_i I_x(P_i)^2 & \sum_i w_i I_x(P_i) I_y(P_i) \\ \sum_i w_i I_y(P_i) I_x(P_i) & \sum_i w_i I_y(P_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i w_i I_x(P_i) I_t(P_i) \\ -\sum_i w_i I_y(P_i) I_t(P_i) \end{bmatrix} \tag{6.59}$$

6.4.4 Template Matching tracking

Using the background subtraction techniques mentioned in the previous chapter, foreground objects can be determined. To perform tracking of these foreground

objects, the part of image containing an object can be used as a template to search for it in the next image frame. This technique is called template matching [147]. The algorithm of template matching is to find the object position by searching for the best matching part in the image by comparing with the template image (shown in Figure 6.9)

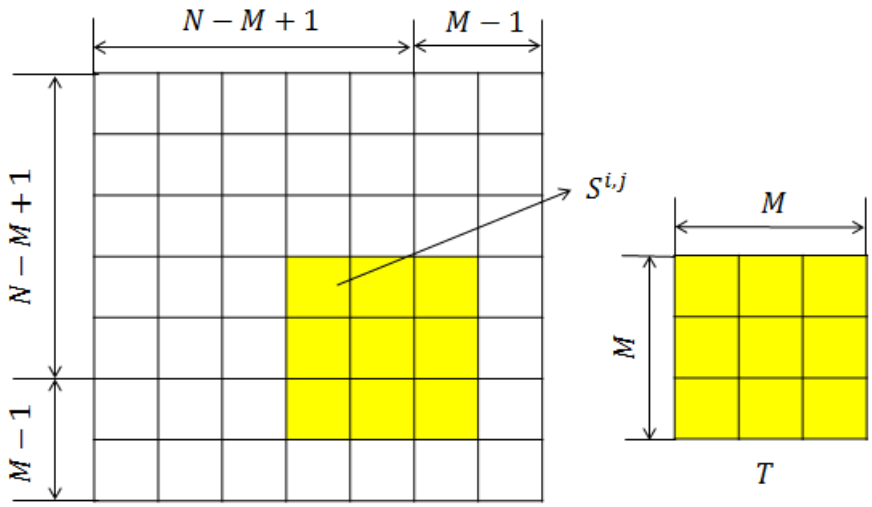


Figure 6.9: Template matching searching window model.

The template image can be represented in different ways by using various object properties - such as colour histograms, histogram of oriented gradients(HOG), intensity values [148] and more complex feature based points, shapes or surface models. Searching the image is performed via a moving template image from left to right and top to bottom by single pixel movement [149]. An object selected as a template for matching in the next frames can be seen in Figure 6.10:

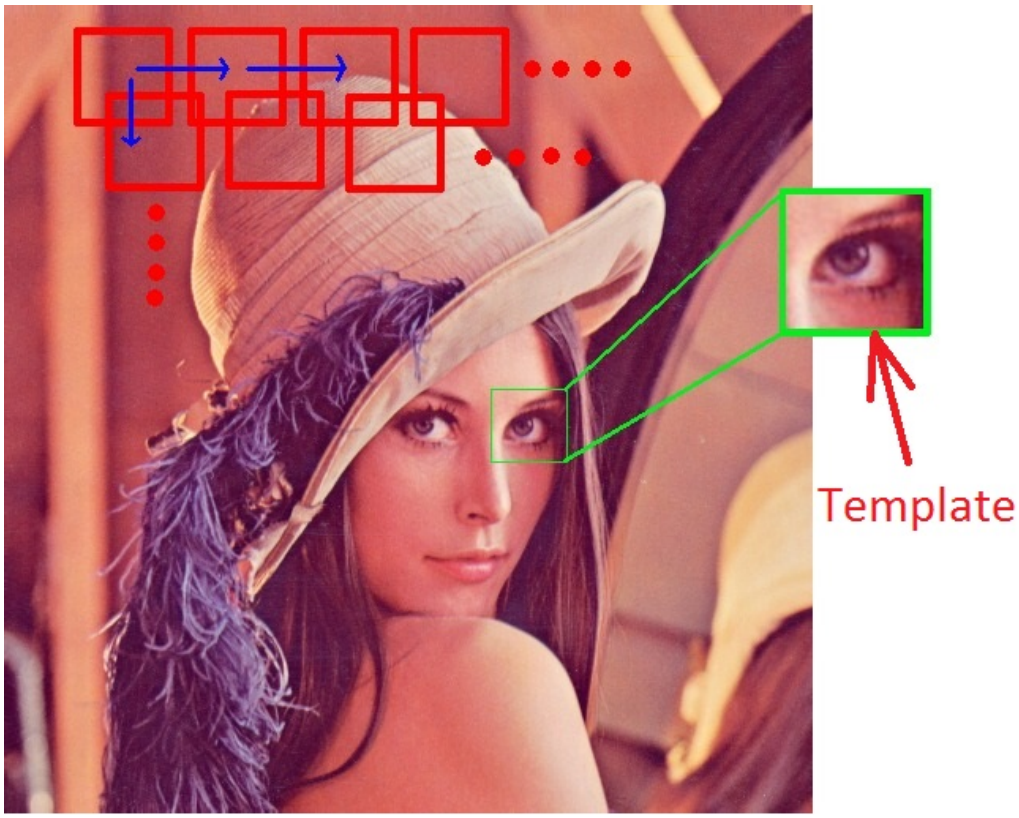


Figure 6.10: An image showing a selected object as template. Area under red rectangle indicates the template image which will be used for searching in the next image [150].

The next key part of this technique is how to measure the similarity between the searched image and the template image. Many types of algorithms have been developed for comparison techniques implemented in different situations. There are some popular comparison calculations which includes - mean absolute difference (MAD), sum of absolute differences (SAD), sum of squared differences (SSD), mean square differences (MSD), normalized cross correlation (NCC) *etc.*

6.5 Summary

Images from background subtraction process provide input for tracking. It is not always true that object(s) can not be tracked without background subtraction. But for automatic multiple object tracking from a crowded scenario (as in the LOC case), background subtraction provides the best results. Moreover, for multiple object tracking from a crowded situation it was found that none of the single tracking meth-

ods provide a good result and it was required to combine multiple approach. This chapter provided some basic information about difficulties of multi-object tracking. Also description of several useful and widely used tracking method also provided here. In the following chapter the specific tracker implementation used for the micro-particles in the LOC experiments will be described.

Chapter 7

PROPOSED MULTIPLE OBJECT TRACKING

Most of the established state estimation (object tracking) methods discussed in previous chapter (Chapter 6) are suitable tracking single objects or multiple objects which keep a reasonable distance between each other over the image sequence. Moreover, it is also required that the target objects contain distinguishable feature(s) compared with the surroundings (background and other object) for making them differentiable from one another. However, tracking multiple objects from a scene where the objects are a very small distance apart and the number of objects is high, additional factors need to be considered compared with the tracking methods discussed in Chapter 6. For multiple object tracking one of the main problems is data association, that is relating the object measurement(s) with each of the predicted states [151]. Other difficulties like false detections, missed detections and objects being in close proximity to one another also make multiple object tracking challenging. Moreover, for efficient tracking, an object should not have significant change in its position between the current frame and the next one, and also its velocity should be reasonably stable [152]. But in real-life, these optimum situations rarely happen.

For solving such problems widely used approaches can be divided into two classes

- 1) approaches for finding the optimum solution and 2) statistical correspondence methods. The Hungarian Method [153] which finds an optimal solution in polynomial time and Rittscher *et al.* [154] (global optimisation method based on expectation maximisation, to divide crowds into individuals) are considered as to be the most widely used optimum solution finding based approaches. On the other hand, Kalman filter, particle filter *etc* use the statistical correspondence approach.

The performance of the basic tracker (Meanshift, Camshift, template matching *etc*) provides moderate performance for tracking single objects having distinguishable feature(s). But for multi object tracking, all these basic trackers fail, moreover if the scene is crowded (that is if the objects are close to each other) then the number of false detections become much higher. So it was found that for tracking multiple objects a combination of a basic tracker (like Meanshift, template matching) and an optimum solution finding approach or statistical correspondence finding approach provides the best performance.

The input sequences that were required to be analysed for this thesis contained micro-particles as the objects to track. This micro-particles do not have distinguishable features to differentiate them. Also, depending on the experiment, the input sequence can be too crowded. So considering all these situations, in the proposed approach, a basic object tracker was combined with a statistical correspondence finding approach for tracking micro-particles.

As a basic tracker, a combination of improved template matching, Meanshift (MS) and optical flow performed well tracking single objects and also performed well with multiple object to some extent. Also this combination was able to solve the drawback of the tracking window drifting away from the target [155] when using Meanshift. This combination of trackers was given the name "Hybrid Meanshift Tracker". As another very common problem for the image sequence is "object becoming invisible" (that is a visible object become invisible for next few frames or rest of the frame), so the Kalman filter was combined with the "Hybrid Meanshift tracker" to overcome this drawback. In the following section, will discuss in depth the proposed approach for tracking multiple objects.

7.1 Proposed Algorithm overview

Tracking starts with detecting the objects of interest from the input sequence. This object detection process is done using the background subtraction techniques discussed earlier in Chapter 5. Background subtraction acts as a very useful way of locating moving objects (also non-moving objects after doing the required modifications) in any scene. Also a robust background subtraction model can provide the tracker with robustness against numerous difficulties, like changes in lighting, low objectbackground contrast *etc.* Details of these issues has already been discussed in Chapter 4 (Background subtraction review). In this Chapter, it is assumed that the tracker gets the detected objects' locations from the background subtracted image sequence directly.

In this tracking approach from the first frame onwards, whenever a new object gets detected it is assumed to be an input for Meanshift tracking, then Meanshift tries to match the centres of object's target and candidate regions. For finding the match between these two regions most typical applications use a 2-D colour histogram comparison. However, for the objects (chemically activated micro-particles) their 2-D colour histogram distributions are very similar, which made the use of colour histogram impossible. So an alternative way for Meanshift to compare and match the centres of object's target and candidate regions was required.

After lots of investigation, it was found that the intensity gradient over the detected object and its orientation could be a choice for matching the same object in consecutive frames. The intensity gradient (including value and orientation) over an object is almost the same within two frames for the same object. So the use of a Histogram of Oriented Gradients (HOG) [156], for matching the same object in two consecutive frames was very suitable in this case. In this way, a template of a detected object was made (using HOG) and it was used for finding the match between centres for object's target and candidate regions.

Using the above approach, it was found that if there were only one micro-particle available to track, or the particles stayed a good distance from each other, Meanshift

combined with template matching (HOG matching) performed efficiently. But for situations, where the objects stay very close to each other and many particles follow the same path across the reaction chamber the above technique produced lots of false detections. So it was required to find a way to restrict the search window of Meanshift, which was done using optical flow. Also, for tracking less visible or invisible objects, out of the several ways available such as particle filter, Kalman filter *etc.*, it was found that the Kalman Filter was very suitable for this task. So, when the "Hybrid Meanshift Tracker" fails to track any object a Kalman filter is used to predict the position of the current object in the next frame and continues the tracking. The overall process for the proposed tracking method is given in the following Figure 7.1.

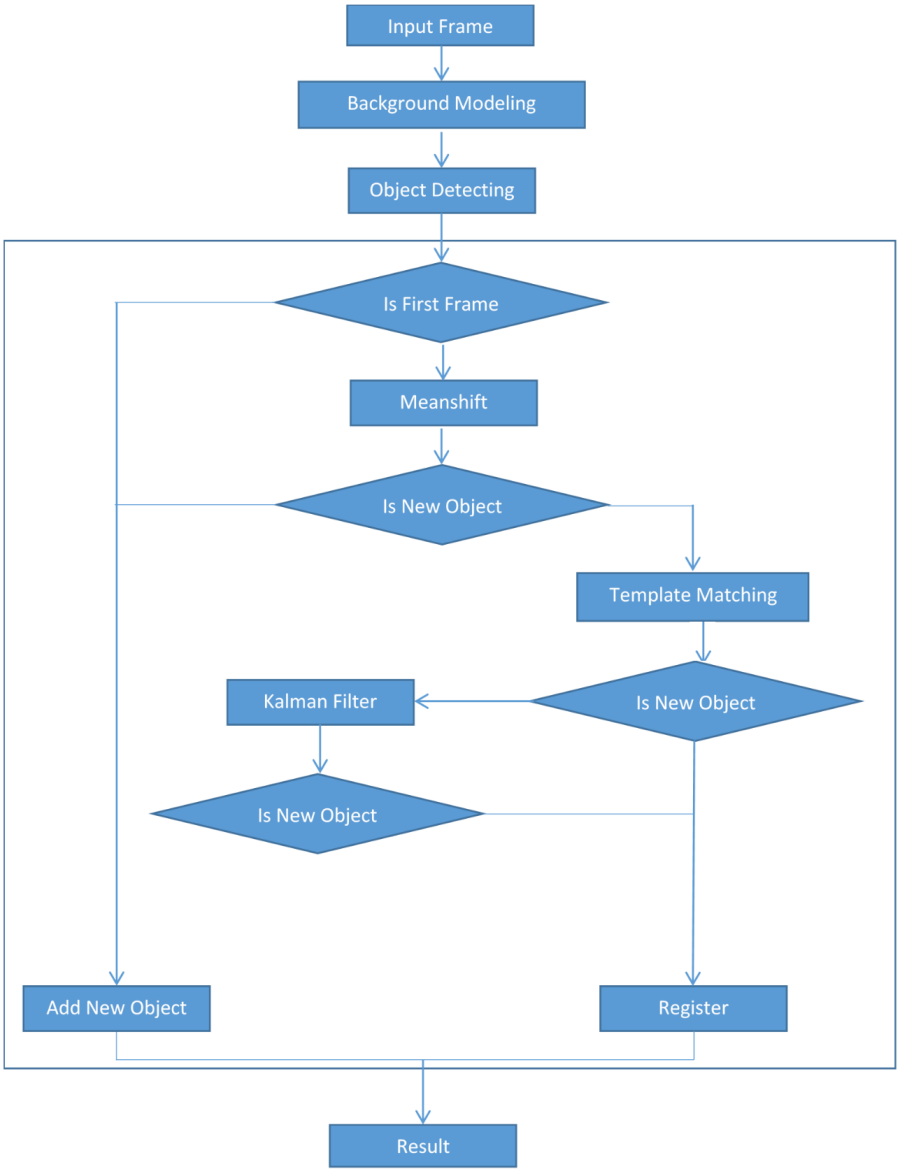


Figure 7.1: Overview of the proposed Algorithm. The detected object from the background subtraction goes to the tracking process. Combination of "Hybrid Meanshift" (Meanshift combined with template matching) with Kalman filter provide here the final track result.

In the following sections, all the steps of this tracking method will be discussed in brief. At the end of each step we will describe the process of combining one step with other will be described.

7.1.1 Hybrid Meanshift (combination Meanshift, Template matching (HOG matching) and Optical flow)

In the proposed approach of tracking objects (micro-particles), the tracking starts with the use of hybrid Meanshift tracking. Here, the introduced hybrid model for Meanshift tracking is based on combination of Meanshift, optical flow and template matching (Histogram of oriented gradients (HOG) matching). Optical flow from LucasKanade method (L-K method) is combined with the direction of Meanshift vector, to produce the resultant vector. This resultant vector restricts the search window for the Meanshift model by moving the search window towards the resultant vector direction. Also the distance for the search window movement is kept limited using the magnitude from the optical flow estimation. Details of the optical flow estimation (L-K method) and basic Meanshift have already been discussed in Chapter 6. After getting a limited area and range, for finding the right object from current frame to the next frame, the object template matching comes in to action. For matching the template, every object is represented using HOG and for finding the similarity between two templates (HOG) Bhattacharyya distance is used. Details of HOG, derived by Dalal and Triggs [156], are described in following Section. In this section only the process of combination of these methods will be discussed.

For this approach it was first assumed that the object motion is uniform, that is the objects in successive frames have a consistent relationship in terms of motion and direction. Also the speed of the object does not change dramatically. These assumptions were found reasonable to some extent for the objects from microfluidic experiments, as the movement of the microparticle in microfluidic experiments is mainly due to vector summation of magnetic force and force due to fluid flow. In general, these two forces for any point is same over a whole experiment. So in theory the object (microparticle) movement is uniform and constant over a small area. Later these assumptions were combined with some other practical observations to make the tracker robust. Which is discussed in Section 7.1.2.

According to the "Hybrid Meanshift" approach after a new object is detected,

Meanshift must first be applied for finding the possible changes between two consecutive frames. In this way, a window for the target location in the current frame is found using the predefined kernel in the first frame. At the same time, when Meanshift detects an object (in each frame, *i.e.* both new and old) the bounding area's Histogram of oriented gradients (HOG) get registered as a template of the object, which will be later used for the later purpose of template matching.

In this situation, for restricting the Meanshift search window and limiting its direction of movement optical flow estimation is introduced. Using optical flow estimation, it is possible to obtain information about object motion by computing the optical flow between consecutive frames, then applying the measurement of uniformity of motion on it. There are many well established methods available for computing optical flow between frames. Here it was done using Lucas-Kanade's [144] gradient-based optical flow measurement, as this method provides the most accurate and computationally inexpensive optical flow estimations [120]. Details of the L-K method have already been discussed in Chapter 6, in Section 6.4.1.

Finding optical flow requires two successive frames $f_(t-1)$ and f_t at time $(t-1)$ and t. For a 2D image L-K's estimation provides a vector of the optical flow field $(\overrightarrow{V})$ between two consecutive frames which contains the magnitude and direction of the movement, as shown in Equation (7.1):

$$\text{Optical Flow } (V_{x,y,t}) = \begin{pmatrix} Magnitude, V_{mag,t} \\ Direction, V_{dir,t} \end{pmatrix} \tag{7.1}$$

The output from L-K optical flow estimation can be represented using the following Figure 7.2:

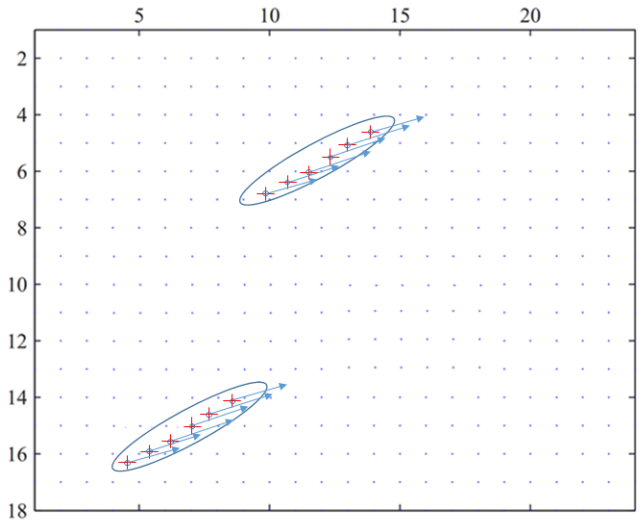


Figure 7.2: Dense optical flow computation for two consecutive frames.

After finding the optical flow for the current frame from the previous one, uniform motion detection is applied by selecting the pixels with non-zero optical velocities. The remaining pixels having similar motion direction and magnitude are grouped as the same component. For the hybrid object tracking approach, every two consecutive frames are used for calculating the optical flow but optical flow is not used as a tracking method. It is only used to limit the search window direction and magnitude of the Meanshift.

All this above information from the current location of the object is used for finding its potential position window in the consecutive frame. This way, a candidate target location with predefined kernel size is found in the next frame. The HOG feature is detected inside this MS candidate window, then a match between the current HOG feature and previously registered HOG feature is sort. For finding the matching between two HOG features the Bhattacharyya distance is used. In this way, for each pair of image boxes $W_{l,j}$ and $W_{r,j+d}$ for frame f_{t-1} and f_t respectively, the Bhattacharyya distance ($d(H_1, H_2)$) between their HOG can be computed as Equation (7.2)

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{H_1}\,\overline{H_2}\,N^2}} \sum_I \sqrt{H_1(I).H_2(I)}} \tag{7.2}$$

Where N is the number of descriptor bins of H_1 and H_2 HOG, I is the position within the histogram partition. This distance is then compared with a user settable value (τ), which is fixed throughout the whole tracking process. It was found for the microfluidic dataset an input of, $\zeta \leq 2.50$ provided good detection, so it can be expressed as Equation 7.3:

$$\text{Match} = \begin{cases} 1 & \text{for } \zeta \leq 2.50 \\ 0 & \text{otherwise} \end{cases}$$

In the present framework, the tracking window always gets an updated HOG template ready to match the next frame for the same object, unless it finds the object is non-moving.

The above discussed hybrid tracker performed well for an object having uniform motion. But in real-life cases it was found that microparticles get stuck to each other or with micro-pillars (part of the chip structure) and become motionless either for few frames or for the rest of the image sequence. This motionless situation can be the result of many other reasons too. The motionless objects also change their shape a lot from their previous motion condition, for example chemically activated micro particles are normally elliptically shaped when they are moving but become more circular when they are stationary. Such changes happen suddenly due to their relative small number of frame captures per second. In this situations, when the objects change their shape too fast, it becomes impossible for the Meanshift to match their templates using the HOG feature.

So the tracking of non-moving objects was not possible using the above procedure. For solving this problem, help was taken from the three frame difference technique used in background subtraction. It has already been discussed in Chapter 5 that if an object does not move (which was initially detected as an object) then it can be detected using the three frame difference technique. So when the three frame difference technique detects any non-moving objects, the tracker combines this information for finding the next position of this object and keeps the same id for the same object at the same position. In this case, the HOG histogram features

do not get updated until it starts to move again. If the object starts to move again only its previous information is used for matching the tracked object.

Histogram of oriented gradients (HOG)

Histograms of oriented gradients (HOG) proposed by Dalal *et al.* [156] mainly represent the targets contour orientation information, that is it uses the object's low level features. For calculating these features, gradient orientations within the region of interest (ROI) (also known as cells) are calculated and represented as histograms. The equations for calculating the HOG features are given as Equation (7.3), (7.4), (7.5) and (7.6):

$$\Delta I_x = I(x+1, y) - I(x-1, y) \tag{7.3}$$

$$\Delta I_y = I(x, y+1) - I(x, y-1) \tag{7.4}$$

$$\theta(x, y) = \tan^{-1} \frac{\Delta I_x}{\Delta I_x} \tag{7.5}$$

$$m(x, y) = \sqrt{{\Delta I_x}^2 + +{\Delta I_y}^2} \tag{7.6}$$

In these above equations (Equation (7.3), (7.4), (7.5) and (7.6)), I is the brightness of each pixel, ΔI_x and ΔI_y are the pixel intensity differences in the x and y directions respectively, m is the magnitude of the gradient and θ is the direction of the gradient. After getting all the directions of gradient (θ), it is required to divide them into groups. Dalal *et al.* [156] divided into groups with a range of 20 degrees. This way 9 groups of orientation from 0^0 to 180^0 (as the sign is not considered) are obtained. In this stage the features are required to be normalise (f_N) within each cell using Equation (7.7):

$$f_N = \frac{\upsilon}{\sqrt{\sum_{i=2}^{k} \upsilon^2 + e^2}} \tag{7.7}$$

where, v is the HOG feature, k is the number of HOG features in the block. Adding $e^2 = 1$ in Equation (7.7) prevents the result from becoming undefined. An illustration of the HOG feature extraction for a microparticle can be seen in Figure 7.3.

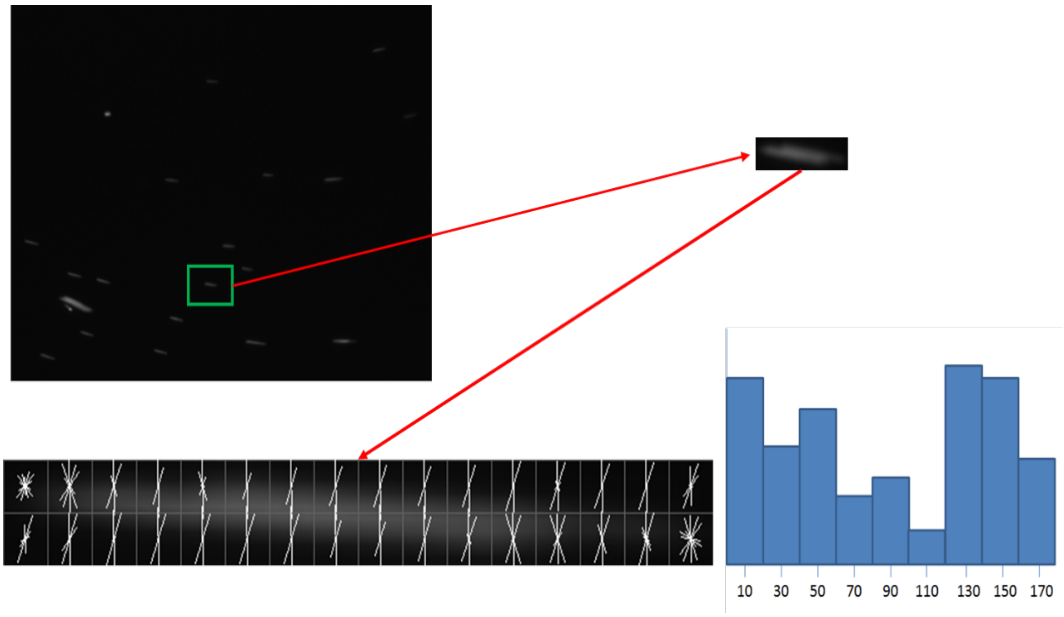


Figure 7.3: Sample for a normalised HOG feature of an Micro-particle. Top most left is the input image (selected object within the green box), the top right side figure is the gradient image. In the second row, the first image is the extracted features of the object. The last image is showing the normalised histogram of the features (histogram is just for illustration purpose, not from program result).

Normally for the objects that need to be tracked, pre-processing and noise reduction is considered during the background modelling implementation stage. Moreover, the approach is actually tracking the object mask from the previous step of background subtraction. As hybrid background model for subtracting the background has already been implemented, so it did not require the implementation of noise filtering to decrease the effect of noise. But for a technique using a less robust background model, it may be useful to use Gaussian frequency domain filtering before calculating HOG.

7.1.2 Kalman Filter

Even after using the "Hybrid Meanshift" tracking model still the problems of tracking objects from input sequence that are crowded objects and objects that travel keeping a small distance from each other is unsolved. As in this situation "Hybrid Meanshift" fails to find the correct target region for the object of interest, so it fails to find the proper match. For these input sequence images, it was found that within a small area of the image lots of objects are moving with a small distance between each other.

Another very common issue in microfluidic input sequences is micro-particle(s) becoming invisible or hardly visible within the captured frame for a few frames or for the rest of the frames in the sequence. So when an object becomes invisible or hardly visible, it is not possible to match the HOG features and then the tracker failed either by providing a negative result (no object detected) or a false positive result (by detecting another object).

To address both these situations, it was required to find a solution where the tracker would not depend on probabilistic density or feature matching for tracking an object. It was found that implementation of a Kalman filter with the above discussed hybrid algorithm could provide tracking results even if an object became invisible. Moreover, the results from magnetophoresis experiments showed that the motion path can be described as a linear equation for a short distance, that is the objects' movements are almost linear. So use of a linear Kalman filter provided acceptable results for tracking crowded and suddenly invisible objects. However, in very rare cases for the input sequence from the experiment micro-particles showed non-linear behaviour, for these cases the Kalman filter performed poorly. But these types of situations occurred mainly due to a poor level of expertise of the experimenter and may be considered as failed or poorly configured experiments. They are not of significant concern for the development of an automated system.

To implement Kalman tracking the first step was to model the system. To model the system according to a Kalman filter (discussed in Chapter 6), required several

initializations. After initializing all the parameters and states of the Kalman filter, it is required to know when it will be in action. Inappropriate use of Kalman filter with Hybrid Meanshift can make the total tracking unsuccessful. That's why Kalman tracking is used for estimating a new location when the value for the Bhattacharyya distance during the process of hybrid tracking failed to satisfy the limit condition. That is if the likeness between the target and candidate models is less than the required value and the target object is not decided as a non-moving object then the Kalman tracker is added to the algorithm for estimating the new location of the object in the next frames. In this project for tracking micro-particle, the limit was chosen as 1.50.

At first a vector was constructed for each particle which included the position and speed of the object within the frame. As a position Kalman filter requires a point, the object's centre of gravity was chosen as the location of the object. The initial velocity was determined by measuring the location changes between two frames and the time difference between the two frames. Therefore, the motion of the tracked object can be modelled using Equation (7.8):

$$\dot{X} = AX + w \tag{7.8}$$

As the measurement of the object's location gets corrupted by uncorrelated noise, so the measurements can be expressed as a linear combination of the system state variables and noise. So the m-dimensional measurement vector can be modelled as Equation (7.9):

$$Y = HX + v \tag{7.9}$$

The state transition matrix F express the system's dynamics, normally it's a

constant matrix, which can be shown as Equation (7.10).

