Controlling by Showing: i-Mimic: A Video-based Method to Control Robotic Arms

Debarati B. Chakraborty* · Mukesh Sharma · Bhaskar Vijay

Received: date / Accepted: date

Abstract A novel concept of vision-based intelligent control of robotic arms is developed here in this work. This work enables the controlling of robotic arm motion only with visual input, that is, controlling by showing the videos of correct movements. This work can broadly be sub-divided into two segments. The first part of this work is to develop an unsupervised vision-based method to control robotic arms in 2-D plane, and the second one is with deep CNN in the same task in 3-D plane. The first method is unsupervised, where our aim is to perform mimicking of human arm motion in real-time by a manipulator. We developed a network, namely the vision-to-motion optical network (DON). Given the input of a video stream containing the hand movements of human on the DON, the velocity and torque information of the hand movements shown in the video would be generated as the output. The output information of the DON is then fed to the robotic arm by enabling it to generate motion according to the real hand videos. The method has been tested on both live-stream video feeds as well as on recorded video obtained from a monocular camera even by intelligently predicting the trajectory of the human hand hand when it gets occluded. This is why the mimicry of the arm incorporates some intelligence to it and becomes an intelligent mimic (i- mimic). Furthermore, to enhance the performance of DON and make it applicable to mimic multi-joint movements with n-link manipulator, a deep network, namely, convolutional neural network (CNN) has been used along with a refiner network as the predecessor of DON. Refiner network has been used to overcome the limitations of inadequate labelled data. Both the proposed methods are validated with off-line as well as with on-line video datasets in real-time. The entire methodology is validated with real-time 1-link and simulated n-link manipulators (an arm with n number of different joints) along with suitable comparisons.

Keywords Visio-based robot control · video processing · deep network · convolutional neural network · robotic arm control

D. B. Chakraborty*
School of Computer Science and Engineering, VIT-AP University, Amaravati, India
E-mail: debarati.earth@gmail.com
OCRID: 0000-0002-3131-012X

M. Sharma and B. Vijay
Department of Mechanical Engineering, Indian Institute of Technology, Jodhpur, India
E-mail: sharma.15@iitj.ac.in and vijay.2@iitj.ac.in

1 Introduction

Robotic arms are used to perform mechanical tasks in industries over decades. It was mainly used for performing repetitive tasks in the industries to cut down the labor cost [2, 1]. Normally, robotic arms are quite complex with five or more degree of freedom as it aims to perform human tasks. Recently, the application of robotic arms in conducting domestic work has drawn attention. Controlling of these robotic arms to perform different tasks is still a major issue to be addressed. Here, in this work we have defined a new concept of controlling robotic arms only with visual information, that is, the motion of different parts of the robotic arms could be controlled by providing according human hand movement videos as the input to those arms.

The entire method could be subdivided into two parts. In the first part of this work we aim to determine a simple solution for unsupervised controlling the robotic arm only with visual information. Here we aim to deal with the issues of i) unavailability of sufficient training datasets, ii) domain adaptation and iii) economic cost. On the way to search for a solution of the two initial issues we concluded that the teaching/ training part should be removed. However, how could it be controlled then? 'Mimic' is the solution that stroked in our minds. The controlling could only be achieved by showing the arm the desired movement and making it enabled to follow it. Visual mimic-based controlling of 1-link and n-linked robotic arms (an arm with n number of different joints) is the primary contribution of this work. The mechanism of this set-up is quite simple and the 1-link manipulator is developed only by the authors. Either the recorded or real-time video could be shown to the arm to achieve the control. It should be noted that the real-time testing of this 'mimic' with robotic arm is in a very primary level where the arm is a simple 1-link robotic manipulator with a degree of freedom of 120 degrees. The rest of the tests are conducted in the form of simulation, where the simulated arm is able to mimic the motion of a single joint (solder or elbow) or real hand. Another contribution in this part of the work is the development of the vision-to-motion optical network (DON) to process the optical flow information of the input video and to convert it into the physical force to be fed to the robotic arm. The proposed DON is different from the existing deep networks in the following manners: i) it does not require any labeled data set or manual intervention, ii) it does neither require background estimation or a large number of input frames for training, iii) the functioning of the intermediate layers are simple which enables computational gain iv) all the intermediate layers are not active simultaneously; some layer gets activated depending on the values of the outputs of its previous layer v) the single network can perform both estimation and prediction and vi) it produces torque and angular velocity as the output. A step-wise illustration of this part of the work is shown here in Fig. 1 in the form of a block diagram. The detail mechanism of DON has been explained in Sec. 3.

In the second part of the work, we focus on controlling a more real robot arm, that is, an arm with multiple joints, or n-link manipulator. To reach a solution of such a problem, we incorporated deep neural network with CNN and Refiner Networks as the predecessors to DON. The block step-wise details for the multi-joint i-mimic have been shown in Fig. 2. Addition of these networks to DON enabled proper identification of hand joints in video frames, which could be further processed by DON for i-mimic with n-link manipulator. The details and underlying architecture of this method have been described in Sec. 4 To deal with unavailability of large set of labelled data, we used a CNN network whose outputs are further fed into the refinement network that smoothens the final output and enables to interpolate to a larger range. The deep CNN is proven to be less effective with 2-D n-link manipulators, but, it performs better in 3-D plane with n-link manipulators.

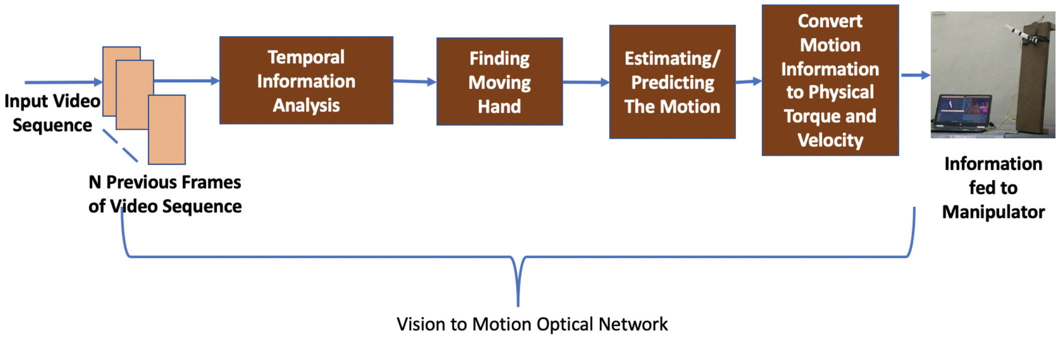

Input Video Sequence
N Previous Frames of Video Sequence
Temporal Information Analysis
Finding Moving Hand
Estimating/ Predicting The Motion
Convert Motion Information to Physical Torque and Velocity
Information fed to Manipulator
Vision to Motion Optical Network

Fig. 1: Block Diagram Representation of i-mimic With Single Joint Manipulator

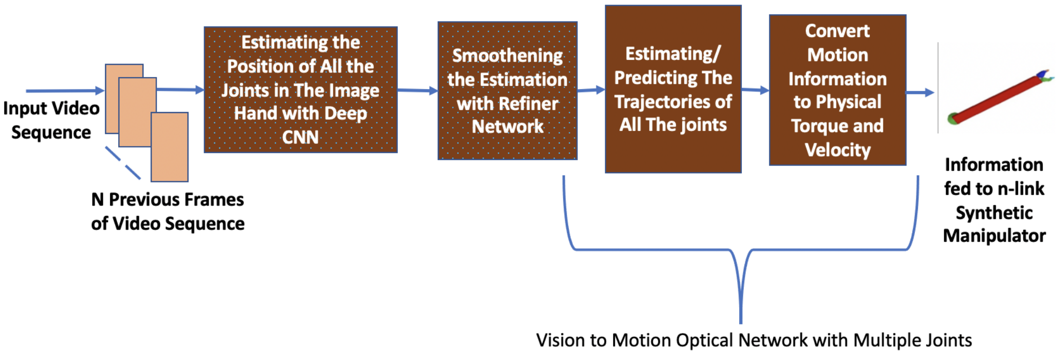

Input Video Sequence
N Previous Frames of Video Sequence
Estimating the Position of All the Joints in The Image Hand with Deep CNN
Smoothening the Estimation with Refiner Network
Estimating/ Predicting The Trajectories of All The joints
Convert Motion Information to Physical Torque and Velocity
Information fed to n-link Synthetic Manipulator
Vision to Motion Optical Network with Multiple Joints

Fig. 2: Block Diagram Representation of i-mimic With Multi-joint Manipulator

We can summarize the underlying novelties of the work described here as follows. The novelties include i) development of a new mechanism, namely i-mimic, to control robot arm only with hand movement videos as input, ii) formulation of an unsupervised network, namely, DON, which is enabled to convert input video streams to physical torque and velocity, iii) successful execution of real-time control of a 1-link manipulator, even with occlusion/ overlapping in the hands, by defining a prediction layer in DON, and iv) enhancement of the performance of DON to n-link manipulator by adding deep CNN and refinement networks as its predecessors.

