A Wireless Mosaic of Eyes to Support Navigation and Control of Mobile Robots

P. Jiang, Z. Feng, Y. Cheng, Y. Ji, J. Zhu, X. Wang, F. Tian, J. Baruch

***Abstract*—Research on intelligent vehicle and mobile robot navigation has focused mostly on the development of a large and smart "brain" in order to gain autonomous capability copying homo sapiens. The approach is, however, facing a reliability problem and a computational bottleneck due to uncertainties in any dynamic environment. This paper reports an intelligent environment with a mosaic of wireless camera eyes to support navigation and the control of mobile robots. The mosaic of camera eyes distributes the massive on-board intelligence required for autonomous systems to the environment. A robot with less intelligence can exhibit sophisticated mobility. The solution reported here uses a multiple Bloom-filter for the efficient storage of routing information and an active contour based scheme for path planning, trajectory generation, and motion control. A prototype intelligent environment consisting of 30 wireless visual sensors was developed for indoor navigation. The integrated experiments demonstrated the mobility of an environment-controlled wheelchair.**

***Index Terms*—Networked Robots, Mobile Robot Navigation, Sensor Networks, Visual Servoing.**

I. Introduction

Autonomous navigation is a traditional research topic in intelligent robotics and vehicles, which requires a robot to perceive its environment through on-board sensors, such as cameras or laser scanners, to enable it to drive to its goal. Most research to date has focused on the development of a large and smart "brain" to gain autonomous capability for robots[1][2].

Manuscript received April 2, 2009. This work was supported in part by the Royal Society of the UK, the National Nature Science Foundation of China, and the National High-tech Research and Development Program of China.

P. Jiang is with the School of Informatics, the University of Bradford, Bradford, BD7 1DP, UK (corresponding author: phone: 0044 1274 233940; fax: 0044 1274 236600; e-mail: p.jiang@ bradford.ac.uk).

Z. Feng is with the Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, 710049 China(e-mail: fzr9910@xjtu.edu.cn).

Y. Cheng is with the School of Engineering, Design and Technology, the University of Bradford, Bradford, BD7 1DP, UK(e-mail:y.cheng4@bradford.ac.uk)

Y. Ji is with the Department of Information and Control, Tongji University, Shanghai, 200092 China(e-mail: jyx851110@hotmail.com).

J. Zhu is with the Department of Information and Control, Tongji University, Shanghai, 200092 China(e-mail: zhujintj@ tongji.edu.cn).

X. Wang is with the Department of Information and Control, Tongji Univeristy, Shanghai, 200092 China(e-mail: dawnyear@tongji.edu.cn).

F. Tian is with the Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, 710049 China(e-mail: ftian@xjtu.ac.edu.cn).

J. Baruch is with the School of Informatics, the University of Bradford, Bradford BD7 1DP, UK, (e-mail:john@cyber.brad.ac.uk)

There are three fundamental questions to be answered by an autonomous mobile robot: "Where am I going?", "Where am I?", "How do I get there?". In order to answer these basic questions, a robot requires massive spatial-memory and considerable computational resources to accomplish perception, localization, path planning, and control. It is not yet possible to deliver the centralized intelligence required for our real life applications, such as autonomous ground vehicles and wheelchairs in care centers.

In fact, most autonomous robots try to mimic how humans navigate, interpreting images taken by cameras and then taking decisions accordingly. They may encounter the following difficulties.

Image processing and scene interpretation

The projection of a 3D world onto a 2D image plane through a pin-hole camera (camera eyes used by vertebrates) makes the motion analysis inherently ill-posed, for example the rotation and translation are commingled in an inseparable way[3] so that a large set of different motions can induce very similar optical flows. This causes difficulties for efficient and reliable perception. To understand the semantics of a scene and to become location-aware from images is difficult for mobile cameras[4]. A slight movement may result in a significant change in the viewed images, due to discontinuities or obstructions in a scene. There is also much less control over illumination and background scene clutter compared with stationary applications.