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.10}$$

In Equation (7.10), Δt is inverse of the frame rate per second (fps).

The measurement matrix, H in Equation (7.9) can be expressed as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{7.11}$$

As the position and speed of a moving object are the states of the object and they are two dimensional (X and Y), so the state of an object can be expressed as Equation (7.12):

$$X = [x(n) \quad y(n) \quad V_x \quad V_y]^T \tag{7.12}$$

In Equation (7.12), n is the number of the frame, $x(n)$ and $y(n)$ are the X and Y component of the objects position respectively. Similarly, $\dot{x}(n) = V_x$ and $\dot{y}(n) = V_y$ are the X and Y component of the objects speed respectively. These speed components can be expressed as Equations (7.13) and (7.14).

$$V_x = \frac{\Delta x}{\Delta t} = \frac{x_n - x_{n-1}}{t_n - t_{n-1}} \tag{7.13}$$

$$V_y = \frac{\Delta y}{\Delta t} = \frac{y_n - y_{n-1}}{t_n - t_{n-1}} \tag{7.14}$$

Therefore, the state and measurement equations of the Kalman tracker (shown

in Equations (7.8) and (7.9)), can be expressed as (7.15) and (7.16):

$$\begin{bmatrix} x(n+1) \\ y(n+1) \\ \dot{x}(n+1) \\ \dot{y}(n+1) \end{bmatrix} = F \begin{bmatrix} x(n) \\ y(n) \\ \dot{x}(n) \\ \dot{y}(n) \end{bmatrix} + w \tag{7.15}$$

$$\begin{bmatrix} x(n) \\ y(n) \end{bmatrix} = H \begin{bmatrix} x(n) \\ y(n) \\ \dot{x}(n) \\ \dot{y}(n) \end{bmatrix} + v \tag{7.16}$$

Here the random variables w and v are the state and measurement noise respectively. For zero mean and with Gaussian process noise having known covariance Q and R respectively, w and v can be shown as Equation (7.17) and (7.18).

$$w \approx N(0, Q) \tag{7.17}$$

$$v \approx N(0, R) \tag{7.18}$$

Now, for getting better performance it is required to get appropriate values for Q and R. Another most important thing is to calculate the right initial values for the Kalman filter, which are $x(n)$, $y(n)$, V_x, V_y (these values get updated in each frame automatically). Incorrect initial values will provide incorrect estimation and this will lead to a failure in tracking. Here Δt remains almost constant over a process, as it is inverse of fps (frame per second) of the captured sequence. From Equation (7.15), the values of $x(n+1)$, $y(n+1)$, $\dot{x}(n+1)$, $\dot{y}(n+1)$ can be found which are the locations and speeds of the object in the next frame. If the likelihood between the target and candidate models is more than the requirement to use Kalman filter as a tracker, then the values relating to the object's location in the next frame are found using Hybrid Meanshift tracking. If Hybrid Meanshift tracking cannot calculate the values in Equation (7.15), then the object's location in next frames are

recalculated until the proper likeliness is found. This cycle of finding likeliness using either Hybrid Meanshift or Kalman filter continues until the end of the process.

7.2 Experimental Results

Like the proposed background modelling and object detection technique in Chapter 5, it was intended to develop a system which can perform tracking with high accuracy for different real-life scenarios. However, it was not possible to use the same approaches for micro-particle tracking as other objects like humans, cars *etc.* The tracking features are different for all of them. So in this section, the main target was to show tracking performance for input data sequences from magnetophoresis LOC experiments. Then also conduct experiments on tracking humans and cars from benchmark datasets. Though the results were not accurate enough in many cases, for objects without having strong long occlusion this method still performed efficiently.

The tracking results for tracking micro-particles in different situations (low concentration analyte, highly crowded, without having motion for few frames) are shown in Figure 7.4, Figure 7.5, Figure 7.6 and Figure 7.7. Tracking result for PETS'09 datasets is shown in Appendix 4

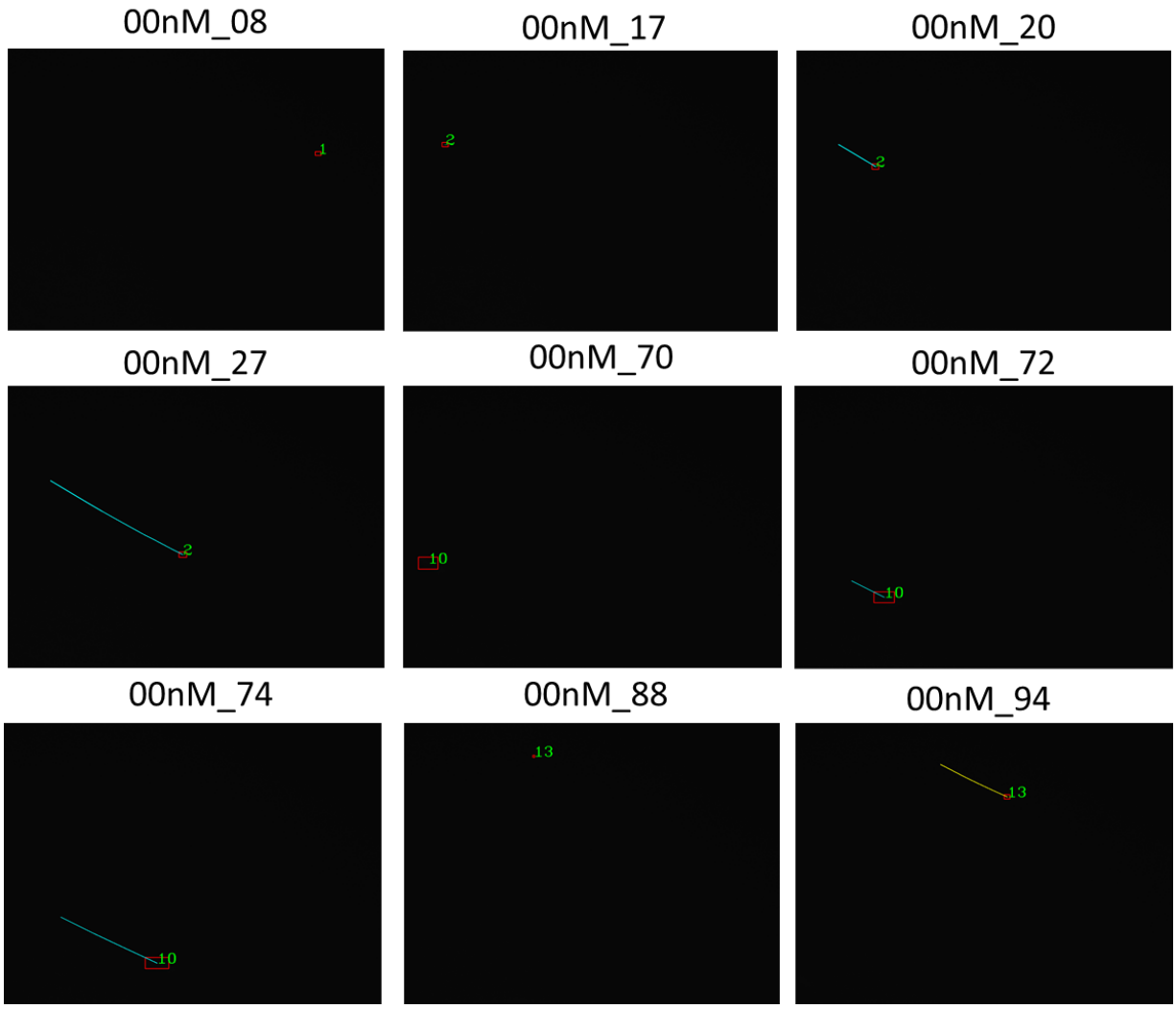


Figure 7.4: Micro particle tracking with analyte concentration of 0 Nm for DNA hybridization experiment. Here traces are showing the travelled paths.

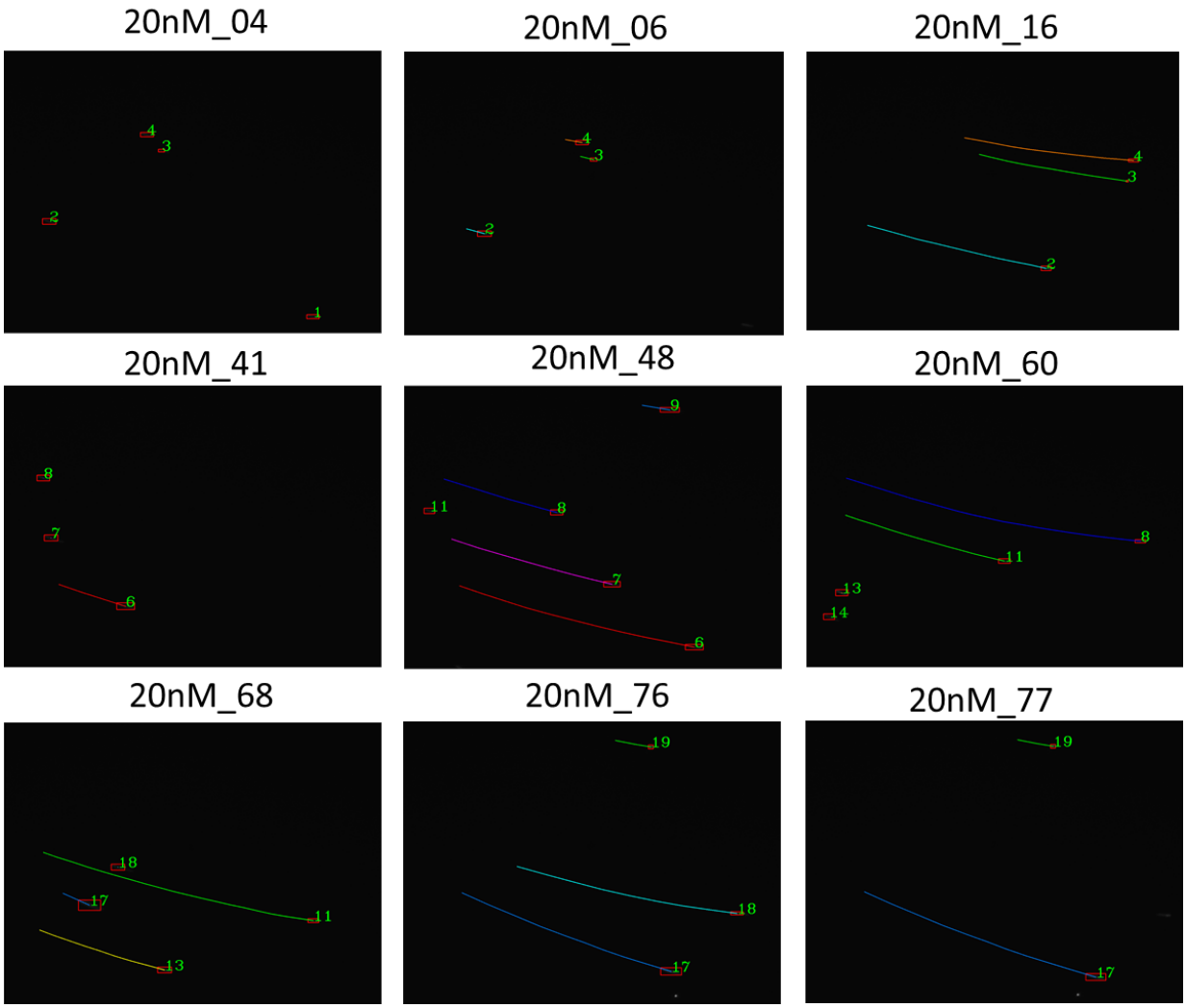


Figure 7.5: Micro particle tracking with analyte concentration of 20 nM for DNA hybridization experiment. Here traces are showing the travelled paths.

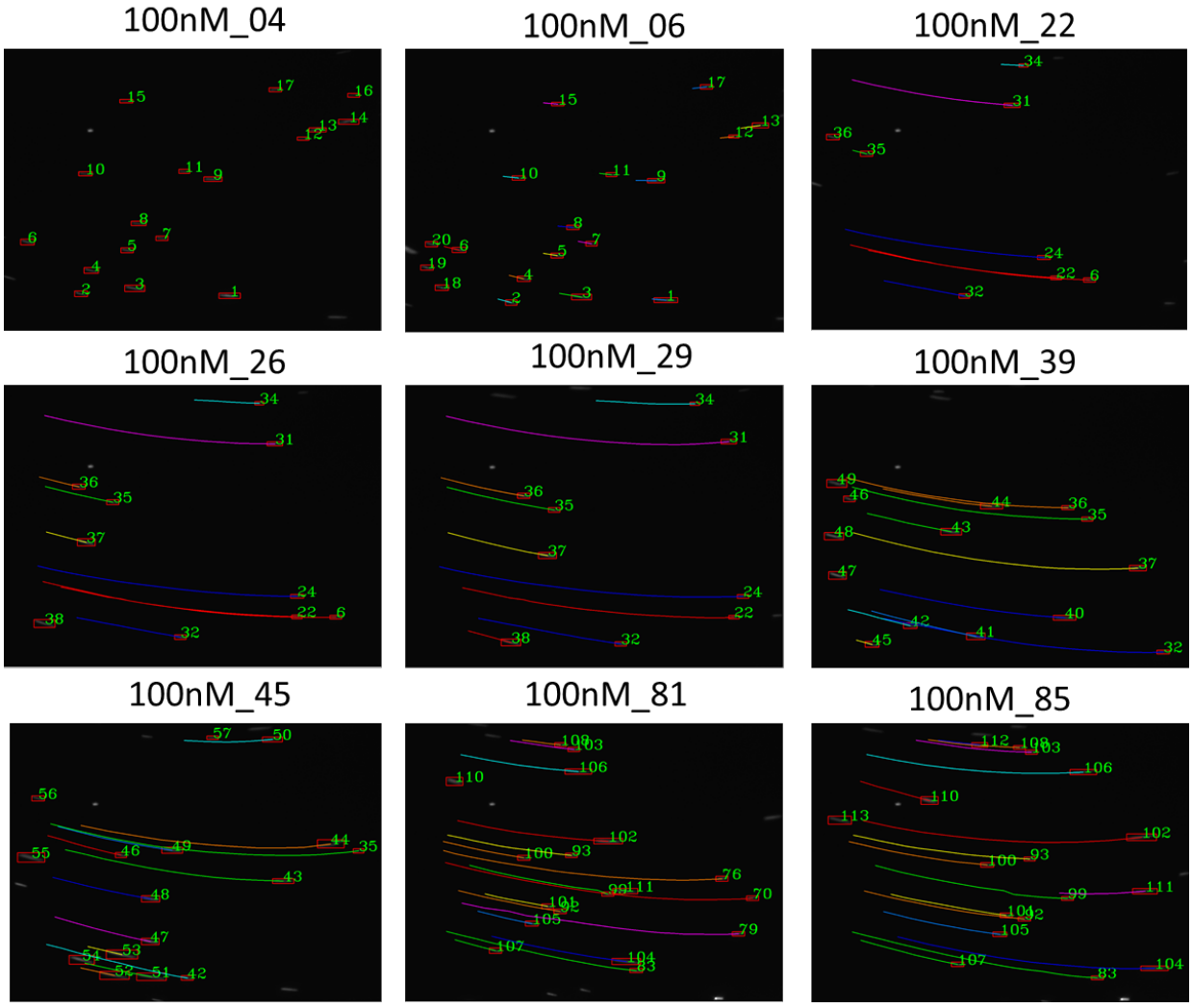


Figure 7.6: Micro particle tracking with analyte concentration of 100 nM for DNA hybridization experiment. Here traces are showing the travelled paths.

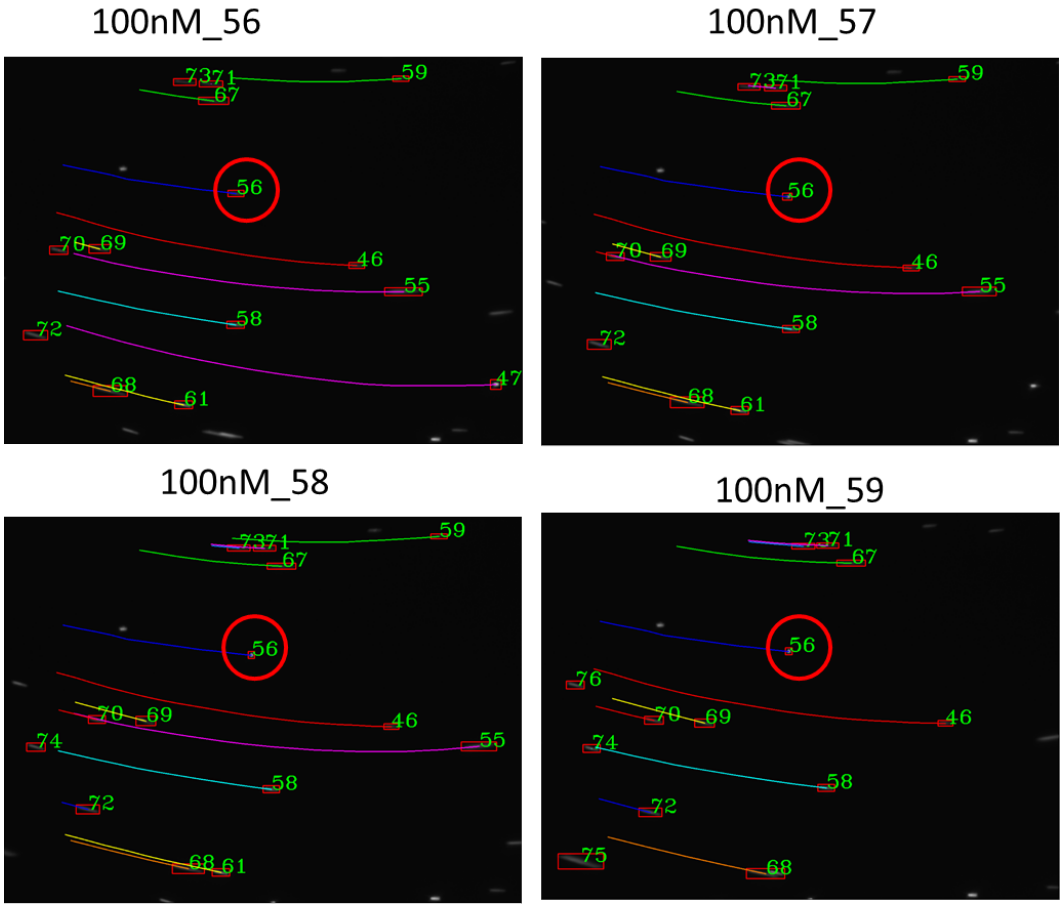


Figure 7.7: Micro particle tracking with analyte concentration of 100 nM for DNA hybridization experiment. Here traces are showing the travelled paths. The object within Red Circle did not move for four frames and kept the same ID 56.

In Figure 7.4 the tracking was performed on an input data sequence from experimental results with 0 nM analyte concentration. In this case the objects have minimum difference from the background. Through use of the background subtraction technique the tracker has knowledge about the location of the objects, but for matching an object from its previous location to the current one required HOG features to match. But in these cases HOG feature matching based Meanshift did not worked well without Kalman tracking. But still the tracker was able to track objects throughout the sequence.

Figure 7.5 is showing the tracking for an input sequence from an experiment with 20 nM concentration. Here the visibility of the objects is higher than the objects from image sequence with 0 nM concentration and the proposed method tracked

all the object without changing any IDs.

In Figure 7.6 the number of objects are much higher than that of previous figures (Figure 7.4 and Figure 7.5). Objects are moving with a very small gap between each other. But the proposed tracker kept track of each object without having many false positive and negative detections.

In Figure 7.7, the object with ID 56 did not move for four frames and it can be seen that its ID did not change for the non-moving positions.

The proposed method was found to be more accurate tracking objects with high concentration (till 20 nM) of analyte, because for low concentration analyte the objects are not detectable in many cases. However, the proposed method was successful in tracking objects in low concentration analytes. Table 7.1 shows the number of total micro-particles and correctly tracked microparticles for various concentrations (00 nM, 20 nM, 40 nM, 60 nM, 80 nM and 100 nM) of analyte.

Concentration (nM)	**Tracked Particles Correctly**	**Total number of particles**	**Accuracy** (%)
0	56	72	77.78
20	44	51	86.27
40	73	81	90.12
60	25	28	89.29
80	154	169	91.12
100	115	127	90.91

Table 7.1: Performance measurement for tracking microparticles.

From Table 7.1, it can be seen that the accuracy of tracking micro-particles for the proposed method is around 90%. Moreover, the number of detected particles are also higher than manual method (where the maximum number is around 15 to 40). Measurement based on more particles should provide more confidence in the results.

7.3 Summary

In this chapter details of the tracker implementation has been presented and showed its tracking performance. For all the comparisons performed the same background modelling and background subtraction pre-processing was used. Also for any parameter like that for the Kalman filter, the same value is used in all cases to facilitate comparisons. The accuracy of the approach to track micro-particle was more than 90%, which can be considered as a good accuracy for tracking multiple objects having very similar features (path of motion and colour feature) and for objects which are overcrowded.

There was a problem with detecting the same object with same ID after occlusion which proved impossible to solve. Unlike humans, vehicles *etc.* it was not possible to define any constant feature, though many known features like shape, HOG, SIFT, corner detection were tried. The main reason behind being unable to find the same object with the same ID was- when an object starts to move after occlusion is that an objects features do not persist.

Despite the problems just mentioned the tracker was more than adequate for the majority of particles which allowed their intensities to be measured, as required by the experiments. Details of this data analysis and the results obtained are discussed in the following chapter.

Chapter 8

DATA ANALYSIS RESULT

In this chapter of thesis, results from using the new approach for making the data analysis process from magnetophoresis Lab On a Chip results will be presented. Though lots of different experimental data from various magnetophoresis experiments were used for checking the output of the new approach, it was not possible to put all the test results here. So to show the performance of the approach, data from a specific experiment which was published by Prof. Nicole Pamme's research group [17] will be presented. Using these image sequences, the new approach is used to automate the production of a calibration curve with error bars. This is compared with the results from manually calculated values. Also the limit of detection indicated by the two approaches is compared.

The published data analysis system is totally manual and hence is both slow and user dependent, which tends to lead to a non-reproducible system. Moreover, as the process is not automatic and user dependent, so the effective limit of detection (LOD) may also vary according to the experimenter. So the aim of this section of work was to develop a system which is automatic, faster and more reproducible.

8.1 Experimental Set-up

The experiment of interest involves DNA hybridisation, which is a widely known and used analytical technique in biochemical research, clinical diagnostics and forensic

science for detecting the presence of specific nucleotide sequences within a sample. The conventional method of DNA hybridisation is quite time consuming and labour intensive, as the current method involves long incubation times, several washing steps and long data processing steps [17]. The whole experiment can be actually divided into two parts- On chip and off chip experiments. The main concern for this section and thesis is the off-chip experiment, also known as the result analysis. In this result analysis section Prof. Nicole Pamme and her group used two well-known established software- Image Pro and ImageJ.

For this experiment the microfluidic chip used had five inlets and two outlets (designed by Prof. Nicole Pamme, widely known as NP50), which had a reaction chamber of $3mm \times 8mm$ and supported by 10 diamond shaped micro pillars [17]. This microfluidic chip was fabricated from glass and using photolithography and wet-etching techniques [157]. Through these inlets solutions of Magnetic particles functionalised with streptavidin (Dynabeads M270, 2.8μm diameter, obtained from Invitrogen in Phosphate Buffered Saline (PBS) buffer), Concentrated Saline-Sodium Citrate (SSC) buffer (20×), Hybridisation buffer (44.115 gL^{-1} sodium citrate , 87.660 gL^{-1} NaCl), PicoGreen and working solution were injected at a velocity of 210 μms^{-1} into the reaction chamber and they were withdrawn at a rate of 50 μLh^{-1}. For generating the magnetic field, a NdFeB magnet ($4mm \times 4mm \times 5mm$, grade N48H, Magnet Sales, UK) was placed on the top of the chip and $1mm$ from the reaction chamber side. The chip was then placed on the stage of an inverted fluorescent microscope (TE2000-U, Nikon, Japan) which is equipped with a Charge-Coupled Device (CCD) camera (Retiga-EXL, QImaging, UK). During the fluorescence intensity measurements, the camera's exposure time was chosen as 300 ms for recoding the exit of the chamber which is close to the magnet. The recording process was done using ImagePro 6.2 software and the grey scale intensity of the particle was measured and analysed using ImageJ (http://rsbweb.nih.gov/ij/). The whole experiment was done at an ambient temperature of 24^oC. The total experimental setup can be shown using the Figure 8.1:

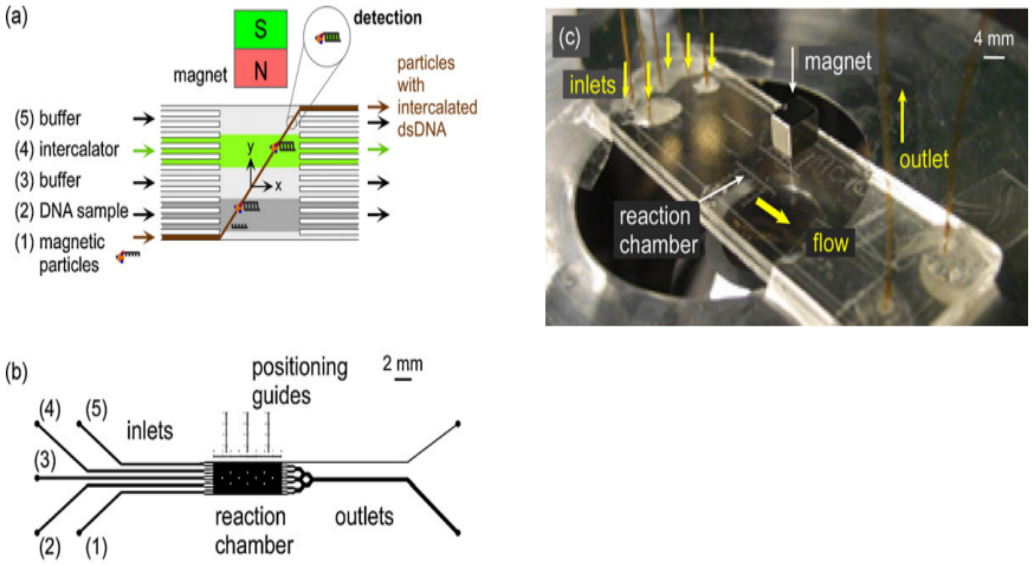


Figure 8.1: (a) Continuous flow DNA hybridisation platform. Magnetic particles with immobilised capture DNA are deflected across a reaction chamber using magnetic field from external magnet and thus several reagent and buffer streams. DNA hybridisation, intercalation as well as washing steps are thus performed on the surface of the particles in continuous flow in an automated and rapid fashion. (b) CAD schematic of the microfluidic chip design featuring five inlets, a 3 mm wide and 8 mm long reaction chamber and two outlets. (c) Photograph of the chip setup, showing the fabricated glass device interfaced with tubing and the NdFeB magnet (N48H) [17].

8.2 Calibration Curve

The 'Calibration curve' is a well-known and established method in analytical chemistry and biology for determining the concentration of a given substance (analyte)in an unknown sample by comparing it with a standard set of known concentration samples [158]. A calibration curve (graph) is actually a plot of experimental means (average) from the response of an analytical signal which changes with the concentration of the unknown substance and standard concentration. In this graph the concentrations of the solutions are plotted on X-axis and observable variable (like solution's absorbance, light intensity, voltage, current *etc.*) are plotted on Y-axis. Then by putting all the observable values (acquired or processed data) with respect to the concentration of several solutions on the graph a calibration curve can be obtained. Then by placing the observable value from the unknown concentration on the calibration curve the concentration of that sample can be determined.

Light intensity as an observable value from a chemical solution is a common

choice, as according to Beer's law (also known as Beer-Lambert Law) chemical solutions absorbs or emit different amount of light depending on their concentrations and the relation between the concentration and light fluorescence is linear [159]. Moreover, it is quite easy to measure the light fluorescence of a solution using a normal video camera with suitable software operation.

Mathematically the relationship between fluorescence and concentration is linear, but it is not always achievable from experimentally determined variables. So in most cases, the trend line for the calibration curve is required to be drawn in such a way that it passes a maximum number of the data points.

Calibration curves provide several advantages for many analytical techniques and experiments. Firstly using calibration curve it is possible to calculate the uncertainty of the concentration (using the statistics of the least squares line fit to the data) [161]. Secondly, the calibration curve provides result that depend on an actual relationship.

8.3 Mathematical Expression of calibration curve

The data points (observed with respect to the concentration) plotted on the graph for a calibration curve can be fitted to a straight line by using linear regression analysis. After the linear regression analysis, the calibration curve model can be described as $y = mx + c$, where 'y' is the observable response, 'm' represents the sensitivity of the curve calculation and c is a constant which describes the background. Using this equation, the analyte concentration (x) of an unknown samples may be calculated.