The rest of the article is organised as follows. Sec. 2 presents the background research, Sec. 3 describes the layer-wise formulation of vision-to-motion optical network (DON). The architecture of CNN and Refinement Network customized loss function, dataset training are described in Sec. 4. The experimental set-ups with four different variations in experimental studies are described in Sec. 5. The real-time experimental results tested under different scenarios like without occlusion, with low occlusion, with high occlusion, and with multiple joints are given in Sec. 6 along with parameter section and comparative study. The overall conclusion of this work with its future scope are discussed in Sec. 7.

2 Background Research

Vision-based robotics to serve domestic purpose has drawn the attention of researchers in several areas. Most of the approaches developed so far for this purpose implied training the system through labeled data, i.e., with supervised or semi-supervised learning. Automated driving [8], grasping [11] and block stacking [10] are among few the applications where this kind of learning were used. But gathering adequate amount of labeled data for training is a challenging issue for this type of approach.

Semi-supervised learning or reinforcement learning has appeared to be the substitute of supervised learning in recent literature's where the training is carried out with less amount of labeled data or weakly labeled data. Rusu *et al.* [12] used learning with progressive network for Jaco robot gripping to have a faster algorithm with less amount of training data. KUKA IIWA robot grasping with deep network and domain adaptation was developed by Bousmalis *et al.* [3]. A method of training with weakly labeled images with adaptation from real world to simulation using a PR2 robot was proposed by Tzeng *et al.* [15]. Zuo *et al.* [16] came up with a solution of semi-supervised method of 3D pose estimation where the training was carried out in a virtual environment. Its real world implementation was done after domain adaptation. Domain adaptation is another challenge while semisupervised/ reinforcement learning is carried out. There are many rich works carried out so far to deal with this problem. Domain adaptation with back-propagation by inducing an 'inverse-gradient layer' to the deep network was formulated by Ganin and Lempitsky [6]. In another work Ganin *et al.* [7] came up with a solution of carrying out the training and testing of the network with the features that are non-discriminative and domain invariant for training and test data. Bousmalis *et al.* [4] came up with another solution of identifying the unique feature of each domain to extract out the common features in the domains. They have recently developed another way of domain adaptation with simultaneous simulation [3]. Sing *et al.* [13] demonstrated that passively collected data can be paired with interaction data to learn visual representations for end-to-end control policies that generalize substantially better to unseen environments. However, less amount of labeled data or synthetically labeled data are always required in all of the aforementioned approaches.

Economic cost of a robotic arm controller is another major issue to be dealt with to make the robotic arms be implementable for domestic purpose. The controlling of the robotic arms are normally carried out with multiple sensors which make it more costly. The robotic arms like PR2 [15], Jaco [12] or KUKA IIWA [3] costs around USD 20,000/- to USD 50,000/-. A pocket friendly robotic arm is developed recently [16] but but it still has the issues of synthetic labeled data and domain adaptation.

3 DON: Vision-to-Motion Optical Network

Here we developed a network based on the information of optical flow from frame-to- frame of a the input hand movement video sequence. The major challenges that are to be addressed in the task of 'i-mimic' are: unavailability of sufficient number of labelled data, computation time, and the lag between the input video to robotic-arm gestures. The proposed deep flow network is able to minimize all these parameters simultaneously. First of all, no labelled

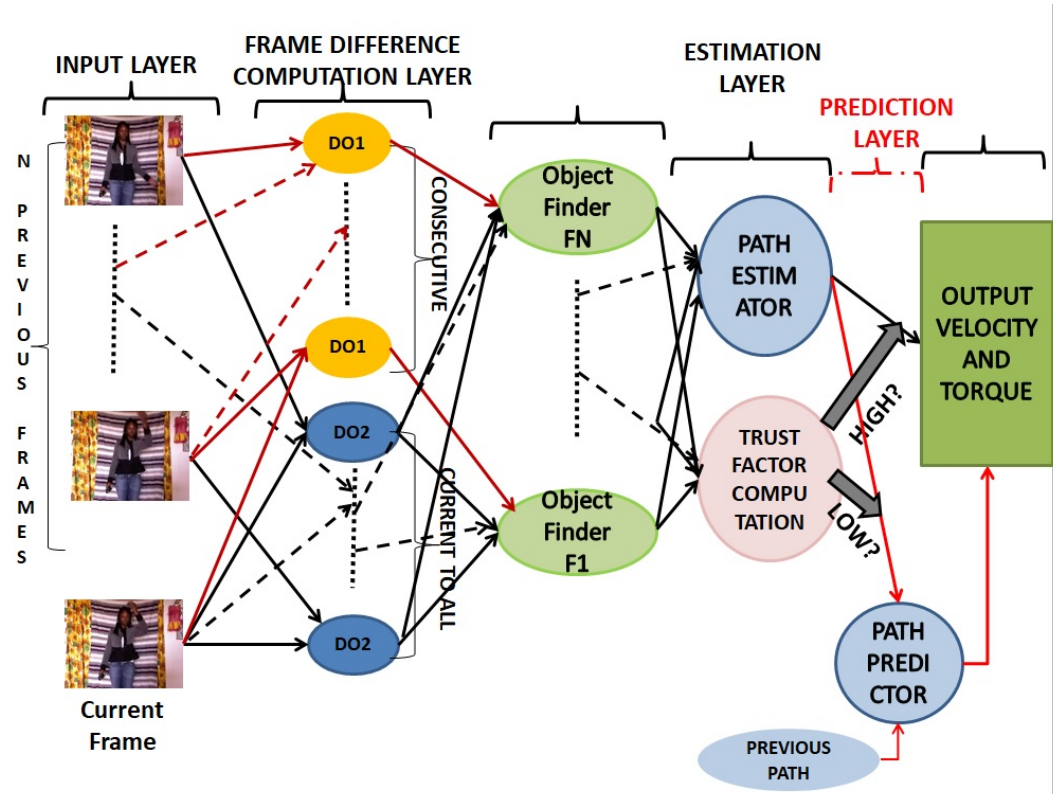


Fig. 3: Architecture of Vision-to-Motion Optical Network

data is required here and the process is fully automatic. The computation complexity of this method is quite low and there the lag is as less as around ten frames here.

The vision-to-motion optical network is a layer-wise network with multiple layers between the input and output layer. Different feature of the optical flow information process in different layers of this network. The layer-wise architecture is shown in Fig. 3. It can be noticed from the diagram that the videos are fed in the input layer of the network, whereas we get velocity and torque values that generate the physical motion at the output layer. That is why it is named as a ’vision-to-motion optical network network’. The output of the previous layer is the input to the next layer. All the layers of the network may not be active at a time. Rather, the activation of some layer of the network is dependent on the outputs of its previous layer. The layer-wise working principles of this network are described in detail in the following sections.

3.1 Layer 1: Input Layer

The video sequence is the input that is fed to the network. But it is not the entire sequence that is given as the input at a time since a on-line processing is going on here with the input video frames. As we have already stated that the proposed method is unsupervised, therefore the output is to be produced only by automated processing of input data. Here, the frame that gets generated at the current instant, say, at the instant t is fed to the network along-with N-number of previous frames that gets generated earlier to the current frame. Let the current frame be denoted as $f_t$here. The frames generated in the earlier instances can be denoted by $f_{t-1}, f_{t-2}, ..., f_{t-N}$. Therefore, the input layer contains the frames: $f_t, f_{t-1}, ..., f_{t-N}$.

3.2 Layer 2: Frame Difference Computation Layer

The network is supposed to deal with the optical flow information. In the case of the video sequence that we are dealing with is captured by static cameras. Therefore the changed information from frame to frame reflects the optical flow of the sequence. Here, two types of differences are computed here in this layer. That is why two different colored nodes (DO1 and DO2) are shown there in Fig. 3. There are total $N + N = 2N$ number of nodes present there in layer 2. The difference operation carried out in DO1 the difference between consecutive frames ($\delta 1$) given in Eqn (1). The difference between the current to all its previous frames ($\delta 2$) is carried out in Eqn (2).

$$\delta 1_p = |f_{t-p} - f_{t-(p-1)}| : p = 0, ..., N - 1 \tag{1}$$

$$\delta 2_p = |f_t - f_{t-p}| : p = 1, ..., N \tag{2}$$

Therefore N number of binarized $\delta 1$ and $\delta 2$ frames, i.e., $2N$ number of difference frames in total, are the output of this layer. $\delta 1_p$ are the binarized outputs from the nodes of type DO1, whereas, $\delta 2_p$ are the output from DO2 type of nodes.