Movement control

The eye-in-head configuration introduces uncertainties from the changing views of moving eyes, so that the changes of illumination due to the environment and robot motion are often commingled together, which cause difficulties for visual tracking and robot motion control[5].

Behavior coordination

Current approaches need to maintain a centralized behavior network[2]. The difficulty appears to be that various behaviors are merged together. There is no guarantee that a goal can be reached. Modification of the network and the prediction of consequences are both difficult.

Localization

The current approaches rely on the fusion of odometry with on-board sensing based on a map or on a bilateral technique registering map data, i.e. SLAM(Simultaneous localization and mapping)[6]. Uncertainties involved in initial location, mapping, odometry and sensing have a significant influence on the

accuracy and reliability of localization. Usually, it is necessary to take a probabilistic approach to identify the belief in an estimation, from single-hypothesis to multi-hypothesis[1]. Poor localization precision may often occur due to sensing errors or imperfect landmarks, e.g. for global localization and the kidnapped robot problems[7][8].

Path planning

Current approaches usually abstract a geometric environment into a topological map based on visual landmarks. Planning is carried out on this topological map[1]. A planned route has to be converted back to the geometric space for continuous motion control. It introduces ambiguity and complexity in the linkages between the planning space and the control space.

In order to overcome these difficulties, a system with cameras distributed in the environment and connected by wireless communication is presented in this paper. It releases the massive requirement for centralized computation into a distributed camera network for the entire environment. These cameras support the navigation of vehicles or robots with a lower level of on-board intelligence. It is economically feasible in applications where a large number of vehicles need to be controlled in a certain area, such as navigation of wheelchairs in a care centre, trading environment intelligence for expensive on-board intelligence in every robot.

Wireless Sensor Networks provide the IT infrastructure for an intelligent environment, which could lead to a new paradigm for navigation. If intelligence can be pervasive in an environment[9], the complexity of the environment, which causes the reliability problem and hinders the broad application of autonomous robots, will be greatly simplified. Initially, a distributed sensor network can provide a topological map of the environment. The *routing* to a geographic goal becomes querying a sensor sequence from the sensor network. The distributed sensors then become active landmarks for robot *localization*, which is more reliable and efficient than the on-board sensors. Finally with a static camera configuration mounted in the environment rather than on a mobile robot, the intelligent environment will facilitate both *perception* and *control*. Each sensor will be in charge of a local region and therefore will face less uncertainty and exhibit higher reliability. This current research into wireless sensor network-based robot navigation took a hierarchical approach. Distributed sensors were coordinated for high level activities only, for example localization[11], routing[12], or event driven behavior coordination[13]. A significant number of tasks of low level perception and control were left to the robots. This paper presents an efficient scheme for unifying routing, path planning, trajectory generation and motion control for distributed wireless sensors. It is a complete navigation solution developed in a project, “Wireless Mosaic Eyes to Support Robot Navigation in an Intelligent Environment(WiME)”. Individual sensors in a wireless sensor network usually have very limited fields of view, ad-hoc communication, limited memory space, computational capacity, and power on-board[9][10]. This paper includes two mechanisms for the coordination of distributed cameras with limited on-board resources for robot navigation: multiple Bloom-filter based routing and snake based navigation.

The paper is organized as follows: in section II, the hardware and software platforms of the WiME system are introduced; section III presents in summary a multiple Bloom-filter and error expectation based design for robot routing; snake based path planning and predictive control are presented in summary in section IV; section V presents experiments using the newly developed techniques; section VI forms a conclusion.

II. System Architecture

With dispersed visual sensors, a prototype of the WiME system has been developed to support wheelchair and robot navigation inside a building, as shown in Fig. 1. The system consists of three hardware systems, 1) networked wireless visual sensors, 2) wireless controlled robots, and 3) a remote console.

The networked wireless visual sensors shown in Fig.1 are the main elements which implement the image processing, the localization, and the robot navigation. They are developed from the Intel IMote2 main microcontroller board. The Intel IMote2 is an advanced sensor network platform with built-in IEEE 802.15.4 radio transceiver CC2420. The sensor board is developed with an OV7620 visual sensor module, which is a highly integrated interlaced/progressive scan CMOS digital video chip with resolution 640×480. The visual sensor board is connected with the IMote2 main controller through DMA (Direct Memory Access) to enable high speed data throughput.