8.4 Data Analysis: Calibration curve

Data analysis or row data analysis is considered as one of the most significant parts of analytical experiments. Among several data analysis methods, the most commonly used methods are developing calibration curves, regression lines and limits of detection. Here for comparing the two different ways to analyse the data, for the same

experiment seven different solutions were made of 0 nM (blank), 20 nM, 40 nM, 60 nM, 80 nM, 100 nM concentration. Firstly, the widely used software ImageJ (Java based) was used for getting the intensities of the objects for the calibration curve, regression line and limit of detection (LOD). Similar analysis was done using the newly developed automatic detection software as well. After achieving the results, the success or accuracy of the analysis was compared in the following section.

8.5 Data analysis using manual method

Here the user is required to find the objects manually one by one. In this case for differentiating between particles and noise, user's experience is the most significant factor. Then for the selected particle the intensity measurement operation can be done in several ways one of them is drawing a rectangle around the detected particle. This rectangle indicates that the user is interested in the information inside the rectangular area. On the shown image in Figure 8.2 the identified object and a drawn rectangle is shown. After drawing the rectangle, the user needs to press M to measure the different intensity level (Maximum, Minimum *etc.*) within the rectangle. The measured value will be shown into a separate window as shown in Figure 8.2:

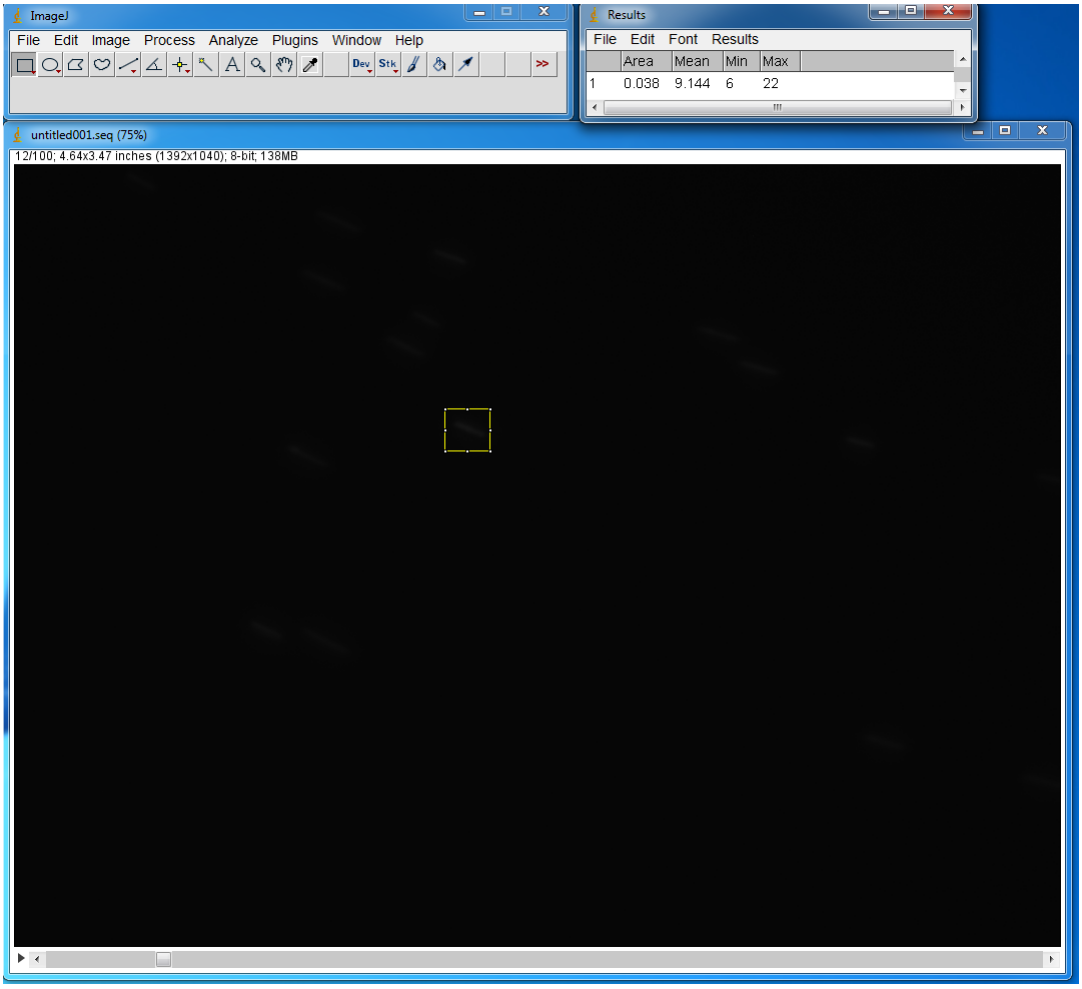


Figure 8.2: ImageJ particle intensity measurement step.

This step is required to be repeated until most of the particles' intensity (at least 20 particles) are measured.

After all the particles from one 'Tiff' or 'SEQ' file have been measured, the user needs to save the values in an Excel format. After saving the measured intensity for the detected particles (for one concentration of the experiment), the user needs to repeat the process for the other image sequences achieved from experiments using different concentration.

After the data has been saved in Excel. It is checked for any suspicious 'Peak Values'. If any are found, the data needs to be checked again, otherwise the Mean (Average) and Standard Deviation for each experimental result are gathered for plotting the calibration curve.

Here as the rectangle, area and the particles are chosen manually, so very often the results are not so accurate. This process is required to be done for every experiment. Each experimental result needs to be processed separately for finding the mean and standard deviation from the raw data, which needs to be compared later with other results. All these processes, if the experimenter gets a success first time, every time, takes more than 2 hours.

8.6 Current Automatic Software

To overcome the drawbacks of the manual data analysis an automatic detection software was developed. This automatic software was developed using C++ as development language. Compared with the analysis steps done using imageJ and Microsoft excel, this software only requires two steps to achieve the results, step one is to provide the path of the images to be analysed and second is to press the button to start "Analyse". The whole process takes about 5 minutes (considering 6 image sequences to analyse). This whole process is totally automatic and fully reproducible, also this process does not require any experience from the experimenter. It is also an unbiased process, as the area selection for measuring the maximum and minimum intensity is covered by the minimum boundary rectangle of the object. The user interface for the software is shown using Figure 8.3:

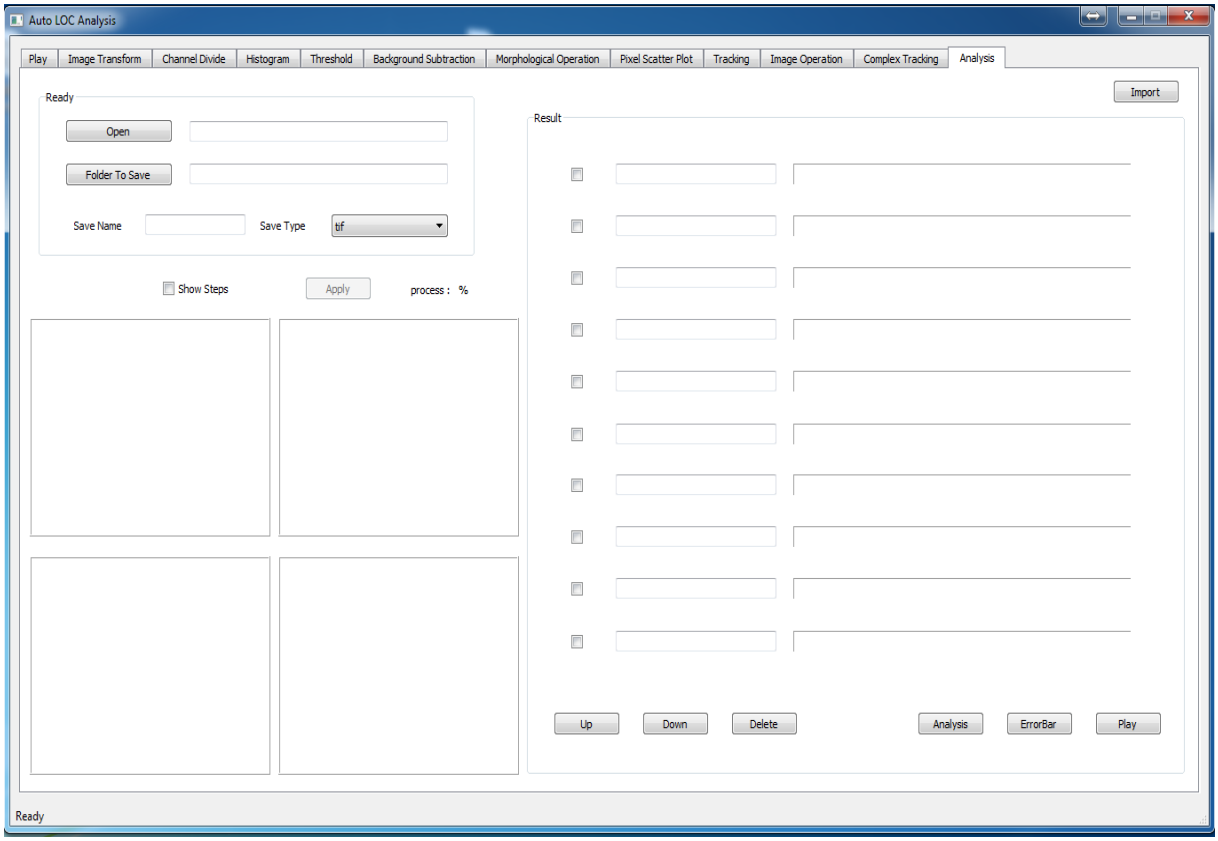


Figure 8.3: Automatic LOC Data Analysis software user interface.

8.7 Result analysis process using automatic software

During the manual process of calculating the mean and standard deviation for any experiment, the user selects the data points (micro-particles) manually to measure the intensity. After measuring the intensity of a few particles (around 15 to 50 number), the mean (average) is calculated and then standard deviation. In this manual process, the user selects only those micro-particles which are moving and representing the analyte. In this way, the user is always biased about getting better results.

For the automatic data analysing system, the results produced were reproducible and non-biased to any condition. However, during the automatic analysing of the

experiments it was found that many objects were detected which do not represent the analyte concentration, either due to false detection or because the object (micro-particle) itself provide false information (*e.g.* stuck particles). So, after analysing such situations the following observations were found:

1. Micro-particles which are not moving produce much higher intensity, which is not proportional to the analyte concentration.

2. Any stopped micro-particle (previously moving) produce higher intensity in the frame which is one frame before it was stopped

3. Any stopped micro-particle produce higher intensity in the frame in which it starts to move.

After considering all these situations, it was found that when the experiments were done using higher concentration of analyte and the number of micro-particles present (detected) within the experiment is high, these three conditions do not make much effect on the result of mean (average) and standard deviation. However, when the analyte concentration is low or the number of detected particles is small, then all these observations have an effect. By analysing the experimental results, it was found that when the analyte has less concentration, the number of detected micro-particle is also less. In this situation, it was required to find an effective way of automatic outlier removal for analysing the data to produce a result which represents the real scenario.

Considering all these factors, an approach for relating the position and velocity of each micro-particle with its intensity was developed as shown in Figure 8.4 and Figure 8.5.

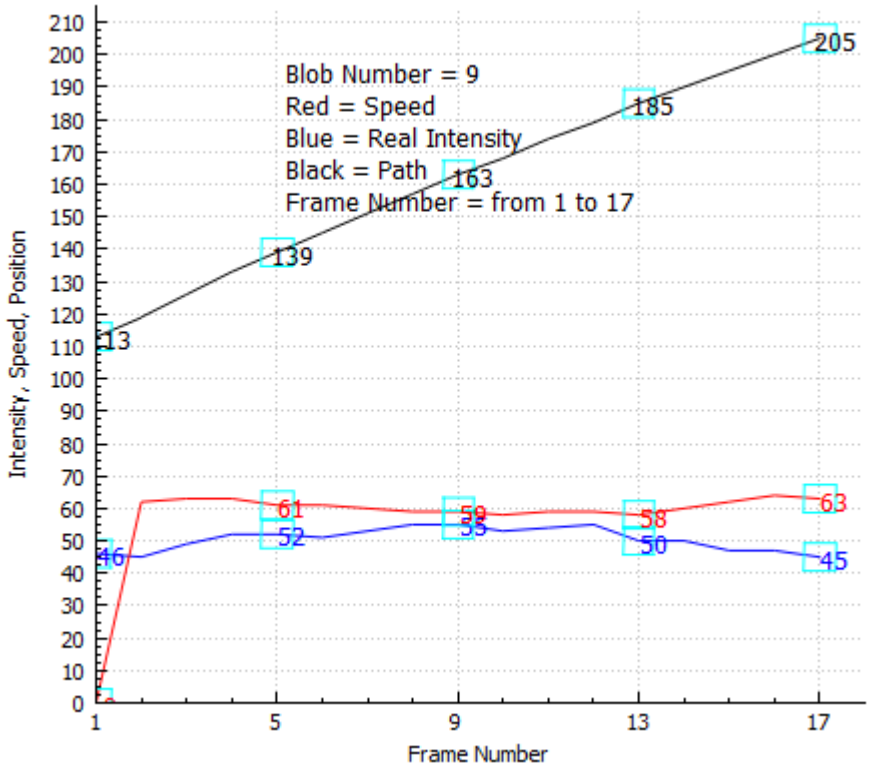


Figure 8.4: Relation between the objects position, velocity and intensity for a detected object with ID 9, from DNA Hybridization experimental result with 80 nM concentration.

In Figure 8.4, the micro-particle with ID 9 travelled the area of capture with almost same velocity without stopping (red line). Also the intensity of the micro-particle is almost the same (blue line) in every frame.

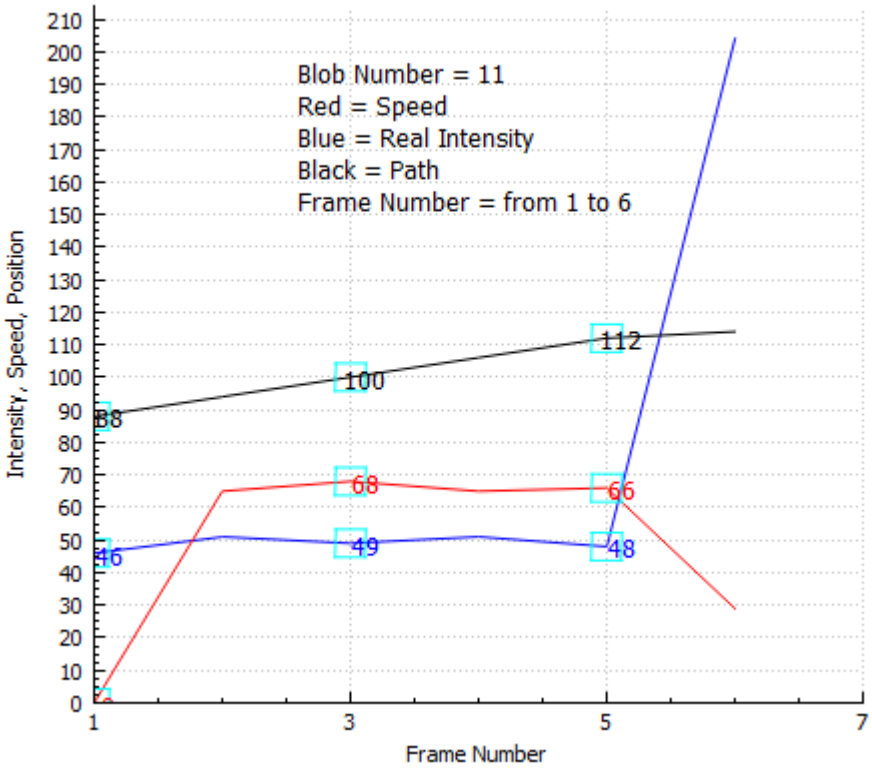


Figure 8.5: Relation between the objects position, velocity and intensity for a detected object with ID 11, from DNA Hybridization experimental result with 80nM concentration.

In Figure 8.5, the velocity of the micro-particle with ID '11' was not similar for all the frames. From frame number 5 to 6 the velocity was much less than its velocity in previous frames. This change in velocity provides an indication that the object was either stopped or about to stop. Also the measured intensity (blue line) when the particle got stopped became very high.

These two figures (Figure 8.4 and Figure 8.5) justify the observations described in the beginning of the section.

After analysing lots of experimental image sequences and considering several statistical approaches (mean, median, Gaussian distribution etc.), the best conditions for outlier are:

1. Intensity of an object was not considered while it was not moving.

2. ' Intensity of an object was ignored from the frame before it gets stopped.

3. ‘ Intensity of an object was not considered from the first frame after it started to move from its non-moving condition.

4. For each object after getting intensity values fulfilled conditions 1, 2 and 3, all the values were arranged in an incremental order and only the median value of the intensity was considered as its representing intensity.

5. Using conditions 1, 2, 3 and 4, for each object only one value was obtained as intensity, these values were used for getting the mean and standard deviation for the input frame.

6. in order to make all the assumptions from 1 to 5 more robust against any false detection, a ratio based intensity measurement for the object was used. In this ratio based approach, the direction of movement for an object was used as its central line. The length of the objects central line was required to be a minimum of 1.5 times bigger than its vertical central line, for the object to be used for measurement purposes.

Using all the steps discussed above the final result for calculating error bars and calibration curves to find the concentration of the unknown analyte was achieved.

8.8 Result comparison

For comparing the results achieved from both the manual way and the ’automatic software’, both the calibration curves for the same input sequence were put together and shown in Figure 8.6. The result for manual calculation was done by Dr Martin Vojtisek during his PhD work and it was represented in Figure 8.6 without any modification.

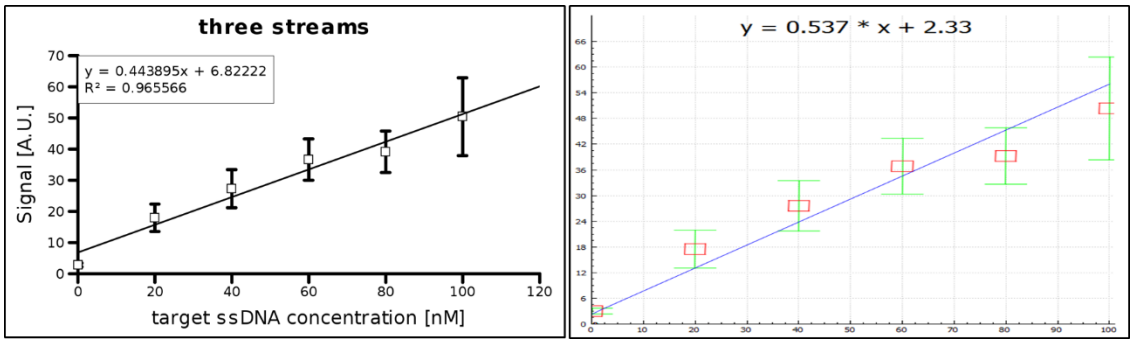


Figure 8.6: Calibration curve comparison [56].

From Figure 8.6, it can be clearly seen that the error bars have a similar range to those from the implementation of automatic software. Here the best-fitting straight line passing through the mean points was expressed as $y = mx + c$ for both the manual and automated methods. However, the values of 'm' and 'c' for both methods are not the same. The best fitting line equation depends on the values that are used for getting the line, for the manual and automated methods the values of mean intensities were not exactly same. These differences between the values produced different values for 'm' and 'c'. However, as it is not so easy to compare the values of mean and standard deviation from the Figure 8.6, so in Figure 8.7 and Table 8.1 these values are presented again for clarity.

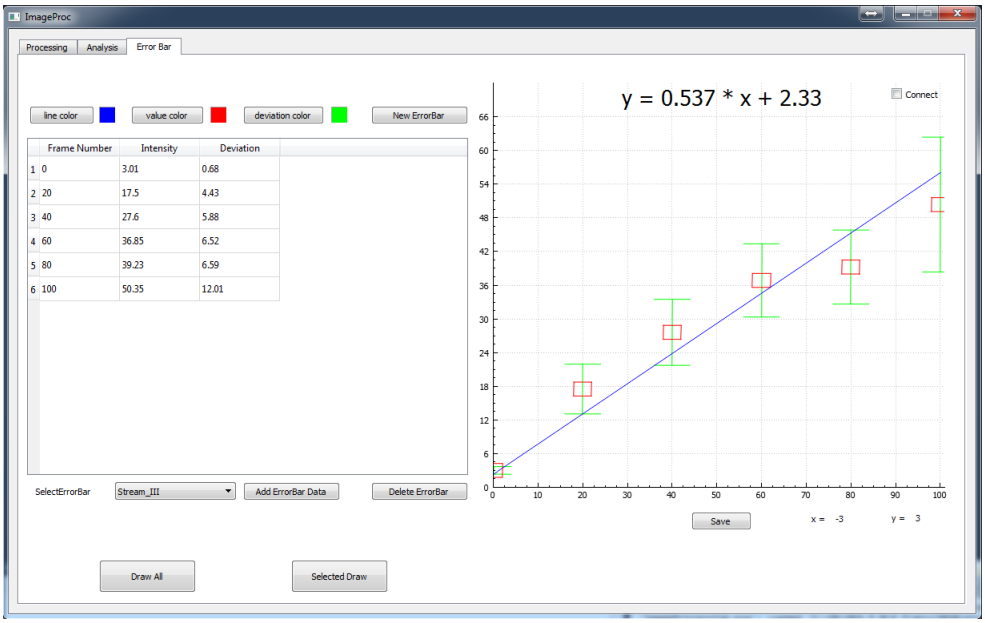


Figure 8.7: Data used for error bar calculation using automatic method.

Concentration (nM)	Mean	Standard Deviation	Counted Particle
0	2.81	0.62	9
20	17.93	4.45	15
40	27.26	6.11	15
60	36.61	6.65	19
80	39.13	6.67	41
100	50.37	12.45	31

Table 8.1: Manually calculated values, Done by Dr Martin Vojtisek.

Comparing Figure 8.7 and Table 8.1, it can be found that both the values of means and standard deviations are almost similar in both cases. But the main differences are this result is reproducible, faster to achieve and no need any special knowledge for selecting an object.

8.9 Limit of detection (LOD)

It is quite impossible to find any instrumental measurement which is not associated with some error, even for baseline (background or black) measurement some signal can be obtained when no analyte is present. So it is important to measure how large or big a signal needs to be distinguishable from background noise of the instrument. Limit of detection (also known as detection limit) is the expression of this lowest quantity of substance that can be differentiable from a blank value within a predefined confidence limit (normally 1%) [160, 161]. There are several types of "detection limits" that are commonly used, including Instrument Detection Limit (IDL), Method Detection Limit (MDL), Practical Quantification Limit (PQL) and Limit of Quantification (LOQ). Moreover, for the same terminology, there can be differences in LOD according to the definition of LOD, noise contributes to the measurement and calibration. Mostly, for the limit of detection (LOD) the mean of the blank, the standard deviation of the blank (sigma (σ)) and confidence factor are required. Here the accuracy and efficiency of the model used for LOD to predict the unknown concentration from the raw analytical signal also affects the accuracy of detection [162].

For most analytical experiments a signal can be found even when analysing with a blank (matrix without analyte). This signal is referred to as the noise level. Generally, to be detected a signal needs to be at least three times greater than the background noise. More accurately, it can be said that for measurements with a normally distributed probability density function for the blank, LOD is defined as a summation of the blank signal (signal for the background) with 3 $\times$ standard deviation (sigma, (σ)) of the blank (background noise). This mathematical relationship for LOD can be expressed as Equation (8.1):

$$y_{lod} = y_{blank} + 3 \times \sigma_{blank} \tag{8.1}$$

As a summary, LOD is the measurement which represents a level, below which it cannot be confidently said that the analyte is actually present or not. As there

is signal available from blank, so it can be said clearly that there is nothing called pure zero concentration.

So for determining the effectiveness detection using automatic method, limit of detection (LOD) is one of the best techniques. So for both the case (manual method and automatic software), LOD was calculated. The calculation steps for LOD for both manual and automatic software are shown in following section:

Using automatic software, the fluorescence intensity for image sequence input with analyte concentration of $0nM$ is $3.01 \pm 0.68 a.u$, so $LOD_A = 3.01 + 3 \times 0.68 = 5.05$.

Using manual method, the fluorescence intensity for the same input sequence is $= 2.81 \pm 0.62 a.u$, so $LOD_M = 2.81 + 3 \times 0.62 = 4.66$.

Here limit of detection is higher for automatic detection, but lower LOD indicates better confidence level. The automatic method has 7.72 % less detection capability then that of using manual method. However, this difference is not particularly significant. Though the LOD performance is better in manual method, but it was done in such a way that it cannot be tested again, as the selection criteria of the particles cannot be reproducible.

For determining the lowest concentration that can be calculated using our proposed method, we can consider the Figure 8.7 again, where the calibration curve is shown as Equation (8.2):

$$y = 0.537 * x + 2.33 \tag{8.2}$$

Now if $y = 5.05$, then from Equation (8.2) we get

$$x = \frac{5.05 - 2.33}{0.537} = 5.065 \tag{8.3}$$

Therefore, from Equation (8.3), it can be said that 5.065 mug/mL is the lowest concentration of analyte that can be measured using the automatic method.

8.10 Summary

The difference between automatic and manual detection is not so high for both the performance and accuracy, this is to be expected given that the manual results were obtained by an experienced operator. The main differences were reproducibility and time requirement. Using the automatic detection method, the same calculations were done three times in three different days and every time the results were exactly same. But for the manual detection reproducibility was not only difficult, but also quite impossible. For automatic detection method limit of detection was slightly higher than the manual detection technique. However, the difference is not particularly significant.

Chapter 9

FINAL CONCLUSION AND FUTURE WORK

9.1 Discussion and Conclusion

This thesis contains mainly two parts, the major part has dealt with automatic object detection, tracking and intensity measurement of micro particles movement in Lab-on-Chip magnetophoretic analytical systems, the rest has used the Comsol simulation tool to simulate magnetophoresis experiments with respect to the influence of the magnetic field.

Lab-On-Chip magnetophoretic analytical system allows integration of a microfluidic device with sensors and actuators, which results in complex system automatically performing various processes. Over the last five decades, research about microfluidics has become very popular with a wide range of applications in the fields of medicine, environmental monitoring and chemical analysis [1, 2]. However, the automatic result detection and analysis are still a challenging work in this field with less development than manual experimental procedure.

The work of this thesis is based on analysing the results obtained from magnetophoresis LOC experiments done by Prof Pamme and her research group [1, 13, 15]. The results contain a series of image frames which records micro particles move-

ments. Aiming for automatic micro particles detection, tracking and analysis, a suitable and successful principle has been developed in this thesis, which includes three main steps: object detection, object tracking and result analysis.

Object detection is the key procedure for the entire work. Without robust and accurate object detection technique, it is almost impossible for the next step of tracking the micro-particles, including automatic analysis. There are many available methods which have been developed for traffic monitoring, human face recognition and human behaviour identification. However, none of them were useful in this case for detecting micro-particles. One of the main problems for these techniques is that they were not able to detect very low intensity objects. Also most of the available techniques required at least 10 frames (except ViBe) for training. But it was not possible to use 10 frames from experimental input sequences, because in most cases there were only around 100 frames to analyse. The new method presented here only used 3 frames for starting the detection. It was also able to detect most of the objects separately, even when they were very close to each other. Which means, considering the features of micro particles and image quality obtained from Lab-on-Chips experiments, the proposed method provided robustness compared to other methods. The proposed object detection technique works in four steps: pre-processing, background modelling, extracting foreground object features and object detection (discussed in Chapter 5). Out of all these four steps background modelling and object detection were the most important step and was performed via combining three frame difference and improved GMM (by introducing $T-AMBP$ with GMM). The results from our approach was compared with GMM, KDE, Vibe, adaptive BGL was the result was better than these techniques (showed in Chapter 5).

Following object detection, an object tracking algorithm has been developed by integrating Hybrid Meanshift (combining Meanshift, template matching (HOG matching) and optical flow) and Kalman filter (Chapter 7). Lucas-Kanade optical flow was used to restrict the search window in Meanshift to improve tracking accuracy. HOG was used to represent each object and find the object in next time

frame.

The objects for tracking are micro particles in LOC experiments. They are very similar to the background in colour and also there is not many distinguishable features to differentiate between them. These characteristics make it challenging to separate them from the background and distinguish between individual objects for tracking. Though the developed method could achieve most object detection and tracking tasks, it still has certain levels of limitation. They are- I) when more than one object follows the same path and occlusion occurs between them, after occlusion the tracker can not find the same object with same ID, II) when the number of objects within a small area become too high and III) when the object's intensity becomes too low to be identified. One of the reasons behind these problems is having no feature which is constant over the whole tracking process especially after occlusion. Also very low contrast of the objects limited the tracking in some cases.