3.3 Layer 3: Object Identification Layer

The third layer of this network is developed to find out the locations and the shape of the moving hand in all the N-number of previous frames. As it can be observed from Fig. 3 that this layer contains N number of nodes, labeled as $ObjectFinderF1, ..., ObjectFinderFN$. The input fed to a certain node $ObjectFinderFp$ are: $\delta 2_p : p = 1, ..., N$ and $\delta 2_p$. That is, all the DO2- difference frames and only p^{th} DO1 difference frame are input to the said node of layer 3. Let the location of the moving object segment in the p^{th}-frame be represented by l_p. The operation that is carried out in each node of the third layer is given by the Eqn. (3).

$$l_p = (\cup_{p=1}^{N} \delta 2_p) \cap \delta 1_p \tag{3}$$

Please note that the union of $\delta 2_p \forall p = 1, ..., N$ is taken here to have the entire moving obeject region as a subset of that union and intersection of it to that of $\delta 1_p$ is carried out to extract out the obvious moving region in the p^{th}-frame. The pixels those belong to the set l_p are in the region that definitely belong to the moving object in the p^{th} frame. For the sake of simplicity, here in this work we consider only the skeleton and the locations of the corner pixels of l_p (moving hand) to be the output from each node of this layer as we need to find out the angular velocity and torque from the hand movement video.

3.4 Layer 4: Estimation Layer

This layer contains two nodes and the operations and functioning of these two nodes are different from each other. The location of the object in N-number of frames are input to this layer. Two types of estimations are performed simultaneously in this layer with the two nodes. The path estimation node gives the probable trajectory of the moving object as the output whereas the trust factor estimator node computes the reliability of the estimated path. The output of the trust factor estimator node determines the activation of the next layer, i.e., the prediction layer. The working principles of the two nodes in the forth layer are described below.

3.4.1 Path Estimator

As discussed before, the prediction of the probable trajectory of the object is carried out here. This is done by computing the optical velocity and acceleration of the moving object from frame-to-frame displacement. Let ς_p be the location l_p (see Eqn. (3)) in the p^{th}-frame. Then the velocity (v_p) and acceleration (a_p) of that object are computed according to Eqn. (4). The velocity and acceleration values for all the N frames are stored in the sets V and A respectively.

$$V = \{v_p : v_p = \varsigma_p - \varsigma_{p-1} \forall p = 1, ..., N\} \tag{4}$$

$$A = \{a_p : a_p = v_p - v_{p-1} \forall p = 2, ..., N\} \tag{5}$$

Please note that signed differences between the locations and velocity are taken while computing v_p and a_p in Eqn. (4). It is known from Sec. 3.3 that the input ς_p could be a scalar or vector component based on the type of object representation. However, the two components v_p and a_p should always be a vector since these components contain both magnitude and signs. Consideration of those signs helps in the incorporation of the information of the direction and the change in the direction of the moving hand.

Here the robotic arm with revolute joint is supposed to mimic the movement of the arm shown in real time or recorded video. Therefore, the movement of the arm is always supposed to be circular in nature with respect to any joint (e.g. elbow) with a maximum 180 degree of freedom. This phenomenon is kept in mind and the determination of the radius w.r.t angular motion is computed by measuring the length of the skeleton of the arm. Let the skeleton of the moving part of the arm be of length r. The angular velocity (ω_p) and torque (θ_p) are then computed as:

$$\omega_p = \frac{v_p}{r}$$

and

$$\theta_p = I\alpha_p$$

where I is mass moment of inertia of the manipulator arm and α_p is the angular acceleration computed as:

$$\alpha_p = \frac{a_p}{r}$$

For any given one-link manipulator the algorithm computes α_p and having I of the manipulator one can compute the torque required to be applied to at the joint.

3.4.2 Trust Factor Computation

The working principle of this particular node is different from any other nodes present in the network. It takes input from the previous layer but does not transmit its output to the next layer. Instead, the output from this layer determines which layer should be the fifth layer of this network. That is, which path should be followed by the output information from the path estimator node is decided with the output of this node. Since only motion of the moving hand of a static person is considered here, it can be assumed that the size of the moving object will remain almost the same throughout the sequence. This assumption is applied during formulation of the trust factor. Let there be M_p be the region of l_p (see Eqn. (3)) in the p^{th}-frame. Let, the set $\{S\}$ the regions of the object in all the N frames and the set $\{S_d\}$ contains the values of change in regions. Those are computed according to Eqn. (6).

$$\begin{aligned} S &= \{M_p : p = 1, ..., N\} \\ S_d &= \{c_p : c_p = |M_1 - M_p| \forall p = 2, ..., N\} \end{aligned} \tag{6}$$

The trust factor (η) is computed as:

$$\eta = 1 - \frac{max(S_d)}{max(S)} \tag{7}$$

In Eqn. (7) max(.) represents the element with maximum magnitude present in a set. Physically, the effectiveness of measuring the η is in determining the amount of occlusion took place over the moving object. If large amount of occlusion is present there for some frames, then the estimation with those frame may lead to a wrong trajectory. Therefore, the prediction should be carried out from the previous set of information and ignoring the wrong (occluded) visual information. That is why the activation of the prediction layer is necessary in this scenario. The path leading to prediction layer gets only activated if the value of η is low.

3.5 Layer 5: Prediction Layer

There is only one node in this layer. But, the input fed to this layer is not only from the previous layer, but the output of layer 4 of the previous execution of the network is also input here. Please note that this layer can not be active in the first execution of the network, but from the second execution onward it could get activated any time. Here the velocity and acceleration values of the frames without occlusion, or with minimal occlusion are considered. There inputs that are provided to this node are: i) the velocity and acceleration information (sets V and A) from the previous execution, ii) the velocity and acceleration information (V and A from Eqn. (4)) from the previous layer and iii) Object regions and change in the regions (S and S_d from Eqn. (6)). Let the velocity and acceleration from the previous be denoted here as $\tilde{V}$ and $\tilde{A}$. We only consider the information of the frames for with $c_p < 0.05Xmax(S)$ (C_p is as defined in Eqn. (6)). That is, the frame maximum with 5% change in object size will be taken into account. Let k number of frames out of the N frames failed to satisfy the criterion. Then only $N-k$ elements from the sets V and A will be taken by merging it with the sets $\tilde{V}$ and $\tilde{A}$ respectively. Therefore, the new sets will be $V^k = \{\tilde{V}|V(1 : N-k)\}$ and $A^k = \{\tilde{A}|A(1 : N-k)\}$ with $N + N - k = 2N - k$ the number of elements in each set. Now we need to predict the information from the $(N-k+1)^{th}$ frame onward. As it is known, the consecutive difference between the elements of V^k forms A^k, i.e., A^k could be said the first order derivative of V^k. We can similarly compute the second-order derivative of V^k or the first order derivative of A^k and represent it by $A^{k'}$. Now, the $(2N-k+1)^{th}$ element of the sets V^k and A^k will be approximated as:

$$\begin{aligned} v_{2N-k+1} &= v_{2N-k} + a_{2N-k-1} \\ a_{2N-k+1} &= a_{2N-k} + a'_{2N-k-1} \end{aligned} \tag{8}$$

In Eqn. 8 the symbols V_p, a_p and a'_p represents the p^{th} element of the sets V^k, A^k and $A^{k'}$ respectively. The element will get inserted to the sets V^k and A^k as the $(2N-k+1)^{th}$ elements of them. The process will be repeated and the next element will be approximated. The process will continue until the set is going to have $2N$ number of elements. Once it is done, the last N elements of V^k and A^k will be stored in the sets V and A respectively and will be given as the output to the output layer.

The experimentation and those are carried out with this proposed DON are described in the following section. Please note that one additional layer, namely the object regression layer to this network is introduced while working with multiple joints. It is shown in Fig. 10.

4 DNN:Deep Neural Network

In our second approach, we designed Convolutional Neural Network(CNN) for which the input corresponds to the image(frames of video stream) and labels correspond to coordinates(x, y) of joints(shoulder, elbow and wrist joints). We have used two networks for our purpose to get the joints coordinates from which further joint angles, joint velocities, and and joint torque can be obtained in terms of pixel coordinates which are mapped to real value using the mapping function or mapping factor. In [14] after using DNN based regression, a DNN based refiner network was added, which takes the cropped images around the prediction as input to improve the prediction, but we used a different approach by mapping a simple neural network to refine the predictions of the DNN based regression network. Our approach reduces the training time of the network with similar accuracy for our dataset.

4.1 Convolutional Neural Network(CNN)

The architecture of CNN is shown in Fig. 4. The input to this network is the images(from a stream of video feed) and labels as the pixel coordinates(x, y) of the shoulder joint, elbow joint and wrist joint. The network has six hidden layers wherein there are 2 Convolutional layers followed by max pooling and flatten. Activation functions for all layers other than the last layer are ReLu activation. Final layer has a Linear Activation function. This network uses Mean Square Error(MSE) Loss function.