In the project, we modified an off-the-shelf wheelchair to be controlled by an Atmega128L processor. It receives commands from the wireless visual sensors, measures motor speeds from two encoders attached to the wheel shafts, drives and steers in response to commands, as shown in Fig.1. It also connects with sonar bumper switches for emergent survival control. A human user can acquire a navigation service through a PDA, which sends the destination to the WiME network to trigger a navigation service.

The remote console is responsible for offline configuration,

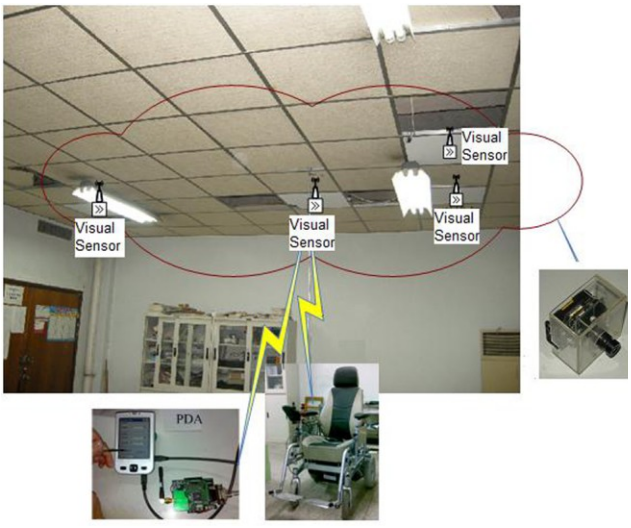

Fig. 1. WiME intelligent environment

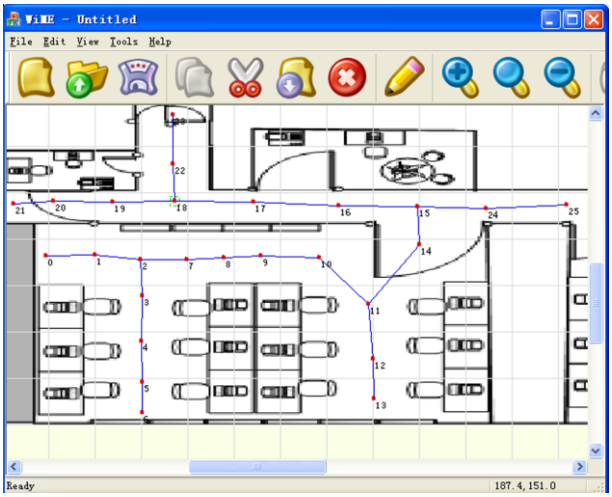

Fig. 2. Map configuration of the WiME, wireless eyes are in red and communication links are in blue.

online navigation requests and monitoring. The remote console can be either a PC or a PDA with an IEEE 802.15.4 adaptor.

In order to provide unambiguous semantics and assign a determinant role to each distributed visual sensor, a configuration software kit was developed in the WiME project. It runs on the remote console and has 3 functional blocks: map configuration, visual context configuration, and data download.

The map configuration provides the tools for creating a geographic map and locating the wireless sensors. It links a geometric map with a sensor topological map. An example of the map configuration for an indoor environment is shown in Fig.2.

The visual context configuration provides tools for camera calibration and visual semantics specification. The meta-data model is shown in Fig.3. It links the topological map of the wireless sensor network with visual clues in the image planes for navigation. Regions of passages in the image plane, overlap regions with other visual sensors, static obstacles on the passages etc. are designated here. Calibration of the homography matrix for each visual sensor is carried out to map