LOC experiments with low concentration of analyte produce very low signal, which requires pre-processing for making the signal (intensity) stronger. This pre-processing was only done when necessary. Because it consumes time, which makes the analysis process slower. Data from a published LOC experiment was analysed fully automatically to produce a calibration curve in the same format as the original work. Comparison of the results indicated that the automated procedure provided results very similar to those obtained manually by an experienced and competent operator. The manual procedure took at least 2 hours whereas the automated process required about 5 minutes for an operator to perform. The automated process does not require significant knowledge on the part of the operator and is very reproducible. The manual approach is more variable as it depends on operator judgement.

A common problem effecting magnetophoresis experiments, which are the subject of the automated analysis in this work, is designing the applied magnetic field. It was found that magnetic field gradient from the applied magnetic field was not homogeneous as required. For achieving such homogeneous magnetic field gradient, it was not possible to use single magnet or a conventional shaped magnet. For this reason, either custom shaped magnet(s) or more than one permanent magnets were

required. However, use of custom shaped magnets are not practical. So in Chapter three a two permanent magnet solution with one placed in the upper and the other in the lower position of the chip was investigated. This set-up of simulation provided nearly homogeneous magnetic field gradient across the reaction chamber.

9.2 Future Work

To improve and extend the proposed methods for background subtraction, tracking and automatic LOC based measurement future work the following future work can be considered:

1. An important and general problem existing in the object tracking field is the integration of foreground object contextual information during the tracking process. For example, in the Lab-on-Chip micro-particle tracking application, introducing the object recognition step during the tracking procedure could identify and separate different types of particles if there are more than one type of particle in the reaction chamber.

2. In this proposed method, the K number of histogram was chosen to represent the background model and subsequently threshold T_B to determine the pixel as foreground or background was closely related with K number. Therefore, further optimization of the K number threshold T_B will be critical and crucial for generating much more accurate results.

3. Further investigation can be done concerning the object feature operator. Instead of using pixel intensity, a texture feature for describing background model and foreground object was applied due to its properties of robustness, adaptiveness and effectiveness against noise. In the future, other more complex object feature describes like SIFT or region covariance could be combined together for more robust object recognition and accurate tracking.

4. One of the assumed conditions for the proposed algorithm is that the camera is fixed with a certain distance above the chamber. The problems occurred

for this situation are that first, the video taken was only for a part of reaction area not the entire reaction chamber, second the method will not work if the camera is moved. In the future, an improved algorithm for tracking objects in a video taken by a moving camera or multiple cameras which cover the entire reaction area could be considered.

5. Design of a Lab-on-Chip device with one camera or two which could record the video across the entire reaction chamber.

6. Investigate the use of magnetic modelling and simulation to help predict the path of the particles to improve tracking. This will integrate the two theories of this thesis. Unfortunately, there was insufficient time to pursue this during this work.

More generally and more importantly, the overall aim of future work is to develop a complete fully automated measurement system. This could perform all the processes of the physical experiment as well as the data analysis and provide to potential for real world analysis for example in medical point of care situations.

Bibliography

[1] N. Pamme, "Continuous flow separations in microfluidic devices," *Lab on a Chip*, vol. 7, no. 12, pp. 1644–1659, 2007.

[2] F. Van De Pol and J. Branebjerg, "Micro liquid-handling devices-a review," in *Micro System Technologies 90*, pp. 799–805, Springer, 1990.

[3] G. M. Whitesides, "The origins and the future of microfluidics," *Nature*, vol. 442, no. 7101, pp. 368–373, 2006.

[4] S. Bau, N. Schracke, M. Kränzle, H. Wu, P. F. Stähler, J. D. Hoheisel, M. Beier, and D. Summerer, "Targeted next-generation sequencing by specific capture of multiple genomic loci using low-volume microfluidic dna arrays," *Analytical and bioanalytical chemistry*, vol. 393, no. 1, pp. 171–175, 2009.

[5] P. Gravesen, J. Branebjerg, and O. S. Jensen, "Microfluidics-a review," *Journal of Micromechanics and Microengineering*, vol. 3, no. 4, p. 168, 1993.

[6] F. K. Balagaddé, L. You, C. L. Hansen, F. H. Arnold, and S. R. Quake, "Long-term monitoring of bacteria undergoing programmed population control in a microchemostat," *Science*, vol. 309, no. 5731, pp. 137–140, 2005.

[7] D. Wlodkowic and J. M. Cooper, "Tumors on chips: oncology meets microfluidics," *Current opinion in chemical biology*, vol. 14, no. 5, pp. 556–567, 2010.

[8] F. E. Regnier, B. He, S. Lin, and J. Busse, "Chromatography and electrophoresis on chips: critical elements of future integrated, microfluidic analytical sys-

tems for life science," *Trends in biotechnology*, vol. 17, no. 3, pp. 101–106, 1999.

[9] A. Manz, D. J. Harrison, E. M. Verpoorte, J. C. Fettinger, A. Paulus, H. Lüdi, and H. M. Widmer, "Planar chips technology for miniaturization and integration of separation techniques into monitoring systems: capillary electrophoresis on a chip," *Journal of Chromatography A*, vol. 593, no. 1, pp. 253–258, 1992.

[10] V. Dolník, S. Liu, S. Jovanovich, *et al.*, "Capillary electrophoresis on microchip," *Electrophoresis*, vol. 21, no. 1, pp. 41–54, 2000.

[11] C.-M. Ho and Y.-C. Tai, "Micro-electro-mechanical-systems (mems) and fluid flows," *Annual Review of Fluid Mechanics*, vol. 30, no. 1, pp. 579–612, 1998.

[12] H. A. Stone and S. Kim, "Microfluidics: basic issues, applications, and challenges," *AIChE Journal*, vol. 47, no. 6, pp. 1250–1254, 2001.

[13] N. Pamme, "Magnetism and microfluidics," *Lab on a Chip*, vol. 6, no. 1, pp. 24–38, 2006.

[14] N. Pamme, J. C. Eijkel, and A. Manz, "On-chip free-flow magnetophoresis: Separation and detection of mixtures of magnetic particles in continuous flow," *Journal of Magnetism and Magnetic Materials*, vol. 307, no. 2, pp. 237–244, 2006.

[15] N. Pamme and A. Manz, "On-chip free-flow magnetophoresis: continuous flow separation of magnetic particles and agglomerates," *Analytical chemistry*, vol. 76, no. 24, pp. 7250–7256, 2004.

[16] S. A. Peyman, A. Iles, and N. Pamme, "Rapid on-chip multi-step (bio) chemical procedures in continuous flow–manoeuvring particles through co-laminar reagent streams," *Chemical Communications*, no. 10, pp. 1220–1222, 2008.

[17] M. Vojtíšek, A. Iles, and N. Pamme, "Rapid, multistep on-chip dna hybridisation in continuous flow on magnetic particles," *Biosensors and Bioelectronics*, vol. 25, no. 9, pp. 2172–2176, 2010.

[18] S. Bronzeau and N. Pamme, "Simultaneous bioassays in a microfluidic channel on plugs of different magnetic particles," *Analytica chimica acta*, vol. 609, no. 1, pp. 105–112, 2008.

[19] L. Szekely and A. Guttman, "New advances in microchip fabrication for electrochromatography," *Electrophoresis*, vol. 26, no. 24, pp. 4590–4604, 2005.

[20] S. A. Peyman, A. Iles, and N. Pamme, "Mobile magnetic particles as solid-supports for rapid surface-based bioanalysis in continuous flow," *Lab on a Chip*, vol. 9, no. 21, pp. 3110–3117, 2009.

[21] C. A. Schneider, W. S. Rasband, K. W. Eliceiri, *et al.*, "Nih image to imagej: 25 years of image analysis," *Nat methods*, vol. 9, no. 7, pp. 671–675, 2012.

[22] M. D. Tarn, S. A. Peyman, D. Robert, A. Iles, C. Wilhelm, and N. Pamme, "The importance of particle type selection and temperature control for on-chip free-flow magnetophoresis," *Journal of Magnetism and Magnetic Materials*, vol. 321, no. 24, pp. 4115–4122, 2009.

[23] D. Erickso and D. Li, "Microscale flow and transport simulation for electrokinetic and lab-on-chip applications," in *Biomems and biomedical nanotechnology*, pp. 277–300, Springer, 2006.

[24] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 4, pp. 271–288, 1998.

[25] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 780–785, 1997.

[26] T. M. Squires and S. R. Quake, "Microfluidics: Fluid physics at the nanoliter scale," *Reviews of modern physics*, vol. 77, no. 3, p. 977, 2005.

[27] H. A. Stone, A. D. Stroock, and A. Ajdari, "Engineering flows in small devices: microfluidics toward a lab-on-a-chip," *Annu. Rev. Fluid Mech.*, vol. 36, pp. 381–411, 2004.

[28] D. Auld and K. Srinivas, "Classification of flows, laminar and turbulent flows," 2006.

[29] L. Capretto, W. Cheng, M. Hill, and X. Zhang, "Micromixing within microfluidic devices," in *Microfluidics*, pp. 27–68, Springer, 2011.

[30] B. Lin, *Microfluidics: technologies and applications*, vol. 304. Springer, 2011.

[31] B. H. Weigl and P. Yager, "Microfluidic diffusion-based separation and detection," *Science*, vol. 283, no. 5400, p. 346, 1999.

[32] E. H. Altendorf, E. Iverson, D. Schutte, B. H. Weigl, T. D. Osborn, R. Sabeti, and P. Yager, "Optical flow cytometry utilizing microfabricated silicon flow channels," in *Photonics West'96*, pp. 267–276, International Society for Optics and Photonics, 1996.

[33] J. Labroquere, "Navier-stokes computation with num3sis poiseuille flow and backward facing step," 2012.

[34] H. E. Knoepfel, *Magnetic fields: a comprehensive theoretical treatise for practical use*. John Wiley & Sons, 2008.

[35] J. Kraus and D. Fleisch, *Electromagnetics: With Applications*. McGraw-Hill series in electrical and computer engineering, WCB/McGraw-Hill, 1999.

[36] R. M. Bozorth, *Ferromagnetism*, vol. 1. Wiley-VCH, 1993.

[37] G. Rizzoni and T. Hartley, *Principles and Applications of Electrical Engineering*. McGraw-Hill Higher Education, 2007.

[38] Magnasense, "Superparamagnetic nanospheres," 2015.

[39] Q. A. Pankhurst, J. Connolly, S. Jones, and J. Dobson, "Applications of magnetic nanoparticles in biomedicine," *Journal of physics D: Applied physics*, vol. 36, no. 13, p. R167, 2003.

[40] S. Sun, H. Zeng, D. B. Robinson, S. Raoux, P. M. Rice, S. X. Wang, and G. Li, "Monodisperse mfe2o4 (m= fe, co, mn) nanoparticles," *Journal of the American Chemical Society*, vol. 126, no. 1, pp. 273–279, 2004.

[41] R. Ganguly and I. K. Puri, "Field-assisted self-assembly of superparamagnetic nanoparticles for biomedical, mems and biomems applications," *Advances in applied mechanics*, vol. 41, pp. 294–336, 2007.

[42] Z. Ma, Y. Guan, X. Liu, and H. Liu, "Synthesis of magnetic chelator for high-capacity immobilized metal affinity adsorption of protein by cerium initiated graft polymerization," *Langmuir*, vol. 21, no. 15, pp. 6987–6994, 2005.

[43] K. S. Ryu, K. Shaikh, E. Goluch, Z. Fan, and C. Liu, "Micro magnetic stir-bar mixer integrated with parylene microfluidic channels," *Lab on a Chip*, vol. 4, no. 6, pp. 608–613, 2004.

[44] Y. Iiguni, M. Suwa, and H. Watarai, "High-magnetic-field electromagnetophoresis of micro-particles in a capillary flow system," *Journal of Chromatography A*, vol. 1032, no. 1, pp. 165–171, 2004.

[45] T. Kimura, M. Yamato, and A. Nara, "Particle trapping and undulation of a liquid surface using a microscopically modulated magnetic field," *Langmuir*, vol. 20, no. 3, pp. 572–574, 2004.

[46] T. Kimura, Y. Sato, F. Kimura, M. Iwasaka, and S. Ueno, "Micropatterning of cells using modulated magnetic fields," *Langmuir*, vol. 21, no. 3, pp. 830–832, 2005.

[47] A. Winkleman, K. L. Gudiksen, D. Ryan, G. M. Whitesides, D. Greenfield, and M. Prentiss, "A magnetic trap for living cells suspended in a paramagnetic buffer," *Applied physics letters*, vol. 85, no. 12, pp. 2411–2413, 2004.

[48] J. Happel, "Viscous flow in multiparticle systems: slow motion of fluids relative to beds of spherical particles," *AIChE Journal*, vol. 4, no. 2, pp. 197–201, 1958.

[49] A. Sinha, "Characterizing magnetic particle transport for microfluidic applications," 2008.

[50] L. Yuan-Hui and S. Gregory, "Diffusion of ions in sea water and in deep-sea sediments," *Geochimica et cosmochimica acta*, vol. 38, no. 5, pp. 703–714, 1974.

[51] Y. Okuhata, "Delivery of diagnostic agents for magnetic resonance imaging," *Advanced drug delivery reviews*, vol. 37, no. 1, pp. 121–137, 1999.

[52] K. Schulze, A. Koch, B. Schöpf, A. Petri, B. Steitz, M. Chastellain, M. Hofmann, H. Hofmann, and B. von Rechenberg, "Intraarticular application of superparamagnetic nanoparticles and their uptake by synovial membranean experimental study in sheep," *Journal of magnetism and magnetic materials*, vol. 293, no. 1, pp. 419–432, 2005.

[53] W. M. Saltzman, *Drug delivery: engineering principles for drug therapy*. Oxford University Press, 2001.

[54] K. C. Warnke, "Finite-element modeling of the separation of magnetic microparticles in fluid," *Magnetics, IEEE Transactions on*, vol. 39, no. 3, pp. 1771–1777, 2003.

[55] T. Schneider, S. Karl, L. R. Moore, J. J. Chalmers, P. S. Williams, and M. Zborowski, "Sequential cd34 cell fractionation by magnetophoresis in a magnetic dipole flow sorter," *Analyst*, vol. 135, no. 1, pp. 62–70, 2010.

[56] C. Phurimsak, M. D. Tarn, S. A. Peyman, J. Greenman, and N. Pamme, "On-chip determination of c-reactive protein using magnetic particles in continuous flow," *Analytical chemistry*, vol. 86, no. 21, pp. 10552–10559, 2014.

[57] J. Fraden, *Handbook of modern sensors: physics, designs, and applications*. Springer Science & Business Media, 2004.

[58] Y. Miwa, M. Kawabe, T. Okumura, Y. Yamashita, K. Kenmoku, and T. Kano, "Effectiveness of the vnd prevention device using magnets," in *World Congress on Medical Physics and Biomedical Engineering May 26-31, 2012, Beijing, China*, pp. 758–761, Springer, 2013.

[59] D. Heinrich, A. Goñi, T. Osán, L. Cerioni, A. Smessaert, S. Klapp, J. Faraudo, D. Pusiol, and C. Thomsen, "Effects of magnetic field gradients on the aggregation dynamics of colloidal magnetic nanoparticles," *Soft matter*, vol. 11, no. 38, pp. 7606–7616, 2015.

[60] M. Hoyos, L. Moore, P. S. Williams, and M. Zborowski, "The use of a linear halbach array combined with a step-splitt channel for continuous sorting of magnetic species," *Journal of magnetism and magnetic materials*, vol. 323, no. 10, pp. 1384–1388, 2011.

[61] A. van Reenen, A. M. de Jong, J. M. den Toonder, and M. W. Prins, "Integrated lab-on-chip biosensing systems based on magnetic particle actuation–a comprehensive review," *Lab on a Chip*, vol. 14, no. 12, pp. 1966–1986, 2014.

[62] J. Wang, N. Jiao, S. Tung, and L. Liu, "Magnetic microrobot and its application in a microfluidic system," *Robotics and Biomimetics*, vol. 1, no. 1, pp. 1–8, 2014.

[63] A. Alrifaiy, O. A. Lindahl, and K. Ramser, "Polymer-based microfluidic devices for pharmacy, biology and tissue engineering," *Polymers*, vol. 4, no. 3, pp. 1349–1398, 2012.

[64] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithms: a systematic survey," *Image Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 294–307, 2005.

[65] Y. Yacoob and M. J. Black, "Parameterized modeling and recognition of activities," in *Computer Vision, 1998. Sixth International Conference on*, pp. 120–127, IEEE, 1998.

[66] M. Petkovic and W. Jonker, "Content-based video retrieval by integrating spatio-temporal and stochastic recognition of events," in *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, pp. 75–82, IEEE, 2001.

[67] I. Haritaoglu, D. Harwood, and L. S. Davis, "W 4: Real-time surveillance of people and their activities," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 809–830, 2000.

[68] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 747–757, 2000.

[69] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," in *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*, pp. 8–14, IEEE, 1998.

[70] A. Talukder and L. Matthies, "Real-time detection of moving objects from moving vehicles using dense stereo and optical flow," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4, pp. 3718–3725, IEEE, 2004.

[71] A. M. McIvor, "Background subtraction techniques," *Proc. of Image and Vision Computing*, vol. 4, pp. 3099–3104, 2000.

[72] B. Li and R. Chellappa, "A generic approach to simultaneous tracking and verification in video," *Image Processing, IEEE Transactions on*, vol. 11, no. 5, pp. 530–544, 2002.

[73] O. Barnich and M. Van Droogenbroeck, "Vibe: A universal background subtraction algorithm for video sequences," *Image Processing, IEEE Transactions on*, vol. 20, no. 6, pp. 1709–1724, 2011.

[74] M. Cristani and V. Murino, "Background subtraction with adaptive spatio-temporal neighborhood analysis.," in *VISAPP (2)*, pp. 484–489, 2008.

[75] P.-M. Jodoin, V. Saligrama, and J. Konrad, "Behavior subtraction," *Image Processing, IEEE Transactions on*, vol. 21, no. 9, pp. 4244–4255, 2012.

[76] M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 657–662, 2006.

[77] J. C. Nascimento and J. S. Marques, "Performance evaluation of object detection algorithms for video surveillance," *Multimedia, IEEE Transactions on*, vol. 8, no. 4, pp. 761–774, 2006.

[78] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 255–261, IEEE, 1999.

[79] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review*, vol. 11, pp. 31–66, 2014.

[80] L. F. Teixeira, J. S. Cardoso, and L. Corte-Real, "Object segmentation using background modelling and cascaded change detection," *Journal of Multimedia*, vol. 2, no. 5, pp. 55–65, 2007.

[81] K. Greff, A. Brandão, S. Krauß, D. Stricker, and E. Clua, "A comparison between background subtraction algorithms using a consumer depth camera.," in *VISAPP (1)*, pp. 431–436, 2012.

[82] I. Huerta, D. Rowe, M. Á. Viñas, M. Mozerov, and J. Gonzàlez, "Background subtraction fusing colour, intensity and edge cues," 2007.

[83] M. Harville, G. Gordon, and J. Woodfill, "Foreground segmentation using adaptive mixture models in color and depth," in *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, pp. 3–11, IEEE, 2001.

[84] I. Huerta, D. Rowe, J. Gonzàlez, and J. J. Villanueva, "Efficient incorporation of motionless foreground objects for adaptive background segmentation," in *Articulated Motion and Deformable Objects*, pp. 424–433, Springer, 2006.

[85] S. C. Sen-Ching and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Electronic Imaging 2004*, pp. 881–892, International Society for Optics and Photonics, 2004.

[86] V. K. Asari, *Wide Area Surveillance: Real-time Motion Detection Systems*. Springer Publishing Company, Incorporated, 2013.

[87] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Computer VisionECCV 2000*, pp. 751–767, Springer, 2000.

[88] L. M. Fuentes and S. A. Elastin, "From tracking to advanced surveillance," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3, pp. III–121, IEEE, 2003.

[89] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 10, pp. 1337–1342, 2003.

[90] T. Boult, R. Micheals, X. Gao, P. Lewis, C. Power, W. Yin, and A. Erkan, "Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets," in *Visual Surveillance, 1999. Second IEEE Workshop on,(VS'99)*, pp. 48–55, IEEE, 1999.

[91] M. Piccardi, "Background subtraction techniques: a review," in *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4, pp. 3099–3104, IEEE, 2004.

[92] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, IEEE, 1999.

[93] P. W. Power and J. A. Schoonees, "Understanding background mixture models for foreground segmentation," in *Proceedings image and vision computing New Zealand*, vol. 2002, pp. 10–11, 2002.

[94] D.-S. Lee, "Effective gaussian mixture learning for video background subtraction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 5, pp. 827–832, 2005.

[95] M. Yin, H. Zhang, H. Meng, and X. Wang, "An hmm-based algorithm for vehicle detection in congested traffic situations," in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pp. 736–741, IEEE, 2007.

[96] H. Zhang and K. Wu, "A vehicle detection algorithm based on three-frame differencing and background subtraction," in *Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on*, vol. 1, pp. 148–151, IEEE, 2012.

[97] G. Lv, S. Zhao, and J. Zhao, "A new method of object detection based on three-frame difference and connectivity analyses," *Liquid Crystals and Displays*, vol. 22, 2007.

[98] M.-G. Gan, J. Chen, J. Liu, and Y.-N. Wang, "Moving object detection algorithm based on three-frame-differencing and edge information," *Dianzi Yu Xinxi Xuebao(Journal of Electronics and Information Technology)*, vol. 32, no. 4, pp. 894–897, 2010.

[99] K. K. Ng and E. J. Delp, "Background subtraction using a pixel-wise adaptive learning rate for object tracking initialization," in *IS&T/SPIE Electronic Imaging*, pp. 78820I–78820I, International Society for Optics and Photonics, 2011.

[100] L. Shapiro and G. C. Stockman, "Computer vision. 2001," *ed: Prentice Hall*, 2001.

[101] M. Weng, G. Huang, and X. Da, "A new interframe difference algorithm for moving target detection," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, vol. 1, pp. 285–289, IEEE, 2010.

[102] L. Zhao and X. He, "Adaptive gaussian mixture learning for moving object detection," in *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pp. 1176–1180, IEEE, 2010.

[103] P. Suo and Y. Wang, "An improved adaptive background modeling algorithm based on gaussian mixture model," in *Signal Processing, 2008. ICSP 2008. 9th International Conference on*, pp. 1436–1439, IEEE, 2008.

[104] P. L. M. Bouttefroy, A. Bouzerdoum, S. L. Phung, and A. Beghdadi, "On the analysis of background subtraction techniques using gaussian mixture models," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 4042–4045, IEEE, 2010.

[105] D.-S. Lee, J. J. Hull, and B. Erol, "A bayesian framework for gaussian mixture background modeling," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3, pp. III–973, IEEE, 2003.

[106] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 6, pp. 610–621, 1973.

[107] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[108] A. Hafiane, G. Seetharaman, and B. Zavidovique, "Median binary pattern for textures classification," in *Image Analysis and Recognition*, pp. 387–398, Springer, 2007.

[109] A. Hafiane, K. Palaniappan, and G. Seetharaman, "Adaptive median binary patterns for texture classification," in *2014 22nd International Conference on Pattern Recognition (ICPR)*, pp. 1138–1143, IEEE, 2014.

[110] A. Fernández, M. X. Álvarez, and F. Bianconi, "Texture description through histograms of equivalent patterns," *Journal of mathematical imaging and vision*, vol. 45, no. 1, pp. 76–102, 2013.

[111] M. Heikkilä, M. Pietikäinen, and J. Heikkilä, "A texture-based method for detecting moving objects.," in *BMVC*, pp. 1–10, 2004.

[112] S. Zhang and M. A. Karim, "A new impulse detector for switching median filters," *IEEE Signal processing letters*, vol. 9, no. 11, pp. 360–363, 2002.

[113] J. Lee and M. Park, "An adaptive background subtraction method based on kernel density estimation," *Sensors*, vol. 12, no. 9, pp. 12279–12300, 2012.

[114] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Background modeling and subtraction by codebook construction," in *Image Processing, 2004. ICIP'04. 2004 International Conference on*, vol. 5, pp. 3061–3064, IEEE, 2004.

[115] D. L. Olson and D. Delen, *Advanced data mining techniques*. Springer Science & Business Media, 2008.

[116] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikäinen, and S. Z. Li, "Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 1301–1306, IEEE, 2010.

[117] M. Narayana, A. Hanson, and E. G. Learned-Miller, "Background subtraction: separating the modeling and the inference," *Machine vision and applications*, vol. 25, no. 5, pp. 1163–1174, 2014.

[118] J.-C. Tai, S.-T. Tseng, C.-P. Lin, and K.-T. Song, "Real-time image tracking for automatic traffic monitoring and enforcement applications," *Image and Vision Computing*, vol. 22, no. 6, pp. 485–501, 2004.

[119] Q. Zhou and J. Aggarwal, "Object tracking in an outdoor environment using fusion of features and cameras," *Image and Vision Computing*, vol. 24, no. 11, pp. 1244–1255, 2006.

[120] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International journal of computer vision*, vol. 12, no. 1, pp. 43–77, 1994.

[121] S. Iwase and H. Saito, "Tracking soccer player using multiple views.," in *MVA*, pp. 102–105, 2002.

[122] O. Javed, S. Khan, Z. Rasheed, and M. Shah, "Camera handoff: tracking in multiple uncalibrated stationary cameras," in *Human Motion, 2000. Proceedings. Workshop on*, pp. 113–118, IEEE, 2000.

[123] V. Kettnaker and R. Zabih, "Bayesian multi-camera surveillance," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, IEEE, 1999.

[124] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, pp. 1197–1203, IEEE, 1999.

[125] D. Schugk, A. Kummert, and C. Nunn, "Adaptation of the mean shift tracking algorithm to monochrome vision systems for pedestrian tracking based on hog-features," *SAE Technical Paper*, pp. 01–0170, 2014.

[126] M. Yuan, F. Farbiz, C. M. Manders, and K. Y. Tang, "Robust hand tracking using a simple color classification technique," in *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, p. 6, ACM, 2008.

[127] Y. Ukrainitz and B. Sarel, "Mean shift: Theory and applications," *Weizmann Institute of Science, http://www. wisdom. weizmann. ac. il/˜ vision/courses/2004_2/files/mean_shift/mean_shift. ppt*, 2004.

[128] CSDN.NET, "Mean shift," 2014.

[129] Y. Cheng, "Mean shift, mode seeking, and clustering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 17, no. 8, pp. 790–799, 1995.

[130] M. Bayes and M. Price, "An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfrs," *Philosophical Transactions (1683-1775)*, pp. 370–418, 1763.

[131] S. M. Stigler, *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press, 1986.

[132] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 11, pp. 1778–1792, 2005.

[133] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[134] S. Messelodi, C. M. Modena, N. Segata, and M. Zanin, "A kalman filter based background updating algorithm robust to sharp illumination changes," in *Image Analysis and Processing–ICIAP 2005*, pp. 163–170, Springer, 2005.

[135] A. Pnevmatikakis and L. Polymenakos, "2d person tracking using kalman filtering and adaptive background learning in a feedback loop," in *Multimodal Technologies for Perception of Humans*, pp. 151–160, Springer, 2006.

[136] S. J. Press and J. S. Press, *Bayesian statistics: principles, models, and applications*. Wiley New York, 1989.

[137] C. W. Gardiner *et al.*, *Handbook of stochastic methods*, vol. 3. Springer Berlin, 1985.

[138] J. G. Kemeny, J. L. Snell, *et al.*, *Finite markov chains*, vol. 356. van Nostrand Princeton, NJ, 1960.

[139] G. Welch and G. Bishop, "An introduction to the kalman filter. department of computer science, university of north carolina," 2006.

[140] B. K. Horn and B. G. Schunck, "Determining optical flow," in *1981 Technical symposium east*, pp. 319–331, International Society for Optics and Photonics, 1981.

[141] A. Verri and T. Poggio, "Motion field and optical flow: Qualitative properties," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, no. 5, pp. 490–498, 1989.

[142] H. Fradi and J.-L. Dugelay, "Robust foreground segmentation using improved gaussian mixture model and optical flow," in *Informatics, Electronics & Vision (ICIEV), 2012 International Conference on*, pp. 248–253, IEEE, 2012.

[143] B. D. Lucas, "Generalized image matching by the method of differences," 1985.

[144] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision.," in *IJCAI*, vol. 81, pp. 674–679, 1981.

[145] H.-H. Nagel, "On the estimation of optical flow: Relations between different approaches and some new results," *Artificial intelligence*, vol. 33, no. 3, pp. 299–324, 1987.

[146] S. Uras, F. Girosi, A. Verri, and V. Torre, "A computational approach to motion perception," *Biological Cybernetics*, vol. 60, no. 2, pp. 79–87, 1988.

[147] B. Roberto, "Template matching techniques in computer vision," 2009.

[148] L. M. Fonseca and B. Manjunath, "Registration techniques for multisensor remotely sensed imagery," *PE & RS- Photogrammetric Engineering & Remote Sensing*, vol. 62, no. 9, pp. 1049–1056, 1996.