4.2 Refiner Network

The architecture of Refiner Network is shown in Fig. 5. This network is a simple neural network with input corresponding to the output of CNN Network and labels the pixel coordinates of the shoulder joint, elbow joint and wrist joint, respectively. Activation function

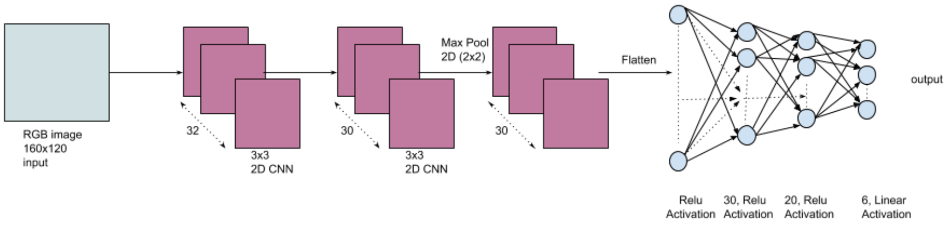


Fig. 4: CNN Network

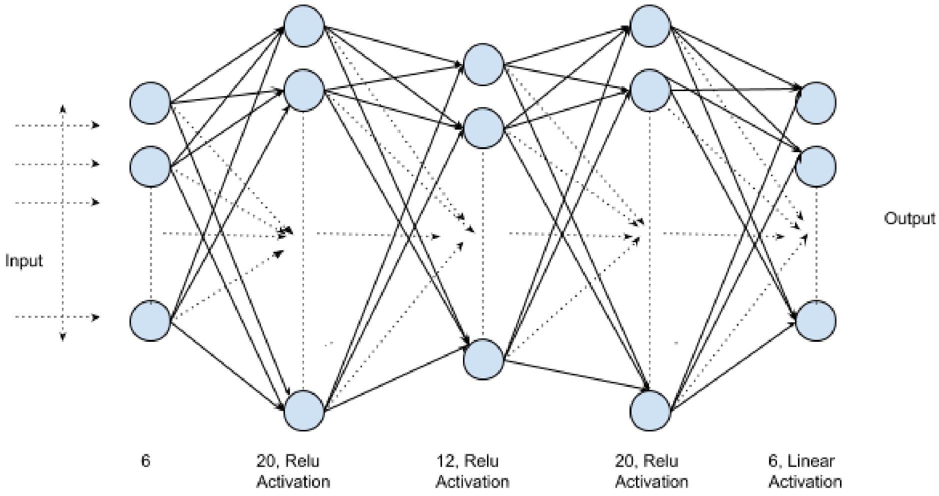


Fig. 5: Refiner Network

for the last layer is linear and for rest is ReLu Activation. This network uses a customized loss function that includes MSE and simple errors in link length.

4.3 Customized Loss Function

For the error in coordinates of the joint, Mean Square Error(MSE) has been used and another error for link length has also been taken into consideration. Let d denote error in link length and p denote the MSE in position. For real joint coordinates (x_s, y_s), (x_e, y_e), (x_w, y_w), and predicted coordinates (x_{sp}, y_{sp}), (x_{ep}, y_{ep}), (x_{wp}, y_{wp}) where subscripts s, e and p represents shoulder, elbow and wrist joints respectively and subscript s, e, w followed by p represents corresponding predicted coordinates respectively. The respective error are computed following the Eqns. (9)-(11).

$$d = (\sqrt{(x_s - x_e)^2 - (y_s - y_e)^2} - \sqrt{(x_{sp} - x_{ep})^2 - (y_{sp} - y_{ep})^2})^2 + \\ (\sqrt{(x_e - x_w)^2 - (y_e - y_w)^2} - \sqrt{(x_{ep} - x_{wp})^2 - (y_{ep} - y_{wp})^2})^2 \quad (9)$$

$$p = (x_s - x_{sp})^2 + (y_s - y_{sp})^2 + (x_e - x_{ep})^2 + (y_e - y_{ep})^2 + (x_w - x_{wp})^2 + (y_w - y_{wp})^2 \tag{10}$$

$$loss = d + p \tag{11}$$

4.4 Dataset

For the purpose of training the deep CNN, we created our own dataset of 300 images and manually labelled the pixel coordinates of the joints for each of the images. From the total dataset, 224 images were used for training and the rest were used for testing.

4.5 Training

Our second approach requires training for which the training and validation dataset has been used as specified under sub-heading 4.4. For the optimization purpose, Adam optimizer has been used with a batch size of 14 images. The training has been done on the sample dataset for the case of one-link and two-link cases(forearm and arm) both on CNN network and refinement networks and the combined network. CNN network has been trained on 10 epoch and Combined Network on 100 epoch.

5 Experiment

The proposed method is tested with both real-time and recorded video sequences. But the processing of both types of the sequences are carried out in real-time since the ’mimic’ of the robotic arm is a real-time task. The experiments adressing four different challenges viz., i)presence of different level of background noise, ii) variation in distance between arm and camera, iii) variation in speed of hand movement, and iv) different number of links are performed to test the proposed methodology and setups are accordingly made. The experiments have been performed at frame rate of 30 fps and video resolution of 240x320 pixels.

5.1 Experiment-1

In the first experiment, the recorded video of hand motion is given as input to the network and the motion of the arm is mimicked by one-link manipulator in virtual environment of PyBullet. This experiment requires the preparation of a virtual environment and no physical setup is required. For the recorded video even noisy data was used to test the method. The recorded data used for testing is of [5] (lossy compressed AVI format, devel-1). Additional setups were not made for getting the recorded video. Videos used are available here as Exp-1 Video-1 and Exp-1 Video-2)

5.2 Experiment-2

In the second experiment, the motion of actual one-link manipulator with stationary background is mimicked in simulation environment. This required preparation of the physical setup of the manipulator and camera. The experimental setup consists of a camera(Model: HP HD 4310 H2W19AA) is mounted(fixed) at a height of 32 cm(can be varied) above the manipulator. The manipulator's link length is 10.5 cm which is made up of paper to reduce the weight of the link mounted on the servo motor(specifications: Model: SM-S2309S, Size: $22.9 \times 12.3 \times 22.2$mm, Weight: 9.9g, Rotation angle $\equiv 120$, Micro analog servo, 4 plastic gears + 1 metal gear). An Arduino UNO board has been used as a controller to provide signal to the servo motor for the motion (Fig. 6(a)). The video of experiment is available as Exp-2 Video-1, and Exp-2 Video-2.

5.3 Experiment-3

In the third experiment, the proposed method is used to mimic the forearm motion of a standing person by an actual one-link manipulator. The one-link manipulator used in Experiment-2 has been used here except the positioning has been changed as shown in Fig. 6(b). Here, the distance between the forearm and the laptop(HP laptop AU030WM Pavilion) camera is varied. whereas the camera was mounted at fixed distance from the manipulator in the earlier setup. Also, the background here is not stationary as noise is introduced while moving the forearm, other body parts move slightly. Moreover, the i-mimic, or the ability of the robot arm to mimic in real-time and even in presence of severe occlusion has been experimentally validated here. validated here. The video of experiment is available as Exp-3 Video-1, and Exp-3 Video-2.

5.4 Experiment-4

In the fourth experiment, we extend our method to n-link planar manipulator(four-link, one-link is fixed). Here, in the case of the human arm, we considered the shoulder, elbow, and wrist having three joints overall. Two additional layers of DON is added for the sake of experimentation here. It is shown in Figs. 2 and 10. The video of the labelling process with real-time video streams could be found here as an example. Please note that no physical setup is prepared for this experiment and the testing is done in simulation. The manipulator used in Pybullet is shown in Fig.6(c). The video of experiment is available as Exp-4 Video-1, and Exp-4 Video-2.

5.5 Parameter Tuning

The actuator of our experimental setup enabled us to test the method of position control. However, velocity control and torque control techniques can also be used with the values obtained from the algorithm with actuators that enables velocity control and torque control. For the uniformity, we use position control in the simulator as well as an actual manipulator.

The joint angle computed in the optical flow and the actual joint angle remains the same. Same is not the case with angular velocity, angular acceleration, torque, and link length. The