[149] R. Brunelli and T. Poggiot, "Template matching: Matched spatial filters and beyond," *Pattern Recognition*, vol. 30, no. 5, pp. 751–768, 1997.

[150] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Aerospace/Defense Sensing, Simulation, and Controls*, pp. 95–102, International Society for Optics and Photonics, 2001.

[151] Y. Bar-Shalom, *Tracking and data association*. Academic Press Professional, Inc., 1987.

[152] S. S. Intille, J. W. Davis, and A. E. Bobick, "Real-time closed-world tracking," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 697–703, IEEE, 1997.

[153] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[154] J. Rittscher, P. H. Tu, and N. Krahnstoever, "Simultaneous estimation of segmentation and shape," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 486–493, IEEE, 2005.

[155] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 810–815, 2004.

[156] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[157] T. McCreedy, "Fabrication techniques and materials commonly used for the production of microreactors and micro total analytical systems," *TrAC Trends in Analytical Chemistry*, vol. 19, no. 6, pp. 396–401, 2000.

[158] D. C. Harris, *Quantitative chemical analysis*. Macmillan, 2010.

[159] D. Swinehart, "The beer-lambert law," *Journal of chemical education*, vol. 39, no. 7, p. 333, 1962.

[160] D. MacDougall and W. B. Crummett, "Guidelines for data acquisition and data quality evaluation in environmental chemistry," *Analytical Chemistry*, vol. 52, no. 14, pp. 2242–2249, 1980.

[161] A. McNaught and A. Wilkinson, "Iupac compendium of chemical terminology, 2000," *International Union of Pure and Applied Chemistry*.

[162] G. L. Long and J. D. Winefordner, "Limit of detection. a closer look at the iupac definition," *Analytical Chemistry*, vol. 55, no. 7, pp. 712A–724A, 1983.

Appendices

Appendix A

IMAGE HISTOGRAM NORMALIZATION

Figure A.1: Histogram Normalization applied on image from PETS'09 dataset.

Appendix B

THREE FRAME DIFFERENCE

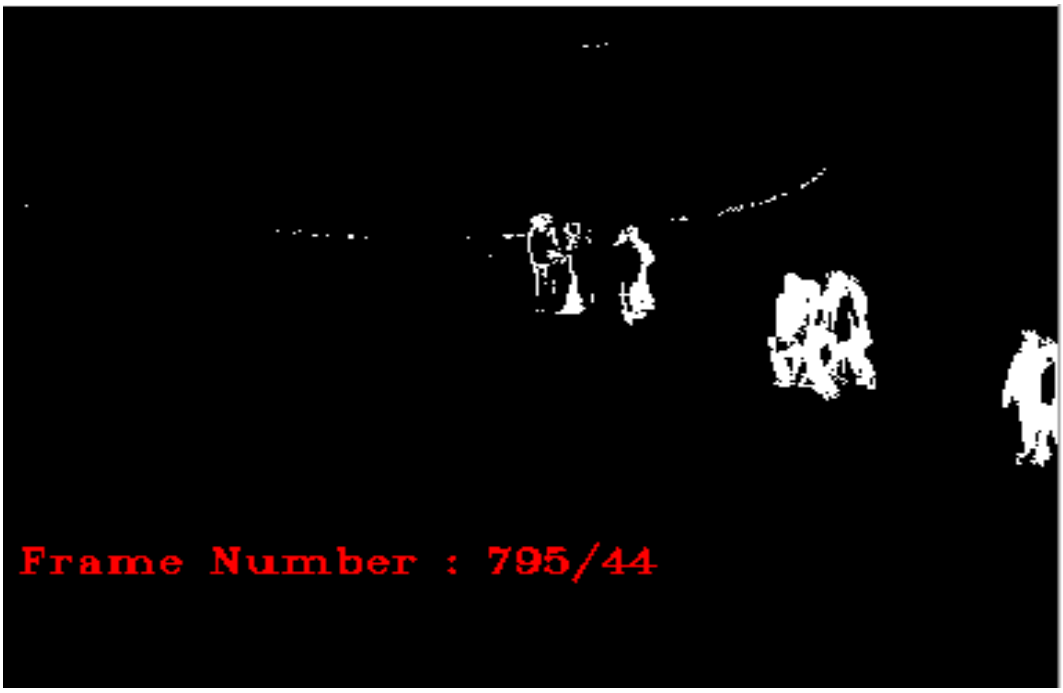


Figure B.1: Three frame difference applied on images from PETS'09 dataset.

Appendix C

GRADES AND PROPERTIES OF NEODYMIUM MAGNETS

Grade	**Br kGs**	**Hcb kOe**	**Hci kOe**	**(BH)max MGOe**	**Density** g/cm^3	**Max working Temp** ^oC
N30AH	10.8	10.2	35	30	7.8	220
N33AH	11.2	10.5	35	33	7.8	220
N35AH	11.8	11.0	35	35	7.8	220
N30EH	10.8	10.2	30	30	7.7	200
N33EH	11.2	10.5	30	33	7.7	200
N35EH	11.8	11.0	30	35	7.7	200
N38EH	12.2	11.4	30	38	7.7	200
N30UH	10.8	10.2	25	30	7.6	180
N33UH	11.2	10.5	25	33	7.6	180
N35UH	11.8	11.0	25	35	7.6	180
N38UH	12.2	11.4	25	38	7.6	180
N40UH	12.5	11.6	25	40	7.6	180
N33SH	11.2	10.5	20	33	7.5	150
N35SH	11.8	11.0	20	35	7.5	150
N38SH	12.2	11.4	20	38	7.5	150

N40SH	12.5	11.6	20	40	7.5	150
N42SH	12.8	11.6	20	42	7.5	150
N45SH	13.2	11.6	20	45	7.5	150
N35H	11.8	10.9	17	35	7.4	120
N38H	12.2	11.3	17	38	7.4	120
N40H	12.5	11.6	17	40	7.4	120
N42H	12.8	12.0	17	42	7.4	120
N45H	13.2	12.5	17	45	7.4	120
N48H	13.8	11.0	17	48	7.4	120
N50H	14.3	13.0	17	50	7.4	120
N35M	11.8	10.5	14	35	7.4	100
N38M	12.2	11.0	14	38	7.4	100
N40M	12.5	11.2	14	40	7.5	100
N42M	12.8	12.0	14	42	7.4	100
N45M	13.2	12.2	14	45	7.4	100
N48M	13.8	13.0	14	48	7.6	100
N50M	14.0	10.0	14	50	7.8	100
N35	11.8	10.9	12	35	7.4	80
N38	12.2	11.3	12	38	7.4	80
N40	12.5	11.6	12	40	7.5	80
N42	12.8	11.6	12	42	7.5	80
N45	13.2	11.0	12	45	7.6	80
N48	13.8	10.5	11	48	7.7	80
N50	14.0	10.0	11	40	7.8	80
N52	14.4	10.5	11	41	7.8	80

** Coating: Ni, Zn, Ni-Cu-Ni, Black Epoxy, Clear Cellulose

** Shape: Block, Disc, Ring, Arc

** Sintered NdFeB Grades (Source www.magnetsales.co.uk))

Appendix D

HUMAN TRACKING

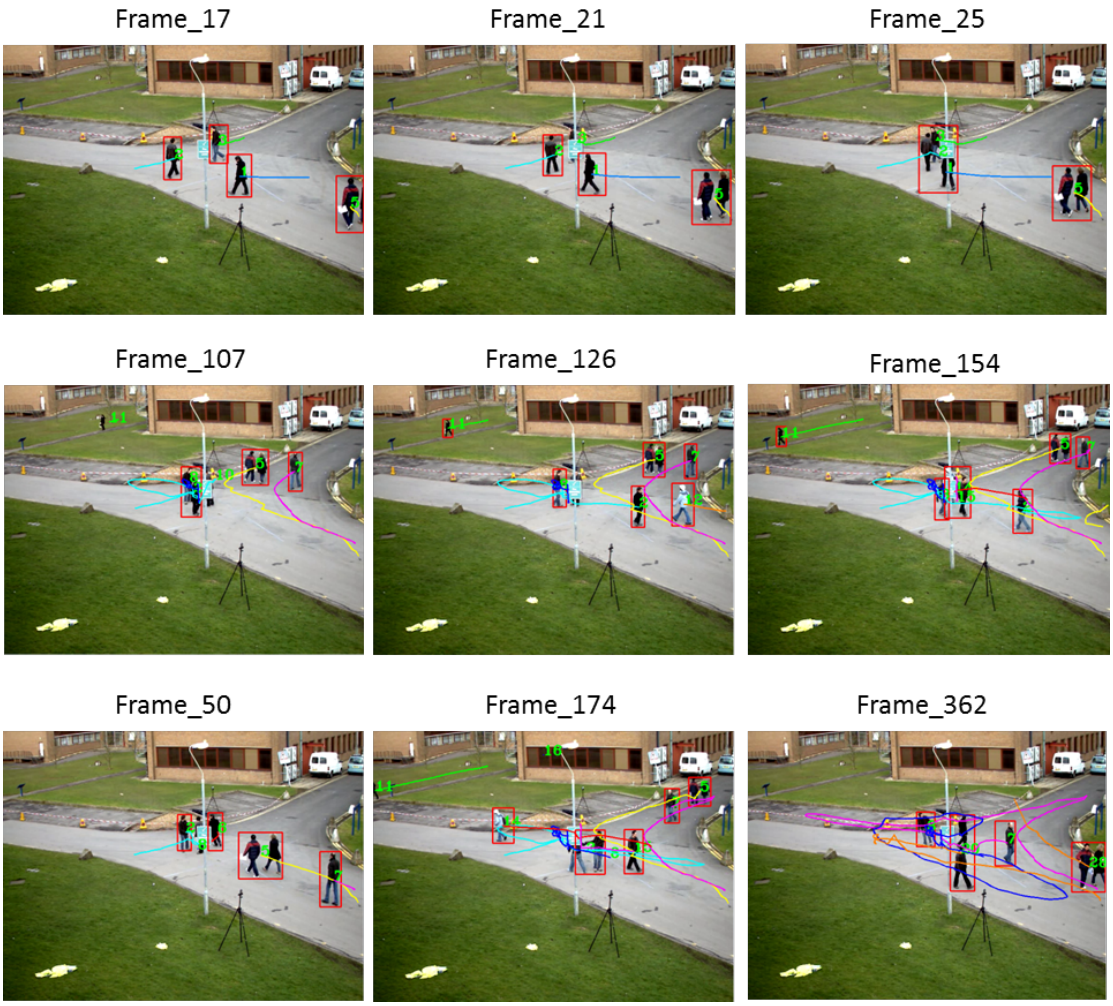


Figure D.1: Human tracking (correctly tracked) using proposed method from PETS'09 dataset.

The observations are

1. The tracking starts with tracking object 1, 2, 3 and 5
2. Object number 11 was tracked all the way since it came to the screen in frame 107 until it goes out from the scene in frame 154.
3. Object number 7 was tracked from Frame 50 till 362. It has several times overlap with other objects, the tracker still successful to recognize it.

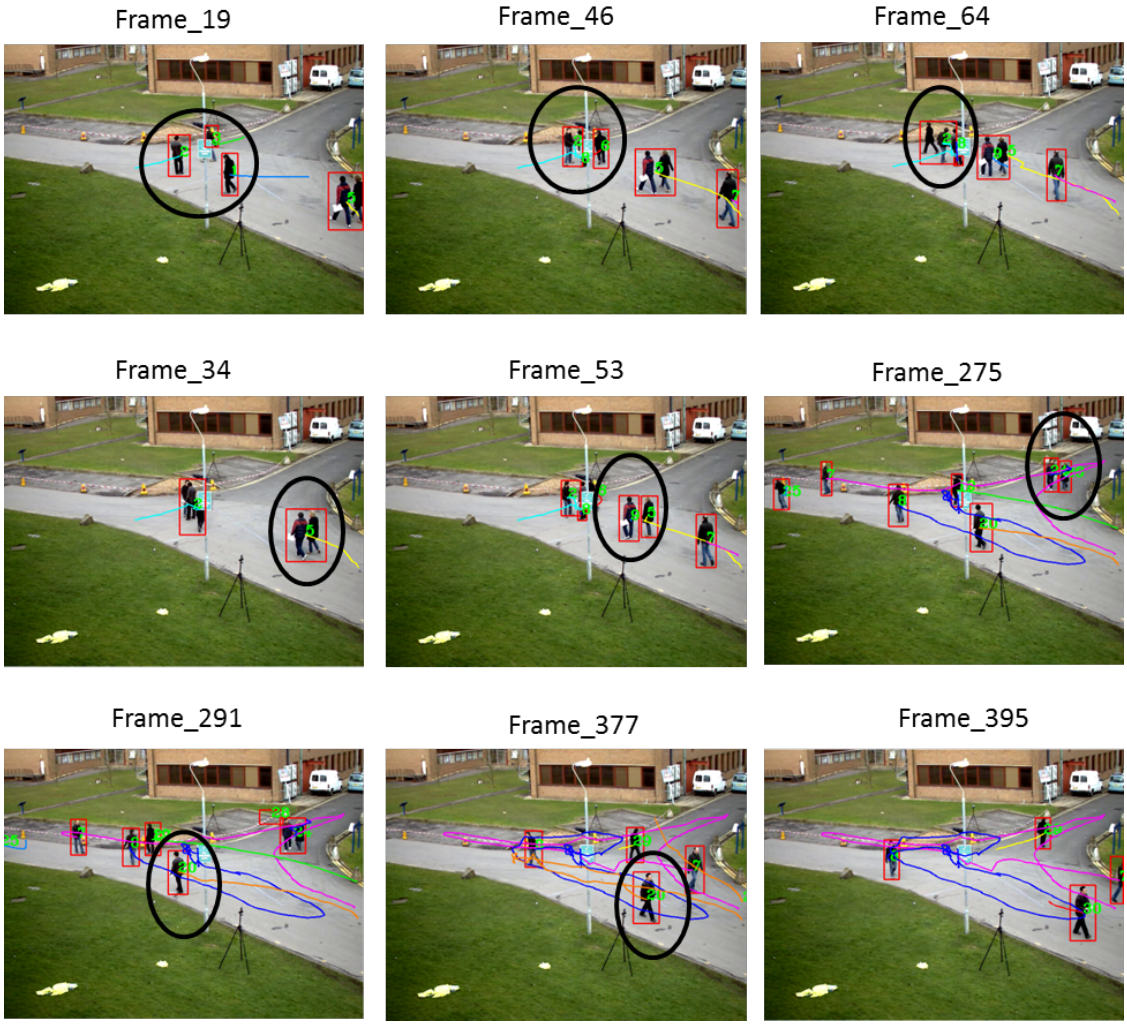


Figure D.2: Human tracking (false detection) using proposed method from PETS'09 dataset.

Observations are:

1. When objects overlap with each other, tracker fails to recognize the same object in next frame.

2. For two nearby objects, the tracker sometime recognizes as one object, sometime recognize as two individual objects with new ID number. For example, number 5 was initially recognized as one object, then ID changes.

3. Number 20 object was tracked over 100 frames and then changed to number 30. Same problem happened for two numbers 9 and 5, later they appeared as 24 and 26.

Appendix E

CODE EXPLANATION

BGModeling TAB

Step 1 : Inputing Image

```
void BgmodelingThread::start()
{
    ……
    QByteArray byteOpenName = m_strOpenFile.toLocal8Bit();
    char* szOpenFile = byteOpenName.data();
    char* szFileType;
    int ch = '.';
    szFileType = strrchr( szOpenFile, ch ) + 1;
    QString strFileType(szFileType);

    if(strFileType == "avi"||strFileType == "mp4")
    {
        VideoCapture video(szOpenFile);

        … …
    }
    else
    {
        TIFFCapture tiff(szOpenFile);

        … …
    }
}
```

Step 2 : Three Frame Different BgModeling

- Adaptive Thresholding

```
void FrameDiff_BgModeling::adaptiveGaussianThresholding(Mat
&foreground, int maxThreshValue, int threshType)
{
    int threshValue, maxValue = 256;
    double result, sum1, sum2, sum3, sum4;

    int min = 0;
    while ((foreground.data[min] == 0) && (min < maxValue))
        min++;

    int max = maxValue;
    while ((foreground.data[max] == 0) && (max > 0))
        max--;

    if (min>=max)
```

```
        threshValue = maxValue / 2;
    else
    {
        int movingIndex = min;
        do
        {
            sum1 = sum2 = sum3 = sum4 = 0.0;
            for (int i = min; i<=movingIndex; i++)
            {
                sum1 += i * foreground.data[i];
                sum2 += foreground.data[i];
            }

            for (int i = (movingIndex+1); i<= max; i++)
            {
                sum3 += i * foreground.data[i];
                sum4 += foreground.data[i];
            }

            result = (sum1/sum2 + sum3/sum4) / 2.0;
            movingIndex++;
        } while (((movingIndex+1) <= result) && (movingIndex < max-
1));

        threshValue = floor(result/10 + 0.5);
    }

    threshold(foreground, foreground, threshValue, maxThreshValue,
threshType);
}
```

- **Three Frame Different BgModeling**

```
void FrameDiff_BgModeling::process(const Mat &img_input, Mat
&img_output)
{
    if(img_input.empty())
        return;

    Mat proc_img;
    cvtColor(img_input, proc_img, CV_BGR2GRAY);

    if(img_input_prev.empty())
    {
        img_input_prev = proc_img.clone();
```

```
        img_output = Mat(proc_img.size(), CV_8UC1);
        img_output = Scalar::all(0);
        return;
    }

    if(img_input_last.empty())
    {
        img_input_last = img_input_prev.clone();
        img_output = Mat(proc_img.size(), CV_8UC1);
        img_output = Scalar::all(0);
        return;
    }

    img_foreground = Mat(proc_img.size(), CV_8UC3);
    img_foreground = Scalar::all(0);

    Mat temp1, temp2;
    absdiff(proc_img, img_input_prev, temp1);
    absdiff(proc_img, img_input_last, temp2);
    bitwise_and(temp1, temp2, img_foreground);

    if(img_foreground.type() == CV_8UC3)
        cvtColor(img_foreground, img_foreground, CV_BGR2GRAY);

    adaptiveGaussianThresholding(img_foreground, 255,
CV_THRESH_BINARY);
    dilate(img_foreground, img_foreground, Mat());

    img_foreground.copyTo(img_output);
    img_input_last = img_input_prev.clone();
    img_input_prev = proc_img.clone();
}
```

Step 3 : MBP BgModeling

- **Calculate MBP Feature**

```
void TextureBGS::MBP(RgbImage& image, RgbImage& texture)
{
    for(int y = TEXTURE_R; y < image.Ptr()->height-TEXTURE_R; ++y)
    {
        for(int x = TEXTURE_R; x < image.Ptr()->width-TEXTURE_R;
++x)
        {
            for(int ch = 0; ch < NUM_CHANNELS; ++ch)
            {
```

```
                vector<int> boundaryValues; // 3*3 window elements
exclude center
                vector<int> windowBins; // 3*3 window elements

                // get 3*3 window elements
                int val = (int)image(y, x, ch); // center value
                windowBins.push_back(val);

                val = (int)image(y-2, x, ch);
                windowBins.push_back(val);
                boundaryValues.push_back(val);

                val = (int)image(y-1, x-2, ch);
                windowBins.push_back(val);
                boundaryValues.push_back(val);

                val = (int)image(y-1, x+2, ch);
                windowBins.push_back(val);
                boundaryValues.push_back(val);

                val = (int)image(y+1, x-2, ch);
                windowBins.push_back(val);
                boundaryValues.push_back(val);

                val = (int)image(y+1, x+2, ch);
                windowBins.push_back(val);
                boundaryValues.push_back(val);

                val = (int)image(y+2, x, ch);
                windowBins.push_back(val);
                boundaryValues.push_back(val);

                // arrange 3*3 window elements
                sort(windowBins.begin(), windowBins.end());
                int centerValue = windowBins[3]; // median value for
MBP

                // This part is for equation 1.
                unsigned char textureCode = 0;

                // this only works for a texture radius of 2
                if(centerValue - boundaryValues[0] + HYSTERSIS >= 0)
                textureCode += 1;
```

```
                if(centerValue - boundaryValues[1] + HYSTERSIS >= 0)
                textureCode += 2;

                if(centerValue - boundaryValues[2] + HYSTERSIS >= 0)
                textureCode += 4;

                if(centerValue - boundaryValues[3] + HYSTERSIS >= 0)
                textureCode += 8;

                if(centerValue - boundaryValues[4] + HYSTERSIS >= 0)
                textureCode += 16;

                if(centerValue - boundaryValues[5] + HYSTERSIS >= 0)
                textureCode += 32;

                texture(y,x,ch) = textureCode;
            }
        }
    }
}
```

- **Calculate MBP Histogram**

```
void TextureBGS::Histogram(RgbImage& texture, TextureHistogram*
curTextureHist)
{
    // calculate histogram within a 2*REGION_R square
    for(int y = REGION_R+TEXTURE_R; y < texture.Ptr()->height-
REGION_R-TEXTURE_R; ++y)
    {
        for(int x = REGION_R+TEXTURE_R; x < texture.Ptr()->width-
REGION_R-TEXTURE_R;            ++x)
        {
            int index = x+y*(texture.Ptr()->width);

            // clear histogram
            for(int i = 0; i < NUM_BINS; ++i)
            {
                curTextureHist[index].r[i] = 0;
                curTextureHist[index].g[i] = 0;
                curTextureHist[index].b[i] = 0;
            }

            // calculate histogram
            for(int j = -REGION_R; j <= REGION_R; ++j)
```

```
            {
                for(int i = -REGION_R; i <= REGION_R; ++i)
                {
                    curTextureHist[index].r[texture(y+j,x+i,2)]++;
                    curTextureHist[index].g[texture(y+j,x+i,1)]++;
                    curTextureHist[index].b[texture(y+j,x+i,0)]++;
                }
            }
        }
    }
}
```

- **Check Similarity**

```
int TextureBGS::ProximityMeasure(TextureHistogram& bgTexture,
TextureHistogram& curTextureHist)
{
    int proximity = 0;
    for(int i = 0; i < NUM_BINS; ++i)
    {
        proximity += std::min(bgTexture.r[i], curTextureHist.r[i]);
        proximity += std::min(bgTexture.g[i], curTextureHist.g[i]);
        proximity += std::min(bgTexture.b[i], curTextureHist.b[i]);
    }

    return proximity;
}

void TextureBGS::BgsCompare(TextureArray* bgModel, TextureHistogram*
curTextureHist,
                unsigned char* modeArray, float threshold, BwImage&
fgMask)
{
    cvZero(fgMask.Ptr());

    for(int y = REGION_R+TEXTURE_R; y < fgMask.Ptr()->height-
REGION_R-TEXTURE_R; ++y)
    {
        for(int x = REGION_R+TEXTURE_R; x < fgMask.Ptr()->width-
REGION_R-TEXTURE_R;
                ++x)
        {
            int index = x+y*(fgMask.Ptr()->width);

            // find closest matching texture in background model
```

```
            int maxProximity = -1;

            for(int m = 0; m < NUM_MODES; ++m)
            {
                int proximity =
ProximityMeasure(bgModel[index].mode[m],
                        curTextureHist[index]);

                if(proximity > maxProximity)
                {
                    maxProximity = proximity;
                    modeArray[index] = m;
                }
            }

            if(maxProximity < threshold)
            fgMask(y,x) = 255;
        }
    }
}
```

- **Update Background Model**

```
void TextureBGS::UpdateModel(BwImage& fgMask, TextureArray* bgModel,
                 TextureHistogram* curTextureHist, unsigned char*
modeArray)
{
    for(int y = REGION_R+TEXTURE_R; y < fgMask.Ptr()->height-
REGION_R-TEXTURE_R; ++y)
    {
        for(int x = REGION_R+TEXTURE_R; x < fgMask.Ptr()->width-
REGION_R-TEXTURE_R;
                ++x)
        {
            int index = x+y*(fgMask.Ptr()->width);

            if(fgMask(y,x) == 0)
            {
                for(int i = 0; i < NUM_BINS; ++i)
                {
                    bgModel[index].mode[modeArray[index]].r[i]
                    = (unsigned
char)(ALPHA*curTextureHist[index].r[i]
                    + (1-
ALPHA)*bgModel[index].mode[modeArray[index]].r[i] + 0.5);
```

```
                }
            }
        }
    }
}
```

Step 4 : Combine Two Method

- Combine Two Method

```
void BgModeling::process(Mat src)
{
        Mat proc_img = src.clone();
        m_pFrameDiff->process(src, m_framediffImg);

        m_pMBP->process(src, m_mbpImg);
        m_histogram = m_pMBP->getHistogram();

        resize(m_mbpImg, m_mbpImg, m_framediffImg.size());
        bitwise_and(m_mbpImg, m_framediffImg, m_mbpImg);
        removeNoise();

        m_maskImg.release();
        Mat mask_fg = m_finalImg.clone();
        vector< vector<Point> > contours;
        vector<Vec4i> hierarchy;
        findContours(mask_fg, contours, hierarchy, CV_RETR_TREE,
                CV_CHAIN_APPROX_SIMPLE);
        if( contours.size() != 0 )
        {
                int idx = 0;
                for( ; idx >= 0; idx = hierarchy[idx][0] )
                {
                        drawContours(proc_img, contours, idx,
Scalar(0, 0, 255), 2);
                }
        }

        proc_img.copyTo(m_maskImg, m_finalImg);
}
```

- **Remove Noise**

```
void BgModeling::removeNoise()
{
        vector< vector<Point> > contours;
        vector<Vec4i> hierarchy;
```

```
        Mat proc_img = m_mbpImg.clone();
        findContours(proc_img, contours, hierarchy, CV_RETR_TREE,
                CV_CHAIN_APPROX_SIMPLE);
        Mat img_foreground = Mat(m_mbpImg.size(), CV_8UC1);
        img_foreground = Scalar::all(0);
        if(contours.size() != 0)
        {
                for(int idx = 0; idx >= 0; idx = hierarchy[idx][0])
                {
                        Rect contourRect =
boundingRect(contours[idx]);
                        if((contourRect.width <
10)&&(contourRect.height < 10))
                                continue;

                        drawContours(img_foreground, contours, idx,
Scalar(255),
                                CV_FILLED);
                 }
        }

        m_finalImg = img_foreground.clone();
}
```

Step 5 : Save Results

- Save Three Frame Different BgModeling Result

```
void BgmodelingThread::saveFrameDiff()
{
    QString strFrameDiffName = m_strSavePath + "/" + m_strSaveName +
"_framediff." +
        m_strSaveType;
    QByteArray byteFrameDiffName = strFrameDiffName.toLocal8Bit();
    char *szFrameDiffName = byteFrameDiffName.data();

    if((m_strSaveType == "avi"))
    {
        if(m_nFrameNum == 0)
            m_pFrameDiffVideo = cvCreateVideoWriter(szFrameDiffName,
CV_FOURCC('M', 'J', 'P', 'G'), m_nFps, m_frameDiffImage.size());

        IplImage saveArr = m_frameDiffImage;
        cvWriteFrame(m_pFrameDiffVideo, &saveArr);
```

```
        if(m_nFrameNum == m_nFrameCnt-1)
            cvReleaseVideoWriter(&m_pFrameDiffVideo);
    }
    else
    {
        if(m_nFrameNum == 0)
            m_pFrameDiffTiff = TIFFOpen(szFrameDiffName, "w");

        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_IMAGEWIDTH,
m_frameDiffImage.cols);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_IMAGELENGTH,
m_frameDiffImage.rows);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_BITSPERSAMPLE,8);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_SAMPLESPERPIXEL,3);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_PLANARCONFIG,
PLANARCONFIG_CONTIG);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_PHOTOMETRIC,
PHOTOMETRIC_MINISBLACK);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_ORIENTATION,
ORIENTATION_TOPLEFT);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_RESOLUTIONUNIT,
RESUNIT_INCH);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_XRESOLUTION, 100.0);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_YRESOLUTION, 100.0);
        TIFFSetField(m_pFrameDiffTiff, TIFFTAG_SUBFILETYPE,
FILETYPE_PAGE);

        for (int j = 0; j < m_frameDiffImage.rows; j++)
        {
            Mat rowImg = m_frameDiffImage.row(j);
            TIFFWriteScanline(m_pFrameDiffTiff, rowImg.data, j, 0);
        }

        TIFFWriteDirectory(m_pFrameDiffTiff);

        if(m_nFrameNum == m_nFrameCnt-1)
            TIFFClose(m_pFrameDiffTiff);
    }
}
```