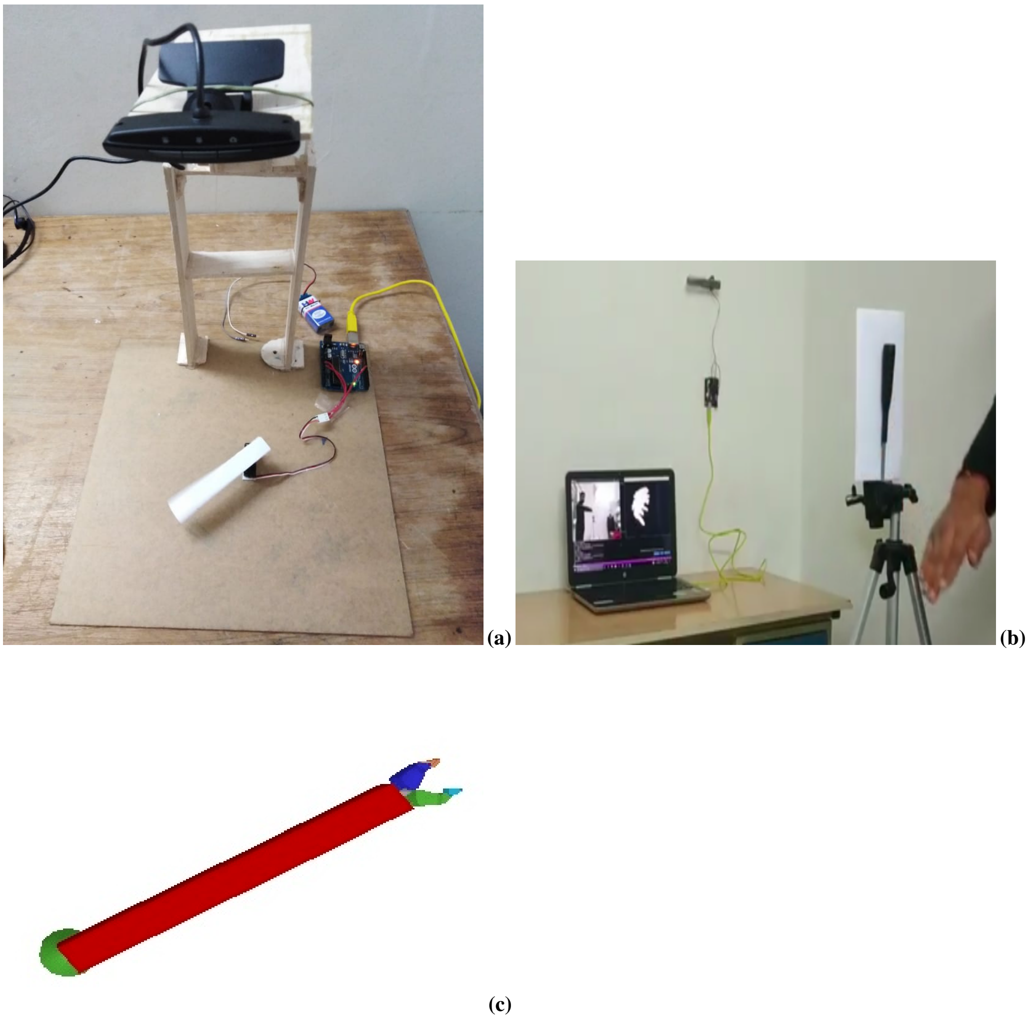
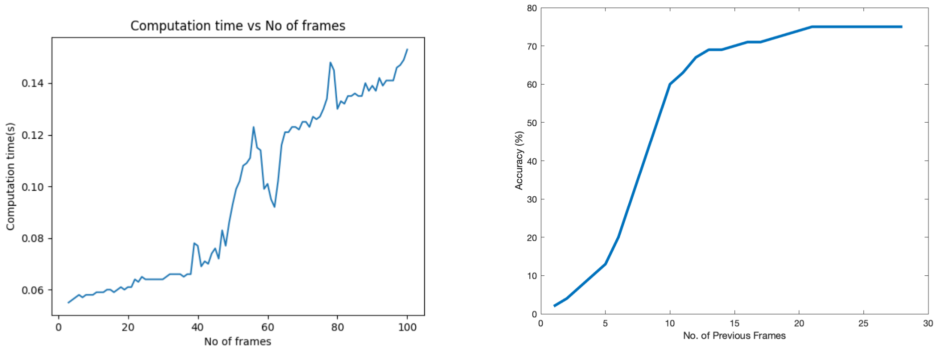


Fig. 6: (a) Experimental setup for experiment-2, (b) Experimental setup for experiment-3, (c) Manipulator for experiment-4

term aspect, the ratio(ratio of value of optical flow and actual value) has been introduced for mapping the optical flow value to the actual value. These values shall be experimentally determined and is dependent on the experimental setup.

Since the algorithm is completely unsupervised, the requirement of labelled data and domain adaptation is not required. The issue of cost is also dealt since the algorithm can easily run on low computation powered devices such as mobile handset, laptops, computers, etc. There is only lag time between the input of the image frame of video and the output signal of the manipulator which is the processing time.

The number of previous frames (N), we need to feed to the input layer of DON is a vital parameter in this work. Increasing N will result in high computation time (as can be observed in Fig. 7(a)), as well as a gain in accuracy (Fig. 7(b)). To come up with the opti-



(a) **(b)**

Fig. 7: (a) Computation time vs No of frames considered, (b) No. of previous frames considered vs. accuracy

mum number of N, the issues of both computation time and accuracy have to be considered. The change of computation time and accuracy in tracking the proper object region along with the increasing number of previous frames is shown in Figs. 7(a) and 7(b) respectively. Based on our observations from Figs. 7(a) and 7(b), we have found that a balance between time and accuracy could be obtained by setting the value of N to 10.

Please note, the segmentation/ tracking accuracies, used in this study are computed based on the number of correctly identified pixels ($P_{TrueDetect}$) by our algorithm compared to that of the ground truth (P_{GT}). The accuracy (A) can be computed with the following Eqn. (12).

$$A = (1 - \frac{|P_{TrueDetect} - P_{GT}|}{P_{GT}}) * 100 \tag{12}$$

Furthermore, performance of Object Identification Layer is also dependent on the intensity of light and background noise(movement of other objects). The algorithm works well for the intensity of light above 50%(determined using the experimental setup in Fig. 6(a).

6 Results and Discussions

6.1 Results of DON

The Experiment-1 performed on the hand motion data set could hardly track the arm joint angle due to extremely random and fast hand movement and very large noise due to movement of other body parts. This experiment was performed on both the RGB video and depth video. The results are available here RGB videos(Video-1 at 10fps, Video-2 at 30fps) and Depth videos (Video-1 at 10fps, Video-2 at 30fps)

The Experiment-2 performed for mimicking the actual one-link manipulator motion in the simulation result is shown in Fig. 8(a). In addition we used [9] to generate depth images from RGB images and tested our algorithm. In both cases, our result obtained is the same. We have also studied the angular velocity of the human hand and the robot arm and it is plotted in Fig. 9(a). The red line in this figure shows the variation in angular velocity of the robot arm, ehereas the blue line reflects the same for the human hand in the video. The error or the difference between these two angular velocities has been plotted in Fig. 9(b).

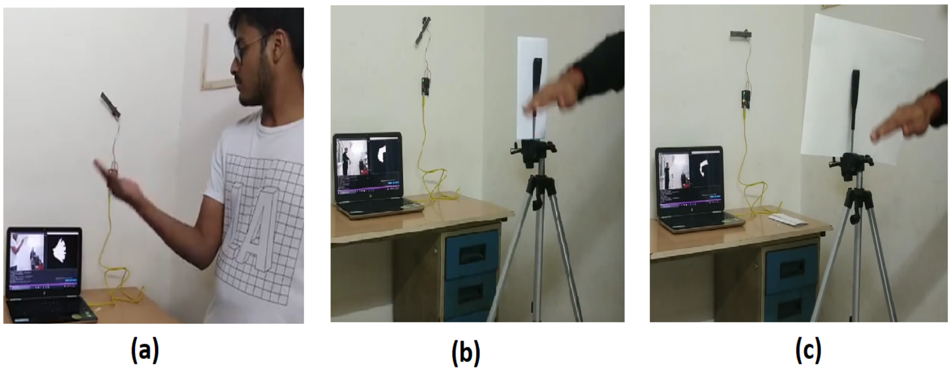
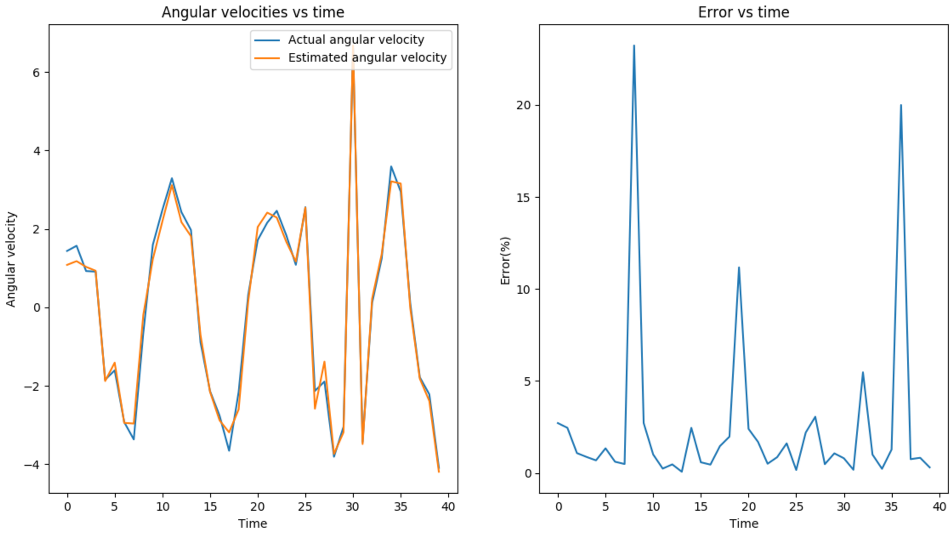


Fig. 8: Visual *i*-mimic in Real-time: (a) Without occlusion ($0.9 < \eta < 1$), (b) Low occlusion to background and human hand ($0.7 < \eta < 0.5$) and (c) High occlusion ($0.4 < \eta < 0.3$)



(a) **(b)**

Fig. 9: Plot of (a)Angular velocity vs Time, (b) Error vs Time

The Experiment-3 is performed with the variation of trust factor (η in Eqn. (7)) that is with various degrees of occlusion. Example frames with arm-mimic for three different types of occlusions are shown in Fig.10, where there is no occlusion present there in Fig. 10(a), low amount of occlusion is present there in Fig. 10(b) and the amount of occlusion is quite high in Fig. 10(c). It is also observed that the proposed method works well for moving hand.