- Save MBP BgModeling Result

  `void BgmodelingThread::saveMBP()`

- Save Mask Result

```
void BgmodelingThread::saveMask()
```

- **Save Histogram Result**

```
void BgmodelingThread::saveHistogram()
```

- **Save Final Result**

```
void BgmodelingThread::saveFinal()
```

Tracking TAB

Step 1 : Inputing Image

```
void TrackingThread::start()
{
    ……
    QByteArray byteOpenName = m_strOpenFile.toLocal8Bit();
    char* szOpenFile = byteOpenName.data();
    char* szFileType;
    int ch = '.';
    szFileType = strrchr( szOpenFile, ch ) + 1;
    QString strFileType(szFileType);

    if(strFileType == "avi"||strFileType == "mp4")
    {
        VideoCapture video(szOpenFile);

        … …
    }
    else
    {
        TIFFCapture tiff(szOpenFile);
        … …
    }
}
```

Step 2 : Background Modeling

```
void TrackingThread::process(Mat src, bool bVideo)
{
    ……
    if(m_nFrameNum > 2)
    {
        m_pTracking->process(src, m_fgImg, bVideo);
        m_resultImg = m_pTracking->getResultImage();
    }
    else
```

```
        m_resultImg = src.clone();

    ......
}
```

Step 3 : Kalman Tracking

```
void KalmanTracker::Update(vector<Point2d>& detections)
{
    if(tracks.size() == 0)
    {
        for(int i = 0; i < detections.size(); i++)
        {
            KalmanTrackEngine* tr = new
KalmanTrackEngine(detections[i], dt,
                Accel_noise_mag);
            tracks.push_back(tr);
        }
    }

    int N = tracks.size();
    int M = detections.size();
    vector< vector<double> > Cost(N,vector<double>(M));
    vector<int> assignment;

    double dist;
    for(int i = 0; i < tracks.size(); i++)
    {
        for(int j = 0; j < detections.size(); j++)
        {
            Point2d diff=(tracks[i]->prediction-detections[j]);
            dist = sqrtf(diff.x*diff.x+diff.y*diff.y);
            Cost[i][j]=dist;
        }
    }

    KalmanAssignment APS;
    APS.Solve(Cost, assignment, KalmanAssignment::optimal);

    vector<int> not_assigned_tracks;
    for(int i = 0; i  <assignment.size(); i++)
    {
        if(assignment[i] != -1)
        {
            if(Cost[i][assignment[i]] > dist_thres)
```

```
            {
                assignment[i] = -1;
                not_assigned_tracks.push_back(i);
            }
        }
        else
        {
            tracks[i]->skipped_frames++;
        }
    }

    for(int i = 0; i < tracks.size(); i++)
    {
        if(tracks[i]->skipped_frames>maximum_allowed_skipped_frames)
        {
            delete tracks[i];
            tracks.erase(tracks.begin()+i);
            assignment.erase(assignment.begin() + i);
            i--;
        }
    }

    vector<int> not_assigned_detections;
    vector<int>::iterator it;

    for(int i = 0; i < detections.size() ;i++)
    {
        It = find(assignment.begin(), assignment.end(), i);
        if(it==assignment.end())
        {
            not_assigned_detections.push_back(i);
        }
    }

    if(not_assigned_detections.size() != 0)
    {
        for(int i = 0; i < not_assigned_detections.size(); i++)
        {
            KalmanTrackEngine* tr = new KalmanTrackEngine
               (detections[not_assigned_detections[i]], dt,
Accel_noise_mag);
            tracks.push_back(tr);
        }
    }
```

```
    for(int i = 0; i < assignment.size(); i++)
    {
        tracks[i]->KF->GetPrediction();

        if(assignment[i] != -1)
        {
            tracks[i]->skipped_frames = 0;

tracks[i]->prediction=tracks[i]->KF->Update(detections[assignment[i]
], 1);
        }else
        {

tracks[i]->prediction=tracks[i]->KF->Update(Point2f(0,0),0);
        }

        if(tracks[i]->trace.size()>max_trace_length)
        {
            tracks[i]->trace.erase(tracks[i]->trace.begin(),
tracks[i]->trace.end()-
                max_trace_length);
        }

        tracks[i]->trace.push_back(tracks[i]->prediction);
        tracks[i]->KF->LastResult=tracks[i]->prediction;
    }

}
```

Step 4 : MeanShift Tracking

```
void MeanShift_Optical_Hog_Tracking::process(Mat src, RESULT_OBJECTS
currentObjects, bool bVideo)
{
    objectBufferFormat(&m_currentMeanshiftObjects);

    for(int i = 0; i < currentObjects.objectPositions.size(); i++)
    {
        m_currentMeanshiftObjects.objectCnt++;

m_currentMeanshiftObjects.objectRects.push_back(currentObjects.objec
tRects[i]);
    }
```

```
    if (m_currentMeanshiftObjects.objectCnt > 0)
    {
        getObjectID(bVideo);

        m_thirdMeanshiftObjects = m_lastMeanshiftObjects;
        m_lastMeanshiftObjects = m_beforeMeanshiftObjects;
        m_beforeMeanshiftObjects = m_currentMeanshiftObjects;
    }
    else
    {
        m_thirdMeanshiftObjects = m_lastMeanshiftObjects;
        m_lastMeanshiftObjects = m_beforeMeanshiftObjects;
        objectBufferFormat(&m_beforeMeanshiftObjects);
    }

    m_resultImg = src.clone();
    if(!m_lastFrame.empty()) m_thirdFrame = m_lastFrame.clone();
    if(!m_beforeFrame.empty()) m_lastFrame = m_beforeFrame.clone();
    if(!m_currentFrame.empty()) m_beforeFrame =
m_currentFrame.clone();
    m_currentFrame = src.clone();

    ……
}
```

Step 5 : Check Direction Using Optical Flow

```
void MeanShift_Optical_Hog_Tracking::getObjectID(bool bVideo)
{
    INDIVIDUAL_OBJECT object;
    vector<int> currentObjectIDs;

    for (int i = 0; i < m_currentMeanshiftObjects.objectCnt; i++)
    {
        //object.frameNum = m_nFrameNum;
        object.objectRect =
m_currentMeanshiftObjects.objectRects[i];

        bool isNewObject = true;

        for (int j = 0; j < m_beforeMeanshiftObjects.objectCnt; j++)
        {
            bool bExistID = false;
            Point currentPt =
Point(m_currentMeanshiftObjects.objectRects[i].x +
```

```
                m_currentMeanshiftObjects.objectRects[i].width/2,
                m_currentMeanshiftObjects.objectRects[i].y +
                m_currentMeanshiftObjects.objectRects[i].height);
            Point beforePt =
Point(m_beforeMeanshiftObjects.objectRects[j].x +
                m_beforeMeanshiftObjects.objectRects[j].width/2,

                m_beforeMeanshiftObjects.objectRects[j].y +
                m_beforeMeanshiftObjects.objectRects[j].height);

            int angle = getAngle(beforePt, currentPt);

            if(!bVideo)
            {
                if((abs(currentPt.x - beforePt.x) >
5)&&(abs(currentPt.y -
                        beforePt.y) > 5))
                    if(currentPt.x < beforePt.x)
                        continue;

                int max_y = m_currentFrame.rows * 95 / 100;
                if(max_y < beforePt.y)
                    continue;

                int max_x = m_currentFrame.cols * 95 / 100;
                if(max_x < beforePt.x)
                    continue;

                if(m_beforeMeanshiftObjects.angles[j] != 1000)
                    if(abs(angle-
m_beforeMeanshiftObjects.angles[j]) > 3)
                        continue;
            }

            ......
    }
}
```

Step 6 : Check Similarity Using Hog

```
bool MeanShift_Optical_Hog_Tracking::isMatched(Mat src1, Mat src2)
{
    Mat img1 = src1.clone();
    Mat img2 = src2.clone();
    resize(img2, img2, img1.size());
```

```
    Mat img1_hsv, img2_hsv;
    cvtColor(img1, img1_hsv, CV_RGB2HSV);
    cvtColor(img2, img2_hsv, CV_RGB2HSV);

    int histSize[] = {img1.size().width, img1.size().height};
    float hranges[] = { 0, 180 };
    float sranges[] = { 0, 256 };
    const float* ranges[] = { hranges, sranges };

    cv::MatND img1Hist, img2Hist;
    int channels[] = {0, 1};

    calcHist( &img2_hsv, 1, channels, cv::Mat(), img2Hist, 2,
histSize, ranges, true,
        false );
    calcHist( &img1_hsv, 1, channels, cv::Mat(), img1Hist, 2,
histSize, ranges, true,
        false );

    double intersection = compareHist(img1Hist, img2Hist,
CV_COMP_CORREL);

    if (intersection > 0.5)
        return true;
    else
        return false;
}
```

Step 7 : Save Results

```
void TrackingThread::saveResult()
{
    QString strResultName = m_strSavePath + "/" + m_strSaveName +
"_tracking." +
        m_strSaveType;
    QByteArray byteResultName = strResultName.toLocal8Bit();
    char *szResultName = byteResultName.data();

    if((m_strSaveType == "avi"))
    {
        if(m_nFrameNum == 0)
            m_pResultVideo = cvCreateVideoWriter(szResultName,
CV_FOURCC('M', 'J', 'P',
                'G'), m_nFps, m_resultImg.size());
```

```
        IplImage saveArr = m_resultImg;
        cvWriteFrame(m_pResultVideo, &saveArr);

        if(m_nFrameNum == m_nFrameCnt-1)
            cvReleaseVideoWriter(&m_pResultVideo);
    }
    else
    {
        if(m_nFrameNum == 0)
            m_pResultTiff = TIFFOpen(szResultName, "w");

        TIFFSetField(m_pResultTiff, TIFFTAG_IMAGEWIDTH,
m_resultImg.cols);
        TIFFSetField(m_pResultTiff, TIFFTAG_IMAGELENGTH,
m_resultImg.rows);
        TIFFSetField(m_pResultTiff, TIFFTAG_BITSPERSAMPLE,8);
        TIFFSetField(m_pResultTiff, TIFFTAG_SAMPLESPERPIXEL,3);
        TIFFSetField(m_pResultTiff, TIFFTAG_PLANARCONFIG,
PLANARCONFIG_CONTIG);
        TIFFSetField(m_pResultTiff, TIFFTAG_PHOTOMETRIC,
PHOTOMETRIC_MINISBLACK);
        TIFFSetField(m_pResultTiff, TIFFTAG_ORIENTATION,
ORIENTATION_TOPLEFT);
        TIFFSetField(m_pResultTiff, TIFFTAG_RESOLUTIONUNIT,
RESUNIT_INCH);
        TIFFSetField(m_pResultTiff, TIFFTAG_XRESOLUTION, 100.0);
        TIFFSetField(m_pResultTiff, TIFFTAG_YRESOLUTION, 100.0);
        TIFFSetField(m_pResultTiff, TIFFTAG_SUBFILETYPE,
FILETYPE_PAGE);

        for (int j = 0; j < m_resultImg.rows; j++)
        {
            Mat rowImg = m_resultImg.row(j);
            TIFFWriteScanline(m_pResultTiff, rowImg.data, j, 0);
        }

        TIFFWriteDirectory(m_pResultTiff);

        if(m_nFrameNum == m_nFrameCnt-1)
            TIFFClose(m_pResultTiff);
    }
}
```

Analysis TAB

Step 1 : Inputing Image

```
void ErrorbarThread::process()
{
    QByteArray byteOpenName = m_strOpenFile.toLocal8Bit();
    char* szOpenFile = byteOpenName.data();
    char* szFileType;
    int ch = '.';
    szFileType = strrchr( szOpenFile, ch ) + 1;
    QString strFileType(szFileType);

    QString strFgFile = m_strSavePath + "/" + m_strResultName +
"(fg).tif";
    QByteArray byteFgFile = strFgFile.toLocal8Bit();
    char* szFgFile = byteFgFile.data();
    m_pFgTiff = TIFFOpen(szFgFile, "w");

    QString strMedianFile = m_strSavePath + "/" + m_strResultName +
"(median).tif";
    QByteArray bytMedianFile = strMedianFile.toLocal8Bit();
    char* szMedianFile = bytMedianFile.data();
    m_pMedianTiff = TIFFOpen(szMedianFile, "w");

    if(strFileType == "avi"||strFileType == "mp4")
    {
        VideoCapture video(szOpenFile);
        m_nFrameNum = 0;
        m_nFrameCnt = video.get(CV_CAP_PROP_FRAME_COUNT);
        m_nFps = video.get(CV_CAP_PROP_FPS);
        m_nHeight = video.get(CV_CAP_PROP_FRAME_HEIGHT);
        m_bVideo = true;

        for(;;)
        {
            Mat currentImg;
            video.operator>>(currentImg);
            if(currentImg.empty())
                break;

            m_intensityImg = currentImg.clone();

            // object buffer reset
            m_nMovingThreshold_X = currentImg.cols/5;
```

```
            m_nMovingThreshold_Y = currentImg.rows/5;

            Track(currentImg);

            m_nProgressValue = 95 * m_nFrameNum / m_nFrameCnt;
            if(m_nProgressValue != m_nLastProgressValue)
            {
                setProgress(m_nProgressValue, m_nThreadIdx);
                m_nLastProgressValue = m_nProgressValue;
            }
        }
    }
    else
    {
        TIFFCapture tiff(szOpenFile);
        m_nFrameNum = 0;
        m_nFps = 30;
        m_nFrameCnt = tiff.get(TIFF_CAP_PROP_FRAME_COUNT);
        m_nHeight = tiff.get(TIFF_CAP_PROP_FRAME_HEIGHT);
        m_bVideo = false;

        for (;;)
        {
            Mat currentImg;
            tiff.operator>>(currentImg);
            if(currentImg.empty())
                break;

            m_intensityImg = currentImg.clone();

            // object buffer reset
            m_nMovingThreshold_X = 150;
            m_nMovingThreshold_Y = 60;

            Track(currentImg);

            m_nProgressValue = 95 * m_nFrameNum / m_nFrameCnt;
            if(m_nProgressValue != m_nLastProgressValue)
            {
                setProgress(m_nProgressValue, m_nThreadIdx);
                m_nLastProgressValue = m_nProgressValue;
            }
        }
    }
```

```
    TIFFClose(m_pFgTiff);
    TIFFClose(m_pMedianTiff);

    completeTrack();
    setProgress(97, m_nThreadIdx);

    saveResultFiles();
    setProgress(98, m_nThreadIdx);
    saveErrorBarData();
    setProgress(99, m_nThreadIdx);

    m_allObjectsArray.clear();
    m_completeObjectsArray.clear();
    objectBufferFormat(&m_currentObjects);
    objectBufferFormat(&m_firstLastObjects);
    objectBufferFormat(&m_secondLastObjects);
    objectBufferFormat(&m_thirdLastObjects);
    setProgress(100, m_nThreadIdx);
}
```

Step 2 : Background Modeling

- Get Threshold Image

```
Mat ErrorbarThread::getThresholdImage(Mat src)
{
    Mat fg_img;

    m_pBgModeling->process(src, fg_img);
    if(fg_img.empty())
    {
        fg_img = Mat(src.size(), CV_8UC1);
        fg_img = Scalar::all(0);
    }

    return fg_img;
}
```

- Detect Objects

```
void ErrorbarThread::detectObjects(Mat src)
{
    Mat procImg = src.clone();
    cv::findContours(procImg, m_contours, m_hierarchy, CV_RETR_TREE,
        CV_CHAIN_APPROX_SIMPLE);
```

```
    if( m_contours.size() != 0 )
    {
        int idx = 0;

        for( ; idx >= 0; idx = m_hierarchy[idx][0] )
        {
            Rect contourRect = boundingRect(m_contours[idx]);

            if ((contourRect.width > src.cols/2)
||(contourRect.height > src.rows/2) ||
                (contourRect.width <
OBJECT_SIZE_TIFF_MIN)||(contourRect.height <
                OBJECT_SIZE_TIFF_MIN))
                 continue;

            m_currentObjects.objectCnt++;
            m_currentObjects.objectRects.push_back(contourRect);
            Point contourCenter =
Point(contourRect.x+contourRect.width/2,
                contourRect.y+contourRect.height/2);
            m_currentObjects.positions.push_back(contourCenter);
            int nIntensity = calcIntensity(contourRect);
            m_currentObjects.intensities.push_back(nIntensity);
        }
    }
}
```

Step 3 : Object Tracking

```
void ErrorbarThread::trackObjects()
{
    INDIVIDUAL_OBJECT object;
    if (m_currentObjects.objectCnt > 0)
    {
        if (m_firstLastObjects.objectCnt > 0)
        {
            setObjectID();
        }
        else
        {
            for (int i = 0; i < m_currentObjects.objectCnt; i++)
            {
                m_nObjectID++;
                m_currentObjects.objectIDs.push_back(m_nObjectID);
                object.frameNum = m_nFrameNum;
```

```
                object.objectID = m_nObjectID;
                object.position = m_currentObjects.positions[i];
                object.objectRect = m_currentObjects.objectRects[i];
                object.intensity = m_currentObjects.intensities[i];
                m_allObjectsArray.push_back(object);
            }
        }

        m_thirdLastObjects = m_secondLastObjects;
        m_secondLastObjects = m_firstLastObjects;
        m_firstLastObjects = m_currentObjects;
    }
    else
    {
        m_thirdLastObjects = m_secondLastObjects;
        m_secondLastObjects = m_firstLastObjects;
        objectBufferFormat(&m_firstLastObjects);
    }
}

void ErrorbarThread::completeTrack()
{
    // rank id
    OBJECTS_ARRAY tempArray;
    vector<int> vecObjectCountList;
    vector<int>::iterator iter;

    for (int i = 0; i < m_nObjectID; i++)
    {
        vecObjectCountList.push_back(0);
    }

    for (int i = 0; i < (int)m_allObjectsArray.size(); i++)
    {
        iter = vecObjectCountList.begin();
        int nObjectCount =
vecObjectCountList[m_allObjectsArray[i].objectID - 1] + 1;
        iter = iter + m_allObjectsArray[i].objectID - 1;
        vecObjectCountList.erase(iter);
        vecObjectCountList.insert(iter, nObjectCount);
    }

    int nMeanObjectCnt = 0;
    for(int i = 0; i < vecObjectCountList.size(); i++)
```

```
{
    nMeanObjectCnt += vecObjectCountList[i];
}
nMeanObjectCnt = nMeanObjectCnt/vecObjectCountList.size();

for (int i = 0; i < (int)vecObjectCountList.size(); i++)
{
    for (int j = 0; j < (int)m_allObjectsArray.size(); j++)
    {
        if (m_allObjectsArray[j].objectID == (i + 1))
        {
            tempArray.push_back(m_allObjectsArray[j]);
        }
    }
}

// remove less 3 object
vector< vector<INDIVIDUAL_OBJECT> > allDatas;
vector<INDIVIDUAL_OBJECT> eachObjDatas;
int id = 1;
for (int i = 0; i < tempArray.size(); i++)
{
    if(i == 0)
        id = tempArray[i].objectID;

    if(id == tempArray[i].objectID)
    {
        eachObjDatas.push_back(tempArray[i]);
    }
    else
    {
        allDatas.push_back(eachObjDatas);
        eachObjDatas.clear();
        id = tempArray[i].objectID;
    }
}

tempArray.clear();
for(int i = 0; i < allDatas.size(); i++)
{
    if(allDatas[i].size() > 3)
    {
        for(int j = 0; j < allDatas[i].size(); j++)
            tempArray.push_back(allDatas[i][j]);
```

```
    }
}

// check moving position
Point startPt, endPt;
vector<int> removeObjectIDs;
int currentID = 0;
for(int i = 0; i < tempArray.size(); i++)
{
    if(i == 0)
    {
        currentID = tempArray[i].objectID;
        startPt = tempArray[i].position;
        endPt = tempArray[i].position;
    }

    if(currentID == tempArray[i].objectID)
    {
        endPt = tempArray[i].position;
    }
    else
    {
        int move_x = abs(endPt.x - startPt.x);
        int move_y = abs(endPt.y - endPt.y);

        if(move_x < OBJECT_NON_MOVE && move_y < OBJECT_NON_MOVE)
        {
            removeObjectIDs.push_back(currentID);
        }

        currentID = tempArray[i].objectID;
        startPt = tempArray[i].position;
        endPt = tempArray[i].position;
    }
}

// remove non-moving objects
OBJECTS_ARRAY removeNonMovingObjects;
for(int i = 0; i < tempArray.size(); i++)
{
    bool isRemoveObject = false;
    for(int j = 0; j < removeObjectIDs.size(); j++)
    {
        if(tempArray[i].objectID == removeObjectIDs[j])
```

```
            {
                isRemoveObject = true;
                break;
            }
        }

        if(!isRemoveObject)
            removeNonMovingObjects.push_back(tempArray[i]);
    }

    // arrange objects
    INDIVIDUAL_OBJECT object;
    int tempID = 0, realID = 0;

    for (int i = 0; i < (int)removeNonMovingObjects.size(); i++)
    {
        if (tempID == removeNonMovingObjects[i].objectID)
        {
            object = removeNonMovingObjects[i];
            object.objectID = realID;
            m_completeObjectsArray.push_back(object);
        }
        else
        {
            tempID = removeNonMovingObjects[i].objectID;
            realID++;
            object = removeNonMovingObjects[i];
            object.objectID = realID;
            m_completeObjectsArray.push_back(object);
        }
    }
}
```

Step 4 : Save Excel Files

```
void ErrorbarThread::saveErrorBarData()
{
    if(m_completeObjectsArray.size() > 0)
    {
        vector< vector<INDIVIDUAL_OBJECT> > allDatas;
        vector<int> realIntensities;
        QString strAllObjectCsvName = m_strSavePath + "/" +
m_strResultName + ".csv";
        QFile allObjFile(strAllObjectCsvName);
```

```
        QString strFirstExcelName = m_strSavePath + "/" +
m_strResultName +
                "_first.csv";
        QFile firstExcelFile(strFirstExcelName);
        firstExcelFile.open(QFile::WriteOnly|QFile::Truncate);
        QString strSecondExcelName = m_strSavePath + "/" +
m_strResultName +
                "_second.csv";
        QFile secondExcelFile(strSecondExcelName);
        secondExcelFile.open(QFile::WriteOnly|QFile::Truncate);

        if (allObjFile.open(QFile::WriteOnly|QFile::Truncate))
        {
            QTextStream stream(&allObjFile);
            QStringList strList;

            strList.clear();
            strList << "ID" << "Frame Number" << "Intensity" <<
"State" << "Speed" <<
                "Position" << "Median Values";
            stream << strList.join( "," ) + "\n";
            strList.clear();

            QTextStream firstStream(&firstExcelFile);
            QTextStream secondStream(&secondExcelFile);
            QStringList strFirst, strSecond;

            strFirst.clear();
            strFirst<<"Object ID"<<"Frame
Number"<<"Intensity"<<"Speed"<<"Moving";
            firstStream<<strFirst.join(",") + "\n";
            strFirst.clear();

            strSecond.clear();
            strSecond<<"Object ID"<<"Intensity";
            secondStream<<strSecond.join(",") + "\n";
            strSecond.clear();

            vector<INDIVIDUAL_OBJECT> eachObjDatas;
            vector<int> eachIntensities;
            vector<int>::iterator iter;

            int id = 1;
            for (int i = 0; i < m_completeObjectsArray.size(); i++)
```

```
            {
                if(id == m_completeObjectsArray[i].objectID)
                {

eachObjDatas.push_back(m_completeObjectsArray[i]);
                }
                else
                {
                    allDatas.push_back(eachObjDatas);
                    eachObjDatas.clear();
                    id = m_completeObjectsArray[i].objectID;
                }
            }

            for(int i = 0; i < allDatas.size(); i++)
            {
                eachIntensities.clear();

                for(int j = 0; j < allDatas[i].size(); j++)
                {
                    int intensity = allDatas[i][j].intensity;
                    if(j == 0)
                    {
                        eachIntensities.push_back(intensity);
                    }
                    else
                    {
                        iter = eachIntensities.begin();
                        bool bInsert = false;
                        for(int k = 0; k < eachIntensities.size();
k++)
                        {
                            if(intensity < eachIntensities[k])
                            {
                                eachIntensities.insert(iter,
intensity);
                                bInsert = true;
                                break;
                            }

                            iter++;
                        }

                        if(!bInsert)
```

```
                    eachIntensities.push_back(intensity);
                }
            }

            int median_idx, median_val, before_val, next_val;
            if(allDatas[i].size() % 2 == 0)
                median_idx = allDatas[i].size() / 2 - 1;
            else
                median_idx = allDatas[i].size() / 2;
            median_val = eachIntensities[median_idx];
            before_val = eachIntensities[median_idx-1];
            next_val = eachIntensities[median_idx+1];

            realIntensities.push_back(before_val);
            realIntensities.push_back(median_val);
            realIntensities.push_back(next_val);

            QString strID, strMedians;
            for(int j = 0; j < allDatas[i].size(); j++)
            {
                strID.clear();
                strID =
QString::number(allDatas[i][j].objectID);
                strList << strID;
                strFirst << strID;

                QString strFrameNumber =
QString::number(allDatas[i][j].frameNum);
                strList << strFrameNumber;
                strFirst << strFrameNumber;

                QString strIntensity =
QString::number(allDatas[i][j].intensity);
                strList << strIntensity;
                strFirst << strIntensity;

                if(j == 0)
                {
                    strList << "Stop";
                    strList << "0";
                }
                else
                {
```

```
                        int move_value = isMoved(allDatas[i][j-
1].position,
                              allDatas[i][j].position);
                        if(move_value > 10)
                            strList << "Move";
                        else
                            strList << "Stop";

                        QString strSpeed =
QString::number(move_value);
                        strList << strSpeed;
                        strFirst << strSpeed;

                        if(move_value > 10)
                            strFirst << "Move";
                        else
                            strFirst << "Stop";
                    }

                    QString strPosition =
QString::number(allDatas[i][j].position.y *
                       255 / m_intensityImg.rows);
                    strList << strPosition;

                    strMedians.clear();
                    strMedians = QString::number(before_val) + ". "
+
                       QString::number(median_val) + ". " +
QString::number(next_val);
                    strList << strMedians;

                    stream << strList.join( "," )+"\n";
                    firstStream << strFirst.join(",") + "\n";
                    strList.clear();
                    strFirst.clear();
                }

                strSecond << strID;
                strSecond << strMedians;
                secondStream << strSecond.join(",") + "\n";
                strSecond.clear();
            }

            allObjFile.close();
```

```
            firstExcelFile.close();
            secondExcelFile.close();
        }

        int total_value = 0;
        for(int i = 0; i < realIntensities.size(); i++)
        {
            total_value += realIntensities[i];
        }
        float final_median = (float)total_value /
(float)realIntensities.size();

        QString strCsvName = m_strSavePath + "/" + m_strResultName +
"(Error Bar).csv";
        QFile file(strCsvName);

        QString strThirdExcelName = m_strSavePath + "/" +
m_strResultName +
                "_third.csv";
        QFile thirdExcelFile(strThirdExcelName);
        thirdExcelFile.open(QFile::WriteOnly|QFile::Truncate);

        if (file.open(QFile::WriteOnly|QFile::Truncate))
        {
            QTextStream stream(&file);
            QStringList strList;

            QTextStream thirdStream(&thirdExcelFile);
            QStringList strThird;

            strList.clear();
            strList<<"Object ID"<<"Average"<<"Standard
Deviation"<<"Start Frame"<<"End
                Frame";
            stream << strList.join( "," )+"\n";
            strList.clear();

            strThird.clear();
            strThird<<"All 30%"<<"med 50%"<<"average"<<"s.d";
            thirdStream<<strThird.join(",") + "\n";
            strThird.clear();

            float sumOfDivations = 0;
            for(int i = 0; i < realIntensities.size(); i++)
```

```
            {
                float divation = (float)((float)realIntensities[i]
-

       final_median)*(float)((float)realIntensities[i] -
final_median);
                sumOfDivations += divation;
            }

            float final_div = sumOfDivations /
realIntensities.size();
            final_div = sqrtf(final_div);

            strList << "final value";
            QString strAverage = QString::number(final_median);
            strList<<strAverage;
            QString strDivation = QString::number(final_div);
            strList<<strDivation;
            stream << strList.join( "," )+"\n";
            strList.clear();

            for(int i = 0; i < allDatas.size(); i++)
            {
                int value1 = realIntensities[i*3];
                int value2 = realIntensities[i*3+1];
                int value3 = realIntensities[i*3+2];

                float median_value =
(float)(value1+value2+value3)/3;
                float divation = (float)(value1-
median_value)*(float)(value1-
                        median_value) + (float)(value2-
median_value)*(float)(value2-
                        median_value) + (float)(value3-
median_value)*(float)(value3-
                        median_value);
                divation = sqrtf(divation/3);

                QString strID = QString::number(i+1);
                strList << strID;
                strAverage = QString::number(median_value);
                strList << strAverage;
                strDivation = QString::number(divation);
                strList << strDivation;
```

```
                QString strStartFrame =
QString::number(allDatas[i][0].frameNum);
                strList << strStartFrame;
                QString strEndFrame =
QString::number(allDatas[i][allDatas[i].size()-
                        1].frameNum);
                strList << strEndFrame;
                stream << strList.join( "," )+"\n";
                strList.clear();

                strThird << strID;
                strThird << QString::number(value1);
                strThird << strAverage;
                strThird << strDivation;
                thirdStream << strThird.join(",") + "\n";
                strThird.clear();

                strThird << strID;
                strThird << QString::number(value2);
                strThird << "";
                strThird << "";
                thirdStream << strThird.join(",") + "\n";
                strThird.clear();

                strThird << strID;
                strThird << QString::number(value3);
                strThird << "";
                strThird << "";
                thirdStream << strThird.join(",") + "\n";
                strThird.clear();
            }

            file.close();
            thirdExcelFile.close();
        }
    }
}
```

Step 5 : Save Results

```
void ErrorbarThread::saveResultFiles()
{
    QString strObjID;
    QByteArray byteObjID;
    char* szObjID;
```

```
    int tempID = 0;
    Point befPt = Point(-1, -1);
    Point lastBefPt = Point(-1, -1);

    VideoCapture video;
    TIFFCapture tiff;
    Mat currentImg;
    int nFrameNum = 0;

    TIFF *tiffTrace, *tiffNoTrace, *tiffOrg;
    VideoWriter videoTrace, videoNoTrace;

    QString strOrgName = m_strSavePath + "/" + m_strResultName + "."
+ m_strSaveType;
    QByteArray byteOrgName = strOrgName.toLocal8Bit();
    char *szOrgName = byteOrgName.data();

    QString strTraceResultName = m_strSavePath + "/" +
m_strResultName + "(with
        trace)." + m_strSaveType;
    QByteArray byteTraceResultName =
strTraceResultName.toLocal8Bit();
    char *szTraceResultName = byteTraceResultName.data();

    QString strNoTraceResultName = m_strSavePath + "/" +
m_strResultName + "(without
        trace)." + m_strSaveType;
    QByteArray byteNoTraceResultName =
strNoTraceResultName.toLocal8Bit();
    char *szNoTraceResultName = byteNoTraceResultName.data();

    QByteArray byteName = m_strOpenFile.toLocal8Bit();
    char* szFileName = byteName.data();
    char* szFileType;
    int ch = '.';
    szFileType = strrchr( szFileName, ch ) + 1;
    QString strFileType(szFileType);

    bool bFirst = true;

    for(;;)
    {
        if(strFileType == "avi"||strFileType == "mp4")
        {
```

```
            if(bFirst)
            {
                video = VideoCapture(szFileName);
                bFirst = false;
            }

            video.operator >>(currentImg);
        }
        else
        {
            if(bFirst)
            {
                tiff = TIFFCapture(szFileName);
                bFirst = false;
            }

            tiff.operator >>(currentImg);
        }

        if(currentImg.empty())
            break;

        if(currentImg.type() == CV_8UC1)
            cvtColor(currentImg, currentImg, CV_GRAY2BGR);

        Mat withoutImg = currentImg.clone();
        Mat confirmImg = currentImg.clone();

        OBJECTS_ARRAY drawObjArray;
        cv::vector<int> currentIDs;

        for (int j = 0; j < (int)m_completeObjectsArray.size(); j++)
        {
            if (m_completeObjectsArray[j].frameNum == nFrameNum)
            {

currentIDs.push_back(m_completeObjectsArray[j].objectID);
                drawObjArray.push_back(m_completeObjectsArray[j]);
            }
        }

        for (int j = 0; j < (int)m_completeObjectsArray.size(); j++)
        {
            if (m_completeObjectsArray[j].frameNum <= nFrameNum)
```

```
            {
                for (int cnt = 0; cnt < currentIDs.size(); cnt++)
                {
                    if (currentIDs[cnt] ==
m_completeObjectsArray[j].objectID)
                    {

drawObjArray.push_back(m_completeObjectsArray[j]);
                    }
                }
            }
        }

        for (int k = 0; k < (int)drawObjArray.size(); k++)
        {
            if (tempID == drawObjArray[k].objectID)
            {
                if(befPt.x < drawObjArray[k].position.x)
                    line(confirmImg, befPt,
drawObjArray[k].position, colors[tempID%8],
                        2);
                lastBefPt = befPt;
                befPt = drawObjArray[k].position;
            }
            else
            {
                if(k != 0)
                {
                    strObjID = QString::number(tempID);
                    byteObjID = strObjID.toLocal8Bit();
                    szObjID = byteObjID.data();
                    cv::putText(confirmImg, szObjID, befPt,
                       CV_FONT_HERSHEY_COMPLEX_SMALL, 1.0,
colors[tempID%8], 2);
                }

                tempID = drawObjArray[k].objectID;
                line(confirmImg, drawObjArray[k].position,
drawObjArray[k].position,
                        colors[tempID%8], 2);
                befPt = drawObjArray[k].position;
            }

            if(k == (drawObjArray.size() - 1))
```

```
            {
                strObjID = QString::number(tempID);
                byteObjID = strObjID.toLocal8Bit();
                szObjID = byteObjID.data();
                cv::putText(confirmImg, szObjID,
drawObjArray[k].position,
                        CV_FONT_HERSHEY_COMPLEX_SMALL, 1.0,
colors[tempID%8], 2);
            }
        }

        for (int j = 0; j < (int)m_completeObjectsArray.size(); j++)
        {
            if (m_completeObjectsArray[j].frameNum == nFrameNum)
            {
                strObjID =
QString::number(m_completeObjectsArray[j].objectID);
                byteObjID = strObjID.toLocal8Bit();
                szObjID = byteObjID.data();
                cv::putText(withoutImg, szObjID,
m_completeObjectsArray[j].position,
                        CV_FONT_HERSHEY_COMPLEX_SMALL, 1.0,
                        colors[m_completeObjectsArray[j].objectID%8],
2);
            }
        }

        if(m_strSaveType == "avi")
        {
            if(nFrameNum == 0)
            {
                videoTrace.open(szTraceResultName, CV_FOURCC('X',
'V', 'I', 'D'),
                        m_nFps, confirmImg.size());
                videoNoTrace.open(szNoTraceResultName,
CV_FOURCC('X', 'V', 'I', 'D'),
                        m_nFps, withoutImg.size());
            }

            videoTrace.write(confirmImg);
            videoNoTrace.write(withoutImg);
        }
        else
        {
```

```
            if (nFrameNum == 0)
            {
                tiffOrg = TIFFOpen(szOrgName, "w");
                tiffTrace = TIFFOpen(szTraceResultName, "w");
                tiffNoTrace = TIFFOpen(szNoTraceResultName, "w");
            }

            TIFFSetField(tiffOrg, TIFFTAG_IMAGEWIDTH,
currentImg.cols);
            TIFFSetField(tiffOrg, TIFFTAG_IMAGELENGTH,
currentImg.rows);
            TIFFSetField(tiffOrg, TIFFTAG_BITSPERSAMPLE,8);
            TIFFSetField(tiffOrg, TIFFTAG_SAMPLESPERPIXEL,3);
            TIFFSetField(tiffOrg, TIFFTAG_PLANARCONFIG,
PLANARCONFIG_CONTIG);
            TIFFSetField(tiffOrg, TIFFTAG_PHOTOMETRIC,
PHOTOMETRIC_MINISBLACK);
            TIFFSetField(tiffOrg, TIFFTAG_ORIENTATION,
ORIENTATION_TOPLEFT);
            TIFFSetField(tiffOrg, TIFFTAG_RESOLUTIONUNIT,
RESUNIT_INCH);
            TIFFSetField(tiffOrg, TIFFTAG_XRESOLUTION, 100.0);
            TIFFSetField(tiffOrg, TIFFTAG_YRESOLUTION, 100.0);
            TIFFSetField(tiffOrg, TIFFTAG_SUBFILETYPE,
FILETYPE_PAGE);

            for (int j = 0; j < currentImg.rows; j++)
            {
                Mat rowImg = currentImg.row(j);
                TIFFWriteScanline(tiffOrg, rowImg.data, j, 0);
            }

            TIFFWriteDirectory(tiffOrg);

            TIFFSetField(tiffTrace, TIFFTAG_IMAGEWIDTH,
confirmImg.cols);
            TIFFSetField(tiffTrace, TIFFTAG_IMAGELENGTH,
confirmImg.rows);
            TIFFSetField(tiffTrace, TIFFTAG_BITSPERSAMPLE,8);
            TIFFSetField(tiffTrace, TIFFTAG_SAMPLESPERPIXEL,3);
            TIFFSetField(tiffTrace, TIFFTAG_PLANARCONFIG,
PLANARCONFIG_CONTIG);
            TIFFSetField(tiffTrace, TIFFTAG_PHOTOMETRIC,
PHOTOMETRIC_MINISBLACK);
```

```
            TIFFSetField(tiffTrace, TIFFTAG_ORIENTATION,
ORIENTATION_TOPLEFT);
            TIFFSetField(tiffTrace, TIFFTAG_RESOLUTIONUNIT,
RESUNIT_INCH);
            TIFFSetField(tiffTrace, TIFFTAG_XRESOLUTION, 100.0);
            TIFFSetField(tiffTrace, TIFFTAG_YRESOLUTION, 100.0);
            TIFFSetField(tiffTrace, TIFFTAG_SUBFILETYPE,
FILETYPE_PAGE);

            for (int j = 0; j < confirmImg.rows; j++)
            {
                Mat rowImg = confirmImg.row(j);
                TIFFWriteScanline(tiffTrace, rowImg.data, j, 0);
            }

            TIFFWriteDirectory(tiffTrace);

            TIFFSetField(tiffNoTrace, TIFFTAG_IMAGEWIDTH,
withoutImg.cols);
            TIFFSetField(tiffNoTrace, TIFFTAG_IMAGELENGTH,
withoutImg.rows);
            TIFFSetField(tiffNoTrace, TIFFTAG_BITSPERSAMPLE,8);
            TIFFSetField(tiffNoTrace, TIFFTAG_SAMPLESPERPIXEL,3);
            TIFFSetField(tiffNoTrace, TIFFTAG_PLANARCONFIG,
PLANARCONFIG_CONTIG);
            TIFFSetField(tiffNoTrace, TIFFTAG_PHOTOMETRIC,
PHOTOMETRIC_MINISBLACK);
            TIFFSetField(tiffNoTrace, TIFFTAG_ORIENTATION,
ORIENTATION_TOPLEFT);
            TIFFSetField(tiffNoTrace, TIFFTAG_RESOLUTIONUNIT,
RESUNIT_INCH);
            TIFFSetField(tiffNoTrace, TIFFTAG_XRESOLUTION, 100.0);
            TIFFSetField(tiffNoTrace, TIFFTAG_YRESOLUTION, 100.0);
            TIFFSetField(tiffNoTrace, TIFFTAG_SUBFILETYPE,
FILETYPE_PAGE);

            for (int j = 0; j < withoutImg.rows; j++)
            {
                Mat rowImg = withoutImg.row(j);
                TIFFWriteScanline(tiffNoTrace, rowImg.data, j, 0);
            }

            TIFFWriteDirectory(tiffNoTrace);
        }
```

```
        currentImg.release();
        nFrameNum++;
    }

    if(m_strSaveType != "avi")
    {
        TIFFClose(tiffNoTrace);
        TIFFClose(tiffTrace);
    }
}
```

Step 6 : Play Result

```
void PlayResultThread::start()
{
        QString strOrgFile, strWithFile, strWithoutFile;
        strOrgFile = m_strOpenPath + m_strOpenName + ".tif";
        strWithFile = m_strOpenPath + m_strOpenName + "(with
trace).tif";
        strWithoutFile = m_strOpenPath + m_strOpenName + "(without
trace).tif";

        QByteArray byteOrgFile = strOrgFile.toLocal8Bit();
        char *szOrgFile = byteOrgFile.data();

        QByteArray byteWithFile = strWithFile.toLocal8Bit();
        char *szWithFile = byteWithFile.data();

        QByteArray byteWithoutFile = strWithoutFile.toLocal8Bit();
        char *szWithoutFile = byteWithoutFile.data();

        VideoCapture orgVideo, withVideo, withoutVideo;
        TIFFCapture orgTiff, withTiff, withoutTiff;
        int nFrameNum = 0;

    for(;;)
    {
        if(g_bPlayStop)
            break;

        while (g_bPlayPause) {
            QThread::sleep(1);
        }
```

```
            Mat orgImg, withImg, withoutImg;
        if(nFrameNum == 0)
        {
            orgTiff = TIFFCapture(szOrgFile);
            withTiff = TIFFCapture(szWithFile);
            withoutTiff = TIFFCapture(szWithoutFile);
        }

        orgTiff.operator>>(orgImg);
        withTiff.operator>>(withImg);
        withoutTiff.operator>>(withoutImg);

                if(orgImg.empty())
                        break;

                if(orgImg.type() == CV_8UC1)
                        cvtColor(orgImg, orgImg, CV_GRAY2RGB);
                else
                        cvtColor(orgImg, orgImg, CV_BGR2RGB);

                if(withImg.type() == CV_8UC1)
                        cvtColor(withImg, withImg, CV_GRAY2RGB);
                else
                        cvtColor(withImg, withImg, CV_BGR2RGB);

                if(withoutImg.type() == CV_8UC1)
                        cvtColor(withoutImg, withoutImg,
CV_GRAY2RGB);
                else
                        cvtColor(withoutImg, withoutImg, CV_BGR2RGB);

                QString strFrameNumber = "frame number " +
QString::number(nFrameNum +
                        1) + " / " + QString::number(m_nFrameCnt);
                QByteArray byteFrameNumber =
strFrameNumber.toLocal8Bit();
                char *szFrameNumber = byteFrameNumber.data();
                cv::putText(orgImg, szFrameNumber,
cv::Point(orgImg.cols/2, orgImg.rows
                        - 10), CV_FONT_HERSHEY_COMPLEX_SMALL, 1.0,
cv::Scalar(255, 255,
                        255), 2);
                cv::putText(withImg, szFrameNumber,
cv::Point(withImg.cols/2,
```

```
                        withImg.rows - 10),
CV_FONT_HERSHEY_COMPLEX_SMALL, 1.0,
                        cv::Scalar(255, 255, 255), 2);
                cv::putText(withoutImg, szFrameNumber,
cv::Point(withoutImg.cols/2,
                        withoutImg.rows - 10),
CV_FONT_HERSHEY_COMPLEX_SMALL, 1.0,
                        cv::Scalar(255, 255, 255), 2);

        if(m_nTraceMode == PLAY_WITH_TRACE)
        {
            QImage org = QImage((const unsigned char*)(orgImg.data),
orgImg.cols,
                orgImg.rows, QImage::Format_RGB888);

m_pOrgView->setPixmap(QPixmap::fromImage(org.scaled(m_pOrgView->widt
h(),
                m_pOrgView->height())));
             QImage with = QImage((const unsigned
char*)(withImg.data), withImg.cols,
                withImg.rows, QImage::Format_RGB888);

           m_pResultView->setPixmap(QPixmap::fromImage(with.scaled(m
           _pResultView->width
                (), m_pResultView->height())));
        }
        else
        {
            QImage org = QImage((const unsigned char*)(orgImg.data),
orgImg.cols,
                orgImg.rows, QImage::Format_RGB888);

m_pOrgView->setPixmap(QPixmap::fromImage(org.scaled(m_pOrgView->widt
h(),
                m_pOrgView->height())));
             QImage without = QImage((const unsigned
char*)(withoutImg.data),
                withoutImg.cols, withoutImg.rows,
QImage::Format_RGB888);

           m_pResultView->setPixmap(QPixmap::fromImage(without.scale
           d(m_pResultView->wi
                dth(), m_pResultView->height())));
        }
```

```
        QThread::msleep(m_nSpeed);
                orgImg.release();

        nFrameNum++;
    }
    g_bPlayStart = false;
    g_bPlayStop = false;
    g_bPlayPause = false;
}
```