The videos corresponding to the experimental results areno occlusion, minor occlusion and major occlusion.

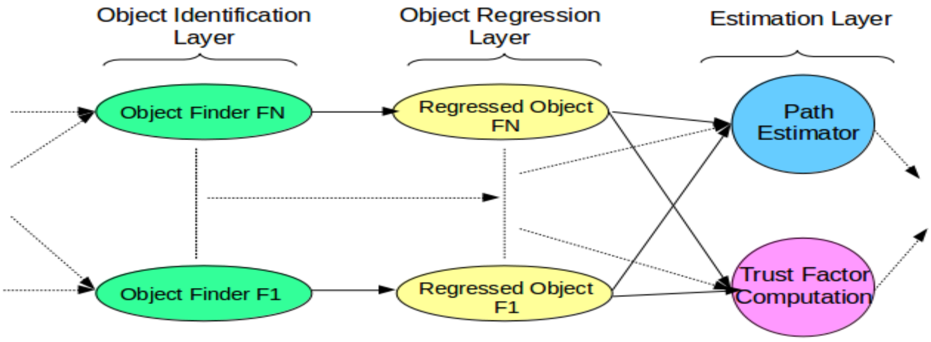


Fig. 10: Adjusted portion of Architecture of Deep Flow Network

In Experiment-4, since there are three rotating links, three-joints angles are to be estimated. To accommodate multiple links, in the architecture of DON, an additional layer is added between Object Identification Layer and Estimation Layer to fit straight lines(Object Regression Layer) on the objects as shown in Fig.6. Rest of the network remains the same. The joint angle between the forearm and fixed link, thumb and forearm, index finger, and forearm are estimated accordingly. The result obtained after Object Regression Layer is shown in Fig. 12. The loss function of training (blue) and testing (red) is shown in the top right image of Fig. 12, where deep CNN network is only used. The plots of the same functions are shown in the top left image of Fig. 12, where the combination of CNN and refiner network is used. The scatterd plot in Fig. 12 reflects the true locations of the true hand segments (red dot), and the locations generated by the synthetic arm (green dots). As it deals with single link manipulator, only the variations for two joints have been plotted.

The video expressing the obtained result is here.

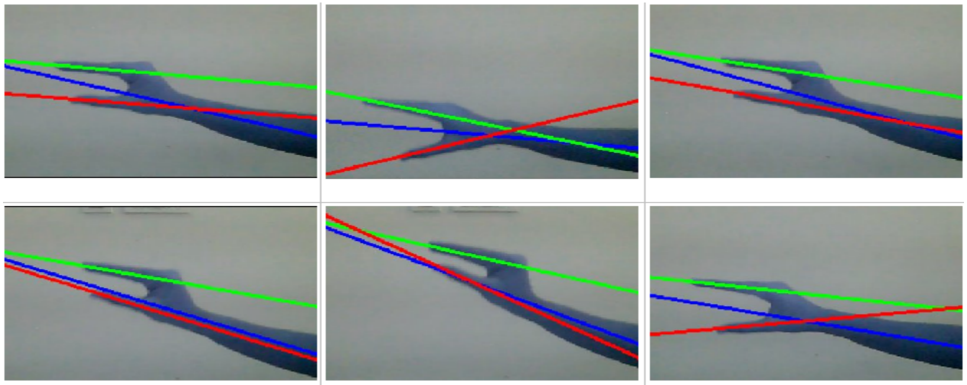


Fig. 11: Snapshot image after Object Regression Layer

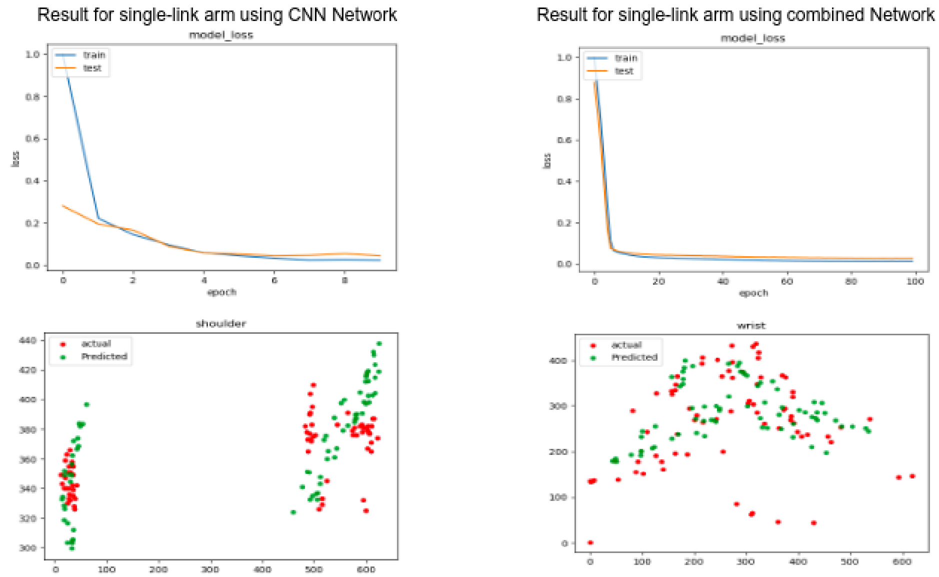


Fig. 12: Result for single link arm

In addition to this, the algorithm has been tested on videos of different resolutions at 30 fps. The average algorithm run time per loop execution for video input of different resolution before giving the signal to manipulator for is presented in Table 1. This experiment has been performed on DELL Laptops with 8 GB RAM and Intel Core i5, the algorithm run time will vary depending upon the computation power of hardware used for testing.

Table 1: Average loop run time for videos of different resolutions.

Resolution	Average loop run time(in milliseconds)
240x320	20
480x640	45
720x960	68
960x1280	124

6.2 Results of DNN with Two Links

Here the CNN network and Refiner network has been tested on two link cases. The loss occurred during the training are shown in Fig. 13. Likewise, the scattered plots of actual coordinates and predicted coordinates by combined Networks of one-link shoulder joint and wrist joint obtained are also shown in Fig. 13.

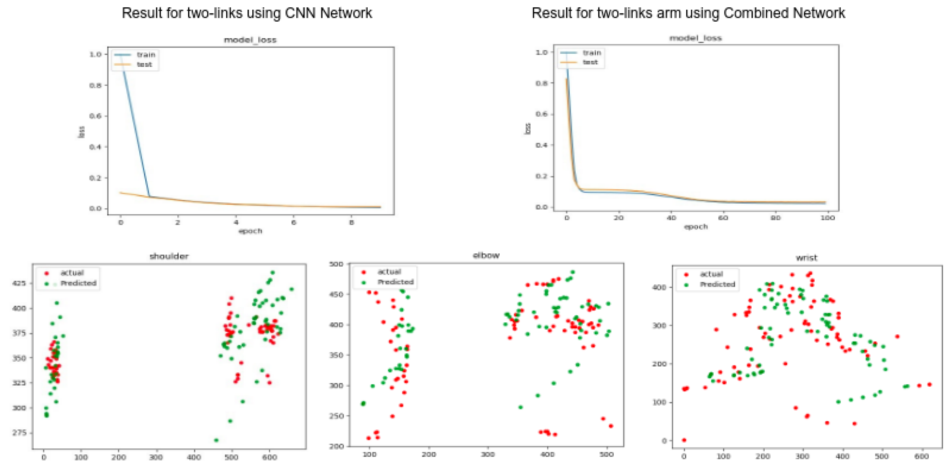


Fig. 13: Results for two link arm

6.3 Discussions

As we stated earlier, four different experiments under different circumstances joints are performed here. In the case of extremely random and very fast hand movement, the method is found to be ineffective as in Experiment-1, while in Experiment-2 where the background noise, arm speed are limited, the method performance i.e. mimicking is near to perfect. There is negligible lag time because the arm control command is given to the simulator which runs nearly at 240Hz in Pybullet. In Experiment-3, the mimicking is performed with small lag time. This lag time(as seen in video) has been caused due to the hardware limitation of the manipulator and setup. This can be reduced with good enough hardware, since the algorithm has been fine-tuned with optimal parameter values. The algorithm extended to n-link planar manipulator in Experiment-4 is able to estimate the three-joint angles between lines accurately as depicted in Fig.6. The algorithm can be extended to n-link planar manipulator just by introducing Object Regression Layer and the outcome would be as desired. Furthermore, experimenting with videos of different resolution shows the algorithm run time increases with the increase in video resolution. In addition, our algorithm tracks/detects the objects on the basis of motion and not probabilistic color distribution or object features. This enables our algorithm to run on both RGB and depth images independently and give the same result.

In case of our second method, for the one-link(forearm) case, the Predicted Joint positions improved after adding a second network. The second network has acted as a smoothing/refining network which refines the outcome of the first network. For the two link cases, the performance of the combined network is better compared to just the use of CNN Network.