Step 7 : Show Analysis Result

```
void analysisDlg::drawRealGraph(int nID)
{
    if (m_graphData.size() == 0)
    {
        return;
    }

    ui->real_graph->clearGraphs();
    ui->real_graph->clearItems();

    std::vector<int> vecIntensity, vecFrameNumber, vecSpeed, vecPos;
    int startFrame, endFrame;
    BOOL bFlag = TRUE;

    for (int i = 0; i < (int)m_graphData.size(); i++)
    {
        if (nID == m_graphData[i].nID)
        {
            vecIntensity.push_back(m_graphData[i].nIntensity);
            vecFrameNumber.push_back(m_graphData[i].nStartFrame);
            vecPos.push_back(m_graphData[i].nPos);

            if(bFlag)
            {
                vecSpeed.push_back(0);
                bFlag = FALSE;
            }
            else
                vecSpeed.push_back(m_graphData[i].nSpeed);
        }
    }
```

```
    for(int i = 0; i < (int)m_showData.size(); i++)
    {
        if (nID == m_showData[i].nID)
        {
            startFrame = m_showData[i].nStartFrame;
            endFrame = m_showData[i].nEndFrame;
        }
    }

        int nSize_Y = 0;
    QVector<double> intensity_X(vecIntensity.size()),
intensity_Y(vecIntensity.size());
    for (int i = 0; i < (int)(vecIntensity.size()); ++i)
    {
        intensity_X[i] = (double)vecFrameNumber[i];
        intensity_Y[i] = (double)vecIntensity[i];

                if(nSize_Y < intensity_Y[i])
                        nSize_Y = intensity_Y[i];
    }
    ui->real_graph->addGraph();
    ui->real_graph->graph()->setData(intensity_X, intensity_Y);
        if(!m_bLine)

        ui->real_graph->graph()->setLineStyle(QCPGraph::LineStyle::l
sNone);

    QVector<double> speed_X(vecIntensity.size()),
speed_Y(vecIntensity.size());
    for (int i = 0; i < (int)(vecIntensity.size()); ++i)
    {
        speed_X[i] = (double)vecFrameNumber[i];
        speed_Y[i] = (double)vecSpeed[i];

                if(nSize_Y < speed_Y[i])
                        nSize_Y = speed_Y[i];
    }
    ui->real_graph->addGraph();
    ui->real_graph->graph()->setData(speed_X, speed_Y);
    QPen speed_pen;
    speed_pen.setColor(QColor(255,0,0));
    ui->real_graph->graph()->setPen(speed_pen);
        if(!m_bLine)
```

```
        ui->real_graph->graph()->setLineStyle(QCPGraph::LineStyle::l
sNone);

    QVector<double> pos_X(vecIntensity.size()),
pos_Y(vecIntensity.size());
    for (int i = 0; i < (int)(vecIntensity.size()); ++i)
    {
        pos_X[i] = (double)vecFrameNumber[i];
        pos_Y[i] = (double)vecPos[i];

                if(nSize_Y < pos_Y[i])
                        nSize_Y = pos_Y[i];
    }

    ui->real_graph->addGraph();
    ui->real_graph->graph()->setData(pos_X, pos_Y);
    QPen pos_pen;
    pos_pen.setColor(QColor(0,0,0));
    ui->real_graph->graph()->setPen(pos_pen);
        if(!m_bLine)

        ui->real_graph->graph()->setLineStyle(QCPGraph::LineStyle::l
sNone);

    ui->real_graph->xAxis->setLabel("Frame Number");
    ui->real_graph->yAxis->setLabel("Intensity, Speed, Position");

        float delta_x = (float)(endFrame - startFrame + 1)/100;
        float delta_y = (float)(nSize_Y + 10)/100;

        int step = vecIntensity.size() / 5 + 1;
        for(int i = 0; i < (int)(vecIntensity.size()); i++)
        {
                int nStep = i % step;

                if(nStep == 0)
                {
                        QPen pen;
                        pen.setColor(QColor(0,255,255));

                        QVector<double> intensity_top_x(2),
intensity_top_y(2);
```

```
                        intensity_top_x[0] = vecFrameNumber[i] -
delta_x*2;
                        intensity_top_x[1] = vecFrameNumber[i] +
delta_x*2;
                        intensity_top_y[0] = vecIntensity[i] +
delta_y*2;
                        intensity_top_y[1] = vecIntensity[i] +
delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(intensity_top_x,
                                intensity_top_y);
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> intensity_bottom_x(2),
intensity_bottom_y(2);
                        intensity_bottom_x[0] = vecFrameNumber[i] -
delta_x*2;
                        intensity_bottom_x[1] = vecFrameNumber[i] +
delta_x*2;
                        intensity_bottom_y[0] = vecIntensity[i] -
delta_y*2;
                        intensity_bottom_y[1] = vecIntensity[i] -
delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(intensity_bottom_x,
                                intensity_bottom_y);
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> intensity_left_x(2),
intensity_left_y(2);
                        intensity_left_x[0] = vecFrameNumber[i] -
delta_x*2;
                        intensity_left_x[1] = vecFrameNumber[i] -
delta_x*2;
                        intensity_left_y[0] = vecIntensity[i] +
delta_y*2;
                        intensity_left_y[1] = vecIntensity[i] -
delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(intensity_left_x,
                                intensity_left_y);
```

```
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> intensity_right_x(2),
intensity_right_y(2);
                        intensity_right_x[0] = vecFrameNumber[i] +
delta_x*2;
                        intensity_right_x[1] = vecFrameNumber[i] +
delta_x*2;
                        intensity_right_y[0] = vecIntensity[i] +
delta_y*2;
                        intensity_right_y[1] = vecIntensity[i] -
delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(intensity_right_x,
                                intensity_right_y);
                        ui->real_graph->graph()->setPen(pen);

                        // intensity value
                        QString strIntensity =
QString::number(vecIntensity[i]);
                        QCPItemText *intensityLabel = new
QCPItemText(ui->real_graph);
                        ui->real_graph->addItem(intensityLabel);


        intensityLabel->position->setType(QCPItemPosition::ptAxisRec
tRatio);
                        double intensity_x =
(double)(vecFrameNumber[i] - startFrame +
                                delta_x) / (double)(endFrame -
startFrame + 1);
                        double intensity_y = 1 -
(double)vecIntensity[i] /
                                (double)(nSize_Y + 10);

        intensityLabel->position->setCoords(intensity_x,
                                intensity_y);
                        intensityLabel->setText(strIntensity);

        intensityLabel->setFont(QFont(font().family(), 10));
                        intensityLabel->setPadding(QMargins(8, 0, 0,
0));
                        intensityLabel->setColor(QColor(0, 0, 255));
```

```
                        QVector<double> speed_top_x(2),
speed_top_y(2);
                        speed_top_x[0] = vecFrameNumber[i] -
delta_x*2;
                        speed_top_x[1] = vecFrameNumber[i] +
delta_x*2;
                        speed_top_y[0] = vecSpeed[i] + delta_y*2;
                        speed_top_y[1] = vecSpeed[i] + delta_y*2;
                        ui->real_graph->addGraph();
                        ui->real_graph->graph()->setData(speed_top_x,
speed_top_y);
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> speed_bottom_x(2),
speed_bottom_y(2);
                        speed_bottom_x[0] = vecFrameNumber[i] -
delta_x*2;
                        speed_bottom_x[1] = vecFrameNumber[i] +
delta_x*2;
                        speed_bottom_y[0] = vecSpeed[i] - delta_y*2;
                        speed_bottom_y[1] = vecSpeed[i] - delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(speed_bottom_x,
                                speed_bottom_y);
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> speed_left_x(2),
speed_left_y(2);
                        speed_left_x[0] = vecFrameNumber[i] -
delta_x*2;
                        speed_left_x[1] = vecFrameNumber[i] -
delta_x*2;
                        speed_left_y[0] = vecSpeed[i] + delta_y*2;
                        speed_left_y[1] = vecSpeed[i] - delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(speed_left_x,
speed_left_y);
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> speed_right_x(2),
speed_right_y(2);
```

```
                        speed_right_x[0] = vecFrameNumber[i] +
delta_x*2;
                        speed_right_x[1] = vecFrameNumber[i] +
delta_x*2;
                        speed_right_y[0] = vecSpeed[i] + delta_y*2;
                        speed_right_y[1] = vecSpeed[i] - delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(speed_right_x,
speed_right_y);
                        ui->real_graph->graph()->setPen(pen);

                        // speed value
                        QString strSpeed =
QString::number(vecSpeed[i]);
                        QCPItemText *speedLabel = new
QCPItemText(ui->real_graph);
                        ui->real_graph->addItem(speedLabel);


        speedLabel->position->setType(QCPItemPosition::ptAxisRectRat
io);
                        double speed_x = (double)(vecFrameNumber[i] -
startFrame +
                                delta_x) / (double)(endFrame -
startFrame + 1);
                        double speed_y = 1 - (double)vecSpeed[i] /
(double)(nSize_Y +
                                10);
                        speedLabel->position->setCoords(speed_x,
                                speed_y);
                        speedLabel->setText(strSpeed);
                        speedLabel->setFont(QFont(font().family(),
10));
                        speedLabel->setPadding(QMargins(8, 0, 0, 0));
                        speedLabel->setColor(QColor(255, 0, 0));

                        QVector<double> position_top_x(2),
position_top_y(2);
                        position_top_x[0] = vecFrameNumber[i] -
delta_x*2;
                        position_top_x[1] = vecFrameNumber[i] +
delta_x*2;
                        position_top_y[0] = vecPos[i] + delta_y*2;
```

```
                        position_top_y[1] = vecPos[i] + delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(position_top_x,
                                position_top_y);
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> position_bottom_x(2),
position_bottom_y(2);
                        position_bottom_x[0] = vecFrameNumber[i] -
delta_x*2;
                        position_bottom_x[1] = vecFrameNumber[i] +
delta_x*2;
                        position_bottom_y[0] = vecPos[i] - delta_y*2;
                        position_bottom_y[1] = vecPos[i] - delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(position_bottom_x,
                                position_bottom_y);
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> position_left_x(2),
position_left_y(2);
                        position_left_x[0] = vecFrameNumber[i] -
delta_x*2;
                        position_left_x[1] = vecFrameNumber[i] -
delta_x*2;
                        position_left_y[0] = vecPos[i] + delta_y*2;
                        position_left_y[1] = vecPos[i] - delta_y*2;
                        ui->real_graph->addGraph();

        ui->real_graph->graph()->setData(position_left_x,
                                position_left_y);
                        ui->real_graph->graph()->setPen(pen);

                        QVector<double> position_right_x(2),
position_right_y(2);
                        position_right_x[0] = vecFrameNumber[i] +
delta_x*2;
                        position_right_x[1] = vecFrameNumber[i] +
delta_x*2;
                        position_right_y[0] = vecPos[i] + delta_y*2;
                        position_right_y[1] = vecPos[i] - delta_y*2;
                        ui->real_graph->addGraph();
```

```
        ui->real_graph->graph()->setData(position_right_x,
                                position_right_y);
                        ui->real_graph->graph()->setPen(pen);

                        // position value
                        QString strPosition =
QString::number(vecPos[i]);
                        QCPItemText *positionLabel = new
QCPItemText(ui->real_graph);
                        ui->real_graph->addItem(positionLabel);


        positionLabel->position->setType(QCPItemPosition::ptAxisRect
Ratio);
                        double position_x =
(double)(vecFrameNumber[i] - startFrame +
                                delta_x) / (double)(endFrame -
startFrame + 1);
                        double position_y = 1 - (double)vecPos[i] /
(double)(nSize_Y +
                                10);

        positionLabel->position->setCoords(position_x,
                                position_y);
                        positionLabel->setText(strPosition);
                        positionLabel->setFont(QFont(font().family(),
10));
                        positionLabel->setPadding(QMargins(8, 0, 0,
0));
                        positionLabel->setColor(QColor(0, 0, 0));
                }
        }

    // set axes ranges, so we see all data
    ui->real_graph->xAxis->setRange(startFrame, endFrame + 1);
    ui->real_graph->yAxis->setRange(0, nSize_Y + 10);

    int n = endFrame - startFrame + 1;
    QVector<double> pTickY, pTickX;
    QVector<QString> piLabelsY, piLabelsX;
    for (int i = 0; i <= n; i = i + step)
    {
        pTickX.push_back(startFrame + i);
```

```
        piLabelsX.push_back(QString::number(startFrame + i));
    }

        for(int i = 0; i < nSize_Y + 10; i = i + 10)
        {
                pTickY.push_back(i);
                piLabelsY.push_back(QString::number(i));
        }

    ui->real_graph->yAxis->setAutoTicks(false);
    ui->real_graph->yAxis->setAutoTickLabels(false);
    ui->real_graph->yAxis->setTickVector(pTickY);
    ui->real_graph->yAxis->setTickVectorLabels(piLabelsY);
    ui->real_graph->xAxis->setAutoTicks(false);
    ui->real_graph->xAxis->setAutoTickLabels(false);
    ui->real_graph->xAxis->setTickVector(pTickX);
    ui->real_graph->xAxis->setTickVectorLabels(piLabelsX);

    QString strData = "Blob Number = " + QString::number(nID) + "
\nRed = Speed \nBlue = Real Intensity \nBlack = Path \nFrame Number
= from " + QString::number(startFrame) +
        " to " + QString::number(endFrame);
    QCPItemText *textLabel = new QCPItemText(ui->real_graph);
    ui->real_graph->addItem(textLabel);
    textLabel->position->setType(QCPItemPosition::ptAxisRectRatio);
    textLabel->setPositionAlignment(Qt::AlignRight|Qt::AlignBottom);
    textLabel->position->setCoords(0.7, 0.3); // lower right corner
of axis rect
    textLabel->setText(strData);
    textLabel->setTextAlignment(Qt::AlignLeft);
    textLabel->setFont(QFont(font().family(), 10));
    textLabel->setPadding(QMargins(8, 0, 0, 0));

    ui->real_graph->replot();
}

void analysisDlg::drawNormalizeGraph(int nID)
{
    if (m_graphData.size() == 0)
    {
        return;
    }

    ui->average_graph->clearGraphs();
```

```
ui->average_graph->clearItems();

std::vector<int> intensity, frameNumber, vecSpeed, vecPos;
int startFrame, endFrame;
BOOL bFlag = TRUE;
for (int i = 0; i < (int)m_graphData.size(); i++)
{
    if (nID == m_graphData[i].nID)
    {
        intensity.push_back(m_graphData[i].nIntensity);
        frameNumber.push_back(m_graphData[i].nStartFrame);
        vecPos.push_back(m_graphData[i].nPos);

        if(bFlag)
        {
            vecSpeed.push_back(0);
            bFlag = FALSE;
        }
        else
            vecSpeed.push_back(m_graphData[i].nSpeed);
    }
}

for(int i = 0; i < (int)m_showData.size(); i++)
{
    if (nID == m_showData[i].nID)
    {
        startFrame = m_showData[i].nStartFrame;
        endFrame = m_showData[i].nEndFrame;
    }
}

QVector<double> x(intensity.size()), y(intensity.size());
int normalizeIntensity = 0;
for (int i = 0; i < (int)(intensity.size()); ++i)
{
    normalizeIntensity += intensity[i];
}

if (intensity.size() == 0)
{
    normalizeIntensity = 0;
}
else
```

```
    {
        normalizeIntensity = normalizeIntensity / intensity.size();
    }

        int nSize_Y = 0;
    for (int i = 0; i < (int)(intensity.size()); ++i)
    {
        x[i] = (double)frameNumber[i];
        if((normalizeIntensity+5) > intensity[i])
            y[i] = (double)intensity[i];
        else
            y[i] = (double)normalizeIntensity - 5.0;

                if(nSize_Y < y[i])
                        nSize_Y = y[i];
    }

    // create graph and assign data to it:
    m_vecNormalize = y;
    ui->average_graph->addGraph();
    ui->average_graph->graph()->setData(x, y);
    QPen intensity_pen;
    intensity_pen.setColor(QColor(0,255,0));
    ui->average_graph->graph()->setPen(intensity_pen);
        if(!m_bLine)

        ui->average_graph->graph()->setLineStyle(QCPGraph::LineStyle
::lsNone);

    QVector<double> speed_X(intensity.size()),
speed_Y(intensity.size());
    for (int i = 0; i < (int)(intensity.size()); ++i)
    {
        speed_X[i] = (double)frameNumber[i];
        speed_Y[i] = (double)vecSpeed[i];

                if(nSize_Y < speed_Y[i])
                        nSize_Y = speed_Y[i];
    }
    ui->average_graph->addGraph();
    ui->average_graph->graph()->setData(speed_X, speed_Y);
    QPen speed_pen;
    speed_pen.setColor(QColor(255,0,0));
    ui->average_graph->graph()->setPen(speed_pen);
```

```
        if(!m_bLine)

        ui->average_graph->graph()->setLineStyle(QCPGraph::LineStyle
::lsNone);

    QVector<double> pos_X(intensity.size()),
pos_Y(intensity.size());
    for (int i = 0; i < (int)(intensity.size()); ++i)
    {
        pos_X[i] = (double)frameNumber[i];
        pos_Y[i] = (double)vecPos[i];

                if(nSize_Y < pos_Y[i])
                        nSize_Y = pos_Y[i];
    }
    ui->average_graph->addGraph();
    ui->average_graph->graph()->setData(pos_X, pos_Y);
    QPen pos_pen;
    pos_pen.setColor(QColor(0,0,0));
    ui->average_graph->graph()->setPen(pos_pen);
        if(!m_bLine)

        ui->average_graph->graph()->setLineStyle(QCPGraph::LineStyle
::lsNone);

    ui->average_graph->xAxis->setLabel("Frame Number");
    ui->average_graph->yAxis->setLabel("Intensity");

        float delta_x = (float)(endFrame - startFrame + 1)/100;
        float delta_y = (float)(nSize_Y + 10)/100;

        int step = intensity.size() / 5 + 1;
        for(int i = 0; i < (int)(intensity.size()); i++)
        {
                int nStep = i % step;
                if(nStep == 0)
                {
                        QPen pen;
                        pen.setColor(QColor(255,0,255));

                        QVector<double> intensity_top_x(2),
intensity_top_y(2);
                        intensity_top_x[0] = frameNumber[i] -
delta_x*2;
```

```
                        intensity_top_x[1] = frameNumber[i] +
delta_x*2;
                        intensity_top_y[0] = intensity[i] +
delta_y*2;
                        intensity_top_y[1] = intensity[i] +
delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(intensity_top_x,
                                intensity_top_y);
                        ui->average_graph->graph()->setPen(pen);

                        QVector<double> intensity_bottom_x(2),
intensity_bottom_y(2);
                        intensity_bottom_x[0] = frameNumber[i] -
delta_x*2;
                        intensity_bottom_x[1] = frameNumber[i] +
delta_x*2;
                        intensity_bottom_y[0] = intensity[i] -
delta_y*2;
                        intensity_bottom_y[1] = intensity[i] -
delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(intensity_bottom_x,
                                intensity_bottom_y);
                        ui->average_graph->graph()->setPen(pen);

                        QVector<double> intensity_left_x(2),
intensity_left_y(2);
                        intensity_left_x[0] = frameNumber[i] -
delta_x*2;
                        intensity_left_x[1] = frameNumber[i] -
delta_x*2;
                        intensity_left_y[0] = intensity[i] +
delta_y*2;
                        intensity_left_y[1] = intensity[i] -
delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(intensity_left_x,
                                intensity_left_y);
                        ui->average_graph->graph()->setPen(pen);
```

```
                        QVector<double> intensity_right_x(2),
intensity_right_y(2);
                        intensity_right_x[0] = frameNumber[i] +
delta_x*2;
                        intensity_right_x[1] = frameNumber[i] +
delta_x*2;
                        intensity_right_y[0] = intensity[i] +
delta_y*2;
                        intensity_right_y[1] = intensity[i] -
delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(intensity_right_x,
                                intensity_right_y);
                        ui->average_graph->graph()->setPen(pen);

                        // intensity value
                        QString strIntensity =
QString::number(intensity[i]);
                        QCPItemText *intensityLabel = new
                                QCPItemText(ui->average_graph);
                        ui->average_graph->addItem(intensityLabel);


        intensityLabel->position->setType(QCPItemPosition::ptAxisRec
tRatio);
                        double intensity_x = (double)(frameNumber[i]
- startFrame +
                                delta_x) / (double)(endFrame -
startFrame + 1);
                        double intensity_y = 1 - (double)intensity[i]
/ (double)(nSize_Y
                                + 10);

        intensityLabel->position->setCoords(intensity_x,
                                intensity_y);
                        intensityLabel->setText(strIntensity);

        intensityLabel->setFont(QFont(font().family(), 10));
                        intensityLabel->setPadding(QMargins(8, 0, 0,
0));
                        intensityLabel->setColor(QColor(0,255,0));
```

```
                        QVector<double> speed_top_x(2),
speed_top_y(2);
                        speed_top_x[0] = frameNumber[i] - delta_x*2;
                        speed_top_x[1] = frameNumber[i] + delta_x*2;
                        speed_top_y[0] = vecSpeed[i] + delta_y*2;
                        speed_top_y[1] = vecSpeed[i] + delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(speed_top_x,
speed_top_y);
                        ui->average_graph->graph()->setPen(pen);

                        QVector<double> speed_bottom_x(2),
speed_bottom_y(2);
                        speed_bottom_x[0] = frameNumber[i] -
delta_x*2;
                        speed_bottom_x[1] = frameNumber[i] +
delta_x*2;
                        speed_bottom_y[0] = vecSpeed[i] - delta_y*2;
                        speed_bottom_y[1] = vecSpeed[i] - delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(speed_bottom_x,
                                speed_bottom_y);
                        ui->average_graph->graph()->setPen(pen);

                        QVector<double> speed_left_x(2),
speed_left_y(2);
                        speed_left_x[0] = frameNumber[i] - delta_x*2;
                        speed_left_x[1] = frameNumber[i] - delta_x*2;
                        speed_left_y[0] = vecSpeed[i] + delta_y*2;
                        speed_left_y[1] = vecSpeed[i] - delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(speed_left_x,
speed_left_y);
                        ui->average_graph->graph()->setPen(pen);

                        QVector<double> speed_right_x(2),
speed_right_y(2);
                        speed_right_x[0] = frameNumber[i] +
delta_x*2;
                        speed_right_x[1] = frameNumber[i] +
delta_x*2;
```

```
                        speed_right_y[0] = vecSpeed[i] + delta_y*2;
                        speed_right_y[1] = vecSpeed[i] - delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(speed_right_x,
                                speed_right_y);
                        ui->average_graph->graph()->setPen(pen);

                        // speed value
                        QString strSpeed =
QString::number(vecSpeed[i]);
                        QCPItemText *speedLabel = new
QCPItemText(ui->average_graph);
                        ui->average_graph->addItem(speedLabel);


        speedLabel->position->setType(QCPItemPosition::ptAxisRectRat
io);
                        double speed_x = (double)(frameNumber[i] -
startFrame + delta_x)
                                / (double)(endFrame - startFrame +
1);
                        double speed_y = 1 - (double)vecSpeed[i] /
(double)(nSize_Y +
                                10);
                        speedLabel->position->setCoords(speed_x,
                                speed_y);
                        speedLabel->setText(strSpeed);
                        speedLabel->setFont(QFont(font().family(),
10));
                        speedLabel->setPadding(QMargins(8, 0, 0, 0));
                        speedLabel->setColor(QColor(255, 0, 0));

                        QVector<double> position_top_x(2),
position_top_y(2);
                        position_top_x[0] = frameNumber[i] -
delta_x*2;
                        position_top_x[1] = frameNumber[i] +
delta_x*2;
                        position_top_y[0] = vecPos[i] + delta_y*2;
                        position_top_y[1] = vecPos[i] + delta_y*2;
                        ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(position_top_x,
```

```
                        position_top_y);
                ui->average_graph->graph()->setPen(pen);

                QVector<double> position_bottom_x(2),
position_bottom_y(2);
                position_bottom_x[0] = frameNumber[i] -
delta_x*2;
                position_bottom_x[1] = frameNumber[i] +
delta_x*2;
                position_bottom_y[0] = vecPos[i] - delta_y*2;
                position_bottom_y[1] = vecPos[i] - delta_y*2;
                ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(position_bottom_x,
                        position_bottom_y);
                ui->average_graph->graph()->setPen(pen);

                QVector<double> position_left_x(2),
position_left_y(2);
                position_left_x[0] = frameNumber[i] -
delta_x*2;
                position_left_x[1] = frameNumber[i] -
delta_x*2;
                position_left_y[0] = vecPos[i] + delta_y*2;
                position_left_y[1] = vecPos[i] - delta_y*2;
                ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(position_left_x,
                        position_left_y);
                ui->average_graph->graph()->setPen(pen);

                QVector<double> position_right_x(2),
position_right_y(2);
                position_right_x[0] = frameNumber[i] +
delta_x*2;
                position_right_x[1] = frameNumber[i] +
delta_x*2;
                position_right_y[0] = vecPos[i] + delta_y*2;
                position_right_y[1] = vecPos[i] - delta_y*2;
                ui->average_graph->addGraph();

        ui->average_graph->graph()->setData(position_right_x,
                        position_right_y);
                ui->average_graph->graph()->setPen(pen);
```

```
                        // position value
                        QString strPosition =
QString::number(vecPos[i]);
                        QCPItemText *positionLabel = new
QCPItemText(ui->average_graph);
                        ui->average_graph->addItem(positionLabel);


        positionLabel->position->setType(QCPItemPosition::ptAxisRect
Ratio);
                        double position_x = (double)(frameNumber[i] -
startFrame +
                                delta_x) / (double)(endFrame -
startFrame + 1);
                        double position_y = 1 - (double)vecPos[i] /
(double)(nSize_Y +
                                10);

        positionLabel->position->setCoords(position_x,
position_y);
                        positionLabel->setText(strPosition);
                        positionLabel->setFont(QFont(font().family(),
10));
                        positionLabel->setPadding(QMargins(8, 0, 0,
0));
                        positionLabel->setColor(QColor(255, 0, 0));
                }
        }

    // set axes ranges, so we see all data:
    ui->average_graph->xAxis->setRange(startFrame, endFrame + 1);
    ui->average_graph->yAxis->setRange(0, nSize_Y + 10);

    int n = endFrame - startFrame + 1;
    QVector<double> pTickY, pTickX;
    QVector<QString> piLabelsY, piLabelsX;
    for (int i = 0; i <= n; i = i = i + step)
    {
        pTickX.push_back(startFrame+i);
        piLabelsX.push_back(QString::number(startFrame+i));
    }

        for(int i = 0; i < nSize_Y + 10; i = i + 10)
```

```
        {
                pTickY.push_back(i);
                piLabelsY.push_back(QString::number(i));
        }

    ui->average_graph->yAxis->setAutoTicks(false);
    ui->average_graph->yAxis->setAutoTickLabels(false);
    ui->average_graph->yAxis->setTickVector(pTickY);
    ui->average_graph->yAxis->setTickVectorLabels(piLabelsY);
    ui->average_graph->xAxis->setAutoTicks(false);
    ui->average_graph->xAxis->setAutoTickLabels(false);
    ui->average_graph->xAxis->setTickVector(pTickX);
    ui->average_graph->xAxis->setTickVectorLabels(piLabelsX);

    QString strData = "Blob Number = " + QString::number(nID) + "
\nRed = Speed \nGreen = Normalize Intensity \nBlack = Path \nFrame
Number = from " + QString::number(startFrame) +
        " from " + QString::number(endFrame);
    QCPItemText *textLabel = new QCPItemText(ui->average_graph);
    ui->average_graph->addItem(textLabel);
    textLabel->position->setType(QCPItemPosition::ptAxisRectRatio);
    textLabel->setPositionAlignment(Qt::AlignRight|Qt::AlignBottom);
    textLabel->position->setCoords(0.7, 0.3); // lower right corner
of axis rect
    textLabel->setText(strData);
    textLabel->setTextAlignment(Qt::AlignLeft);
    textLabel->setFont(QFont(font().family(), 10));
    textLabel->setPadding(QMargins(8, 0, 0, 0));

    ui->average_graph->replot();
}
```

Step 8 : Show ErrorBar

```
void errorbarDlg::drawErrorBar()
{
        QVector<ERRORBAR> drawErrorBarData;
        vector<int> considerIdx;
        for(int i = 0; i < m_nErrorBar.size(); i++)
        {
                bool bConsider = false;
                for(int k = 0; k < considerIdx.size(); k++)
                {
                        if(i == considerIdx[k])
                        {
```

```
                                bConsider = true;
                                break;
                        }
                }

                if(bConsider)
                        continue;

                ERRORBAR errorbar;
                errorbar = m_nErrorBar[i];

                if(i < m_nErrorBar.size()-1)
                {
                        for(int j = i + 1; j < m_nErrorBar.size();
j++)
                        {
                                for(int l = 0; l <
considerIdx.size(); l++)
                                {
                                        if(j == considerIdx[l])
                                        {
                                                bConsider = true;
                                                break;
                                        }
                                }

                                if(bConsider)
                                {
                                        if(j == m_nErrorBar.size()-1)
                                        {

        drawErrorBarData.push_back(errorbar);
                                        }

                                        continue;
                                }

                                if(m_nErrorBar[i].inputName ==
m_nErrorBar[j].inputName)
                                {
                                        considerIdx.push_back(j);
                                        errorbar.average =

        (errorbar.average+m_nErrorBar[j].average)/2;
```

```
                                        errorbar.divation =
        (errorbar.divation+m_nErrorBar[j].divation)/2;
                                }

                                if(j == m_nErrorBar.size()-1)
                                {
        drawErrorBarData.push_back(errorbar);
                                }
                        }
                }
                else
                {
                        drawErrorBarData.push_back(errorbar);
                        considerIdx.push_back(i);
                }
        }

    // generate some data:
    m_bManual = false;
    m_vecGraphData.clear();
        QPen pos_pen;
        pos_pen.setColor(QColor(255,0,0));

    int x_range = 0;
        int y_range = 0;
    EQUATION_VARIOUS various;
    QVector<double> error_x(drawErrorBarData.size()),
error_y(drawErrorBarData.size());
    for (int i = 0; i < drawErrorBarData.size(); i++)
    {
        error_x[i] = drawErrorBarData[i].inputName;
        error_y[i] = drawErrorBarData[i].average;

        QVector<double> lineX(2), lineY(2);
        lineX[0] = error_x[i];
        lineX[1] = error_x[i];
        lineY[0] = error_y[i] - drawErrorBarData[i].divation;
        lineY[1] = error_y[i] + drawErrorBarData[i].divation;
        ui->graph_view->addGraph();
        ui->graph_view->graph()->setData(lineX, lineY);

        if(x_range < error_x[i])
```

```
            x_range = error_x[i];

                if(y_range < lineY[1])
                        y_range = lineY[1];
    }

    if(m_bConnect)
    {
        ui->graph_view->addGraph();
        ui->graph_view->graph()->setData(error_x, error_y);
    }

        float delta_x = (float)x_range / 50;
        float delta_y = (float)y_range/50;
        for (int i = 0; i < drawErrorBarData.size(); i++)
        {
                QVector<double> max_x(2), max_y(2);
                max_x[0] = error_x[i] - delta_x*2;
                max_x[1] = error_x[i] + delta_x*2;
                max_y[0] = error_y[i] +
drawErrorBarData[i].divation;
                max_y[1] = error_y[i] +
drawErrorBarData[i].divation;
                ui->graph_view->addGraph();
                ui->graph_view->graph()->setData(max_x, max_y);

                QVector<double> min_x(2), min_y(2);
                min_x[0] = error_x[i] - delta_x*2;
                min_x[1] = error_x[i] + delta_x*2;
                min_y[0] = error_y[i] -
drawErrorBarData[i].divation;
                min_y[1] = error_y[i] -
drawErrorBarData[i].divation;
                ui->graph_view->addGraph();
                ui->graph_view->graph()->setData(min_x, min_y);

                various.x_pos = error_x[i];
                various.y_pos = error_y[i];
                various.y_pos_max = max_y[0];
                various.y_pos_min = min_y[0];
                m_vecGraphData.push_back(various);

                QVector<double> rectleft_x(2), rectleft_y(2);
                rectleft_x[0] = error_x[i] - delta_x;
```

```
                rectleft_x[1] = error_x[i] - delta_x;
                rectleft_y[0] = error_y[i] - delta_y;
                rectleft_y[1] = error_y[i] + delta_y;
                ui->graph_view->addGraph();
                ui->graph_view->graph()->setData(rectleft_x,
rectleft_y);
                ui->graph_view->graph()->setPen(pos_pen);

                QVector<double> rectbottom_x(2), rectbottom_y(2);
                rectbottom_x[0] = error_x[i] - delta_x;
                rectbottom_x[1] = error_x[i] + delta_x;
                rectbottom_y[0] = error_y[i] - delta_y;
                rectbottom_y[1] = error_y[i] - delta_y;
                ui->graph_view->addGraph();
                ui->graph_view->graph()->setData(rectbottom_x,
rectbottom_y);
                ui->graph_view->graph()->setPen(pos_pen);

                QVector<double> rectright_x(2), rectright_y(2);
                rectright_x[0] = error_x[i] + delta_x;
                rectright_x[1] = error_x[i] + delta_x;
                rectright_y[0] = error_y[i] - delta_y;
                rectright_y[1] = error_y[i] + delta_y;
                ui->graph_view->addGraph();
                ui->graph_view->graph()->setData(rectright_x,
rectright_y);
                ui->graph_view->graph()->setPen(pos_pen);

                QVector<double> recttop_x(2), recttop_y(2);
                recttop_x[0] = error_x[i] - delta_x;
                recttop_x[1] = error_x[i] + delta_x;
                recttop_y[0] = error_y[i] + delta_y;
                recttop_y[1] = error_y[i] + delta_y;
                ui->graph_view->addGraph();
                ui->graph_view->graph()->setData(recttop_x,
recttop_y);
                ui->graph_view->graph()->setPen(pos_pen);
        }

    calcEquation();

    QVector<double> equation_x(x_range + 10), equation_y(x_range +
10);
        int y_min = 0;
```

```
    for(int i = 0; i < x_range+10; ++i)
    {
        equation_x[i] = i;
        equation_y[i] = m_A * (double)i + m_B;

                if(y_min > equation_y[i])
                        y_min = equation_y[i];
    }
    ui->graph_view->addGraph();
    ui->graph_view->graph()->setData(equation_x, equation_y);
    QPen pen;
    pen.setColor(QColor(0,255,0));
    ui->graph_view->graph()->setPen(pen);

    QString strEquation;
        if(m_A == 0)
                strEquation = "y = ";
        else if(m_A == 1)
                strEquation = "y = x ";
        else
                strEquation = "y = " + QString::number(m_A) + " * x
";

        if(m_B < 0)
                strEquation = strEquation + QString::number(m_B);
        else if(m_B > 0)
        {
                if(m_A == 0)
                        strEquation = strEquation +
QString::number(m_B);
                else
                        strEquation = strEquation + " + " +
QString::number(m_B);
        }
        else
        {
                if(m_A == 0)
                        strEquation = "y = 0";
        }

        double ss_reg = 0, ss_tot = 0;
        for(int i = 0; i < m_vecGraphData.size(); i++)
        {
                ss_tot += m_vecGraphData[i].y_pos;
```

```
                double current_y = m_A *
(double)m_vecGraphData[i].x_pos + m_B;
                double delta_y = m_vecGraphData[i].y_pos -
current_y;
                double square_delta = delta_y * delta_y;
                ss_reg += square_delta;
        }
        ss_tot = ss_tot / (double)m_vecGraphData.size();
        double r_2 = 1 - ss_reg / ss_tot;
        QString strR2 = "R^2 = " + QString::number(r_2);
        //QString str_errorbar_equation = strEquation + "\n" +
strR2;

    QCPItemText *textLabel = new QCPItemText(ui->graph_view);
    ui->graph_view->addItem(textLabel);
    textLabel->position->setType(QCPItemPosition::ptAxisRectRatio);
    textLabel->setPositionAlignment(Qt::AlignRight|Qt::AlignBottom);
    textLabel->position->setCoords(0.8, 0.1); // lower right corner
of axis rect
    textLabel->setText(strEquation);
    textLabel->setTextAlignment(Qt::AlignLeft);
    textLabel->setFont(QFont(font().family(), 20));
    textLabel->setPadding(QMargins(8, 0, 0, 0));

        QCPItemText *textLabel1 = new QCPItemText(ui->graph_view);
        ui->graph_view->addItem(textLabel1);
        textLabel1->position->setType(QCPItemPosition::ptAxisRectRat
io);
        textLabel1->setPositionAlignment(Qt::AlignRight|Qt::AlignBot
tom);
        textLabel1->position->setCoords(0.8, 0.2); // lower right
corner of axis rect
        textLabel1->setText(strR2);
        textLabel1->setTextAlignment(Qt::AlignLeft);
        textLabel1->setFont(QFont(font().family(), 20));
        textLabel1->setPadding(QMargins(8, 0, 0, 0));

    // give the axes some labels:
    ui->graph_view->xAxis->setLabel("Input Name");
    ui->graph_view->yAxis->setLabel("Intensity");

    // set axes ranges, so we see all data:
    ui->graph_view->xAxis->setRange(0, x_range + 10);
    ui->graph_view->yAxis->setRange(y_min, y_range + 10);
```

```
    int nx = (x_range + 10) / 10;
        int ny = (y_range + 10) / 10;
    QVector<double> pTickY, pTickX;
    QVector<QString> piLabelsY, piLabelsX;

        for (int i = 0; i <= nx; i++)
    {
        pTickX.push_back(i * 10);
        piLabelsX.push_back(QString::number(i * 10));
    }

        for(int i = y_min; i <= ny; i++)
        {
                pTickY.push_back(i * 10);
                piLabelsY.push_back(QString::number(i * 10));
        }

        if(m_bShowValue)
        {
                for(int i = 0; i < drawErrorBarData.size(); i++)
                {
                        // current value
                        QString strValue =
QString::number(error_y[i]);
                        QCPItemText *valueLabel = new
QCPItemText(ui->graph_view);
                        ui->graph_view->addItem(valueLabel);


        valueLabel->position->setType(QCPItemPosition::ptAxisRectRat
io);
                        double pos_x = (double)error_x[i] /
(double)(x_range + 10);
                        double pos_y = 1 - (double)(error_y[i] -
y_min) /
                                (double)(y_range - y_min + 10);
                        valueLabel->position->setCoords(pos_x,
pos_y);
                        valueLabel->setText(strValue);
                        valueLabel->setFont(QFont(font().family(),
10));
                        valueLabel->setPadding(QMargins(8, 0, 0, 0));
```

```
                        // div value
                        QString strDiv =
QString::number(drawErrorBarData[i].divation);
                        QCPItemText *divLabel = new
QCPItemText(ui->graph_view);
                        ui->graph_view->addItem(divLabel);


        divLabel->position->setType(QCPItemPosition::ptAxisRectRatio
);
                        double divpos_x = (double)error_x[i] /
(double)(x_range + 10);
                        double divpos_y = 1 - (double)(error_y[i] -
                                drawErrorBarData[i].divation/2 -
y_min) /
                                (double)(y_range - y_min + 10);
                        divLabel->position->setCoords(divpos_x,
divpos_y);
                        divLabel->setText(strDiv);
                        divLabel->setFont(QFont(font().family(),
10));
                        divLabel->setPadding(QMargins(8, 0, 0, 0));
                        divLabel->setColor(QColor(255, 0, 0));
                }
        }

    ui->graph_view->yAxis->setAutoTicks(false);
    ui->graph_view->yAxis->setAutoTickLabels(false);
    ui->graph_view->yAxis->setTickVector(pTickY);
    ui->graph_view->yAxis->setTickVectorLabels(piLabelsY);
    ui->graph_view->xAxis->setAutoTicks(false);
    ui->graph_view->xAxis->setAutoTickLabels(false);
    ui->graph_view->xAxis->setTickVector(pTickX);
    ui->graph_view->xAxis->setTickVectorLabels(piLabelsX);

    ui->graph_view->replot();
}

void errorbarDlg::calcEquation()
{
    if (m_vecGraphData.size() == 1)
    {
        m_A = 0;
        m_B = m_vecGraphData[0].y_pos;
```

```
    }
    else if (m_vecGraphData.size() == 2)
    {
        m_A = (m_vecGraphData[1].y_pos - m_vecGraphData[0].y_pos) /
               (m_vecGraphData[1].x_pos - m_vecGraphData[0].x_pos);
        m_B = m_vecGraphData[0].y_pos - m_A *
m_vecGraphData[0].x_pos;
    }
    else
    {
        bool bTrue = false;
        for (double y0 = m_vecGraphData[0].y_pos_min; y0 <=
              m_vecGraphData[0].y_pos_max; y0 = y0 + 0.1)
        {
            for (double y1 = m_vecGraphData[1].y_pos_min; y1 <=
              m_vecGraphData[1].y_pos_max; y1 = y1 + 0.1)
            {
                m_A = (y1 - y0) / (m_vecGraphData[1].x_pos -
m_vecGraphData[0].x_pos);
                m_B = y0 - m_A * m_vecGraphData[0].x_pos;
                bool bTrueTemp = true;

                for (int variousCnt = 0; variousCnt <
(m_vecGraphData.size() - 2);
                      variousCnt++)
                {
                    double tempY =
m_A*m_vecGraphData[variousCnt+2].x_pos + m_B;
                    if((tempY >
m_vecGraphData[variousCnt+2].y_pos_max)||(tempY <
                       m_vecGraphData[variousCnt+2].y_pos_min))
                        bTrueTemp = false;

                    if(!bTrueTemp)
                        break;
                }

                if(bTrueTemp)
                {
                    bTrue = true;
                    break;
                }
            }
```

```
            if(bTrue)
                break;
        }

        if(!bTrue)
        {
            m_A = 0;
            m_B = 0;
        }
    }

        int a = m_A * 10;
        int b = m_B * 10;

        m_A = (double)a / 10;
        m_B = (double)b / 10;
}
```