6.4 Comparative Study

Please note that first approach's application that we proposed here is new to literature. Therefore, no similar method is available to compare with. Therefore, we focus on comparing the proposed tracking algorithm. We could not conduct any direct comparative study for our tracking method too, since no other tracking algorithm, formulated so far gives torque and velocity as the output. For example, we carried out the same experimentation with two other robust and popular unsupervised tracking algorithms, namely, MoG2 and CAMSHIFT for the sake of comparison. In this study, we verified that the proposed algorithm tracks down the object within about 30 ms, whereas the time consumed by CAMSHIF and MoG2 are about 149 ms and 100ms respectively on HP laptop AU030WM Pavilion. Besides, these methods are not robust enough since they loose the object trajectory even to the stationary background, therefore, its performance gets reduced with the reduction of trust factor. Above all, none of the algorithms enables us to compute the values of kinematic and dynamic parameters of motion like our algorithm for mimicking and hence failed to mimic. This is the main cause why we failed to carry out a suitable comparative study for this application.

For the second approach of deep learning, we compare our results to those of [14] using Percentage correct parts(PCP) at a link length threshold of 0.5(PCP 0.5), for upper arm and lower arm. PCP 0.5 was calculated on our model using our dataset on 20 images and on 40 images. The comparison results are shown in Table 2 and 3 respectively. The data taken from [14] is generalised on the large dataset that has been used, in our case, the method has not been generalised, and results shown are obtained on our dataset.

Table 2: Comparison PCP(0.5) on 20 images

Model	Upper Arm	Lower Arm
Deep Pose 1st	0.5	0.27
Deep Pose 2nd	0.56	0.35
Deep Pose 3rd	0.56	0.35
CNN Network	0.40	0.1
CNN + Refinement Network	0.7	0.35

Table 3: Comparison PCP(0.5) on 40 images

Model	Upper Arm	Lower Arm
Deep Pose 1st	0.5	0.27
Deep Pose 2nd	0.56	0.35
Deep Pose 3rd	0.56	0.35
CNN Network	0.25	0.125
CNN + Refinement Network	0.566	0.275

From the above table, it is clear that our combined network comprising of CNN and Refiner Network performs better compared to just CNN network and Deep Pose Network for Upper Arm. However, for lower the results are as good as that of Deep pose for the case of 20 images. With increasing the number of images to 40, performance decreases but still is better than that of Deep Pose and CNN Network for Upper Arm. However, for the lower arm, combined is poor.

7 Conclusions and Future Work

In the proposed work, we aimed to develop a method with which robotic arms could be controlled only by showing video sequences in real-time. The proposed method is proven to be successful with one arm manipulator (Exp. 1), with real-time video streams as input (Exp. 2) even in presence of occlusion (Exp. 3), and with multiple joints (Exp. 4). This approach is proven to be successful with the adequate amount of demonstrations, the links of some such videos are provided in this article itself. The unsupervised DON based method is proven to be effective in achieving control over single-link manipulator in 2-D plane. Control over n-link manipulator plane has been achieved better with hybridization of CNN and refinement network along with DON as shown here in the study. The proposed techniques performs well both with real-arm manipulator and synthetic arms. The mimic-based control therefore could be implemented to control the robotic arm in different tasks. The approach is in its entry level as of now and more complex scenarios could be addressed in the future by mimicking the motion of multiple joints (finger joints) of a manipulator in spatial 3D environment. This way it could have a vast application in different areas of robotics and artificial intelligence.

Conflict of interest

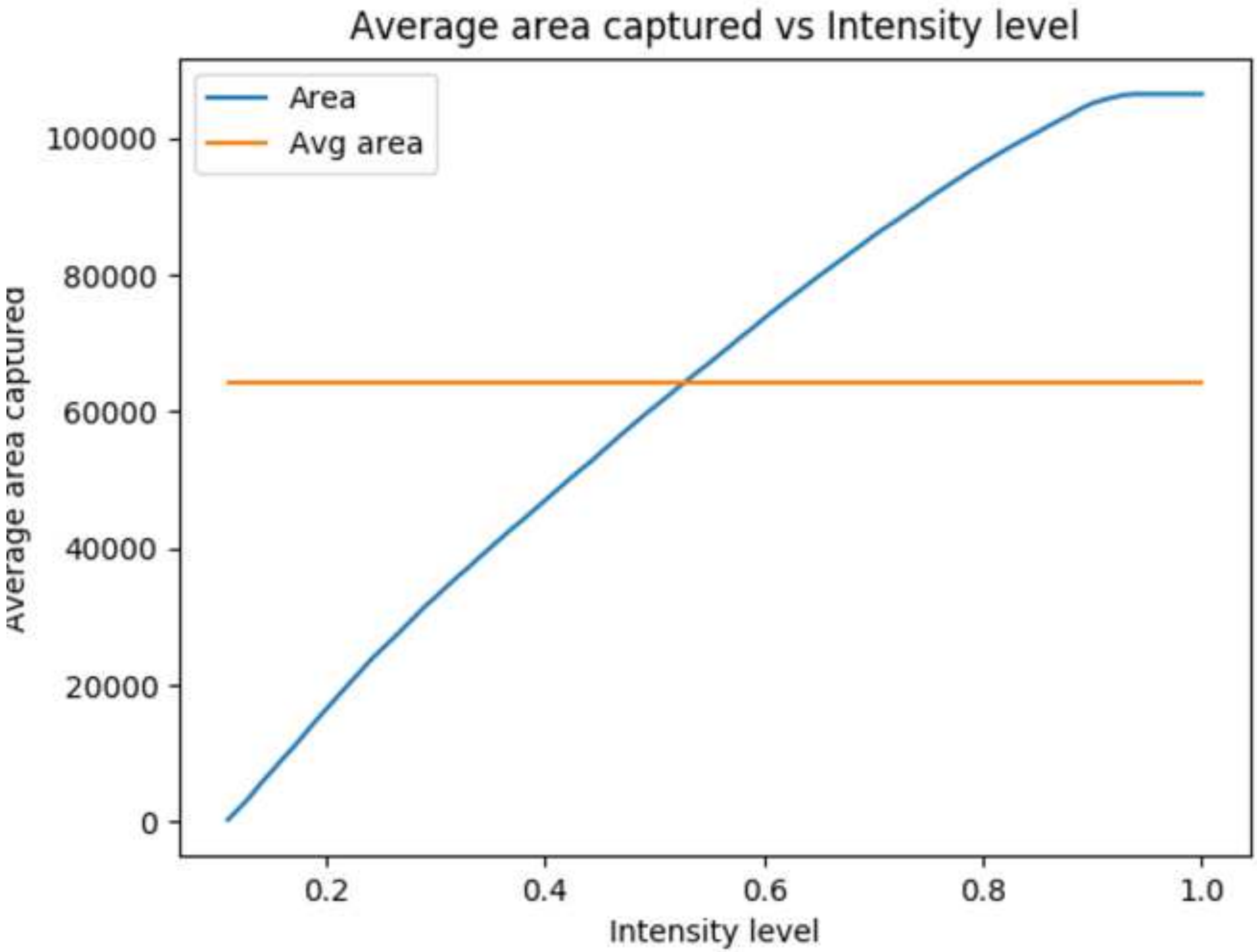
The authors declare that they have no conflict of interest.

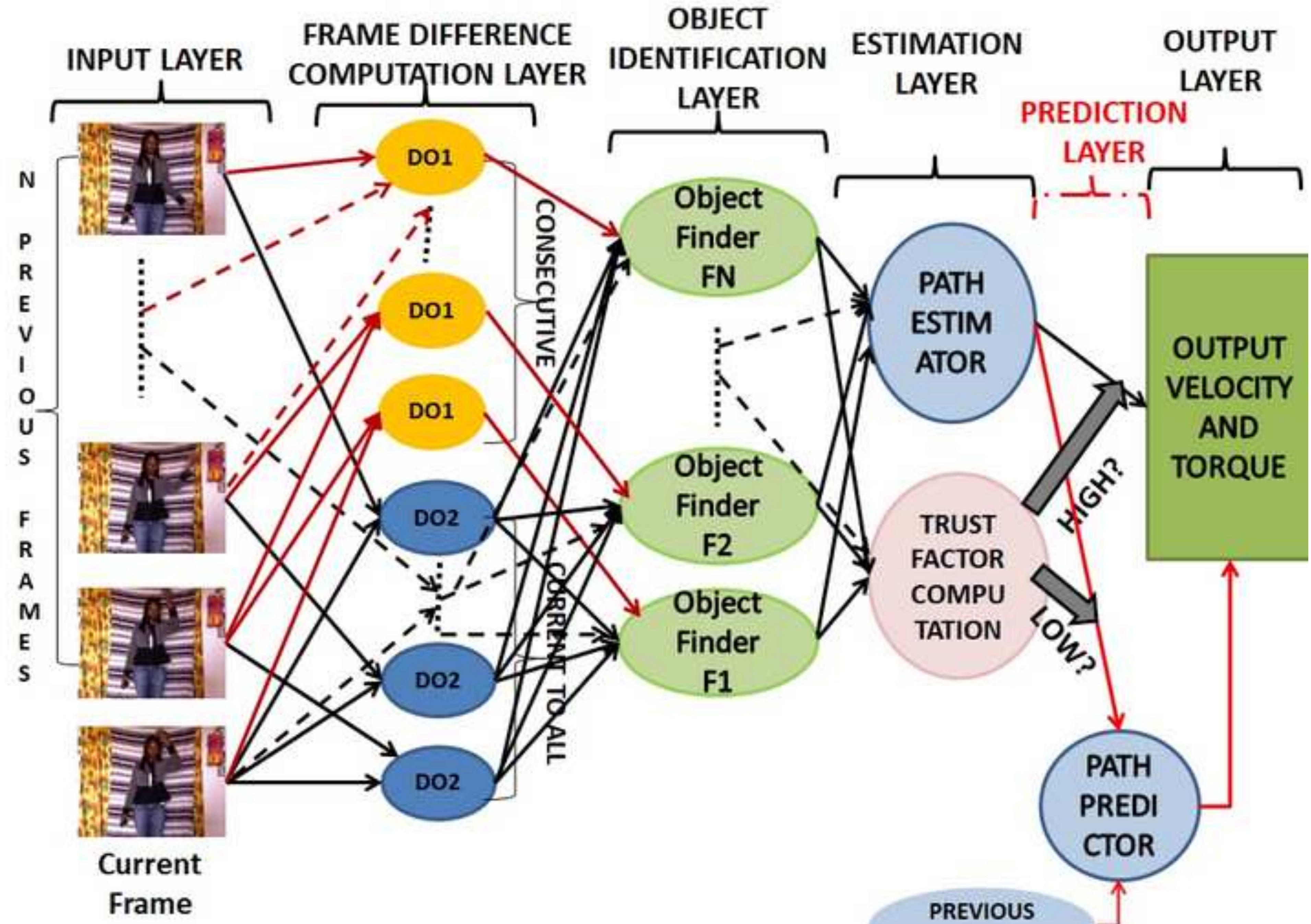
References

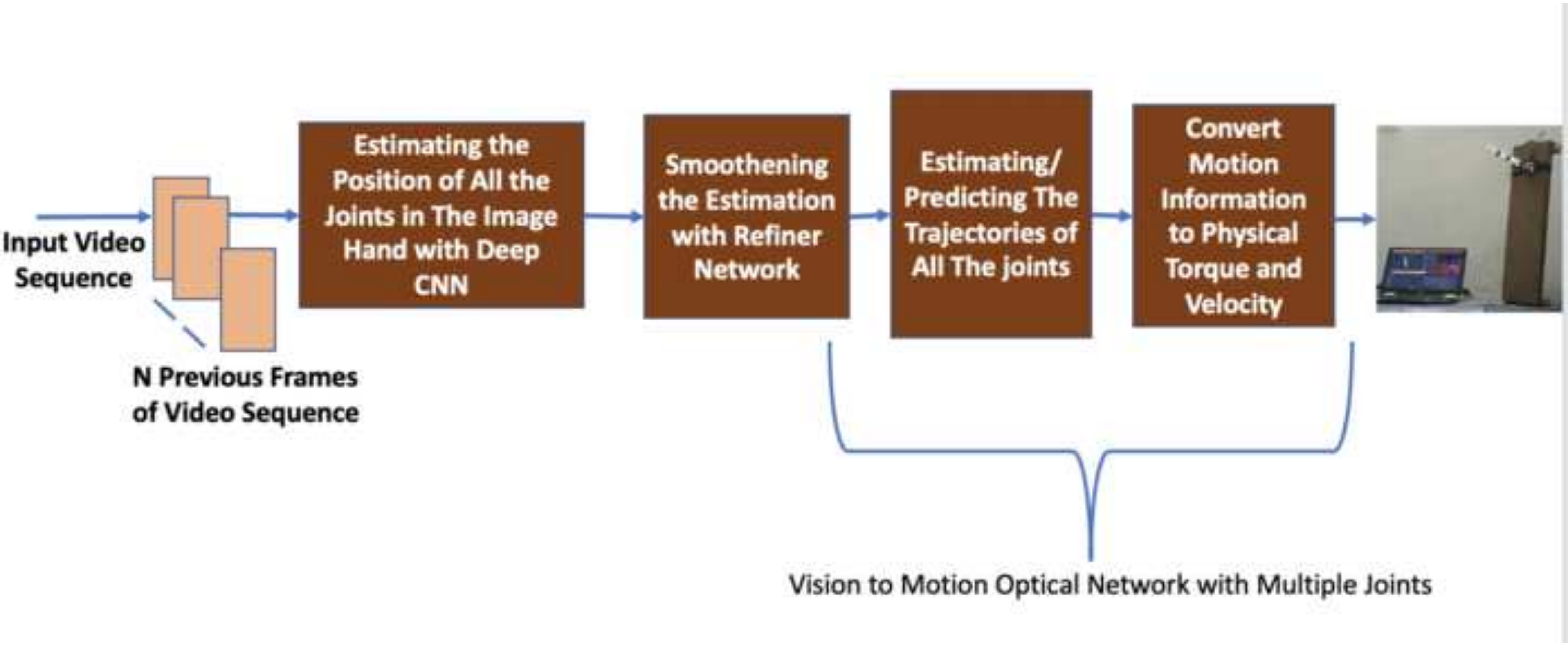
1. Ali, M.H., Aizat, K., Yerkhan, K., Zhandos, T., Anuar, O.: Vision-based robot manipulator for industrial applications. Procedia Computer Science, Elsevier **133**(2), 205–212 (2018)
2. Basu, R., Padage, S.: Development of 5 dof robot arm-gripper for sorting and investigating rtm concepts. Materials Today **4**(2), 1634–1643 (2017)
3. Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., K. Konolige, e.a.: Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In: International Conference on Robotics and Automation (2018)
4. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain separation networks. CoRR **abs/1608.06019** (2016)
5. Dataset, C.G.: Chalearn gesture dataset (cgd 2011), chalearn, california, 2011. Data retrieved from ChaLearn Gesture Dataset , http://gesture.chalearn.org/data/cgd2011
6. Ganin, Y., Lempitsky, V.S.: Unsupervised domain adaptation by backpropagation. In: International Conference on Machine Learning (ICML) (2015)
7. Ganin, Y., Ustinova, E., Ajakan, Germain, P., Larochelle, H., Laviolette, F., Marchand, M.: Domain-adversarial neural networks. Journal of Machine Learning Research **17**(1), 1–35 (2016)
8. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
9. Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth prediction. In: The International Conference on Computer Vision (ICCV) (2019)
10. Hundt, A., Jain, V., Paxton, C., Hager, G.D.: Training frankenstein's creature to stack: Hypertree architecture search. arXiv preprint arXiv:1810.11714 (2018)
11. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research **37**(4-5), 421–436 (2018)
12. Rusu, A.A., Vecerik, M., Rothorl, T., Heess, N., Pascanu, R., Hadsell, R.: Sim-to-real robot learning from pixels with progressive nets. In: International Conference on Robotics Learning (CoRL) (2017)
13. Singh, A., Yang, L., Levine, S.: GPLAC: generalizing vision-based robotic skills using weakly labeled images. CoRR **abs/1708.02313** (2017)
14. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1653–1660 (2014)

15. Tzeng, E., Devin, C., Hoffman, J., Finn, C., Abbeel, P., Levine, S., Saenko, K., Darrell, T.: Adapting deep visuomotor representations with weak pairwise constraints. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2017)

16. Zuo, Y., Qiu, W., Xie, L., Zhong, F., Wang, Y., Yuille, A.L.: Craves: Controlling robotic arm with a vision-based economic system. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4214–4223. IEEE, CA, USA (2019)


Average area captured vs Intensity level
Area
Avg area
100000
80000
60000
40000
20000
0
0.2
0.4
0.6
0.8
1.0
Intensity level
Average area captured


INPUT LAYER
FRAME DIFFERENCE COMPUTATION LAYER
OBJECT IDENTIFICATION LAYER
ESTIMATION LAYER
OUTPUT LAYER
PREDICTION LAYER
N PREVIOUS FRAMES
Current Frame
DO1
DO1
DO1
DO2
DO2
DO2
CONSECUTIVE
CURRENT TO ALL
Object Finder FN
Object Finder F2
Object Finder F1
PATH ESTIMATOR
TRUST FACTOR COMPUTATION
HIGH?
LOW?
OUTPUT VELOCITY AND TORQUE
PATH PREDICTOR
PREVIOUS PATH


Input Video Sequence
N Previous Frames of Video Sequence
Estimating the Position of All the Joints in The Image Hand with Deep CNN
Smoothening the Estimation with Refiner Network
Estimating/ Predicting The Trajectories of All The joints
Convert Motion Information to Physical Torque and Velocity
Vision to Motion Optical Network with Multiple Joints