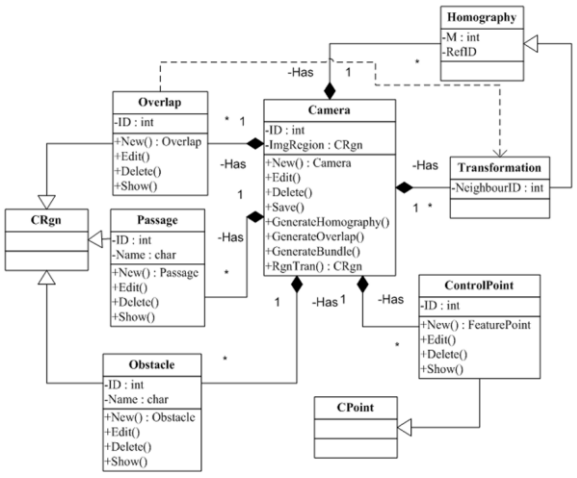

Fig. 3. Meta-data model of visual context configuration

the floor plane from the image plane.

The data download software binds the data generated by the map configuration and the visual context configuration into a data package for each node in the wireless sensor network. It assigns the physical address of a wireless sensor and downloads the data package into it. With the aid of the data package, all wireless visual sensors are organized into a purposed network and each visual sensor is provided with unambiguous semantics and clear visual clues for robot navigation, e.g. this is a camera viewing "John's office" and "linking the D-wing corridor" with "region A" in the image as a passage.

Embedded navigation software has also been developed so that each wireless visual sensor can support vision based object tracking and robot navigation. The embedded navigation software runs on an embedded engine, Linux kernel 2.6.14. It captures video at 30 fps continuously. The visual tracking of a robot is carried out by using the Posterior Probability Measure based template match proposed by the authors[14]. A multiple Bloom-filter routing function and a snake based navigation function are implemented for driving the robots using the distributed visual sensors. This is discussed in the following sections.

III. Multiple Bloom Filters For Distributed Routing

For navigation, global routing is the first task. It is located on the top of the tiered navigation architecture [1]. Given a goal, a robot needs to know a sequence of abstracted locations which can lead it to the goal with the least cost. It is traditionally implemented by searching in a topological map.

A. Routing Tables for Navigation

In the map configuration, a WiME network has been configured as a topological map of visual sensors deployed in the environment. When a vehicle is under the control of a sensor node, it will query the node to determine which edge to follow in order to reach the destination. The query will use a human friendly identity such as "John's office" rather than an artificial code. This means that a user without knowledge of the coding method can query the system to request a navigation service. It also simplifies the program in PDAs, without the necessity to translate a human friendly query into an internal code.

Due to the limited capacity of a wireless node, an on-line shortest route search has to be avoided. Therefore, we take a routing table approach, where routing tables are generated for each node off-line by the Dijkstra search algorithm which is a greedy algorithm to calculate the shortest path from a node to any other node in a topological graph with or without loops. The off-line search algorithm will generate a routing table for each edge from a node. The routing table corresponding to an edge records the set of nodes that can be reached, by following the edge, with the shortest distances.

With the routing tables saved in each node, a robot at the location of a node can query which passage it should take in order to reach a destination, e.g. "meeting room", travelling the least distance. The sensor node can answer by determining if the string of the destination belongs to a routing table. It sacrifices

memory to avoid on-line searching. However, for a big map, the required memory for storing such routing tables could be huge and searching for membership could be computationally intensive. This is often not feasible for sensor nodes with limited on-board memory and limited computational power. In this section, multiple Bloom filters are introduced as a general solution for wireless sensor network applications, although the IMote2 used in this project does not have serious memory concerns.

B. Multiple Bloom Filters for Navigation Routing

Bloom-filters are an efficient and lossy way of describing the membership of elements belonging to a set[15]. They are broadly used for the compression of membership in many fields, such as spell-checkers[16] and dictionaries of passwords[17]. In order to store n elements belonging to a branch path, a routing table T can be implemented by a Bloom-filter that is a bit-vector of length m. Each element can be a human friendly string. To represent n random strings in table T, k independent hash functions are used to generate digital fingerprints in T, which map the strings belonging to the path to k integers in [0,m] and the corresponding bits in bit-vector T are set as shown in Fig. 4. If there are L branch paths for a node, L Bloom-filters are needed for routing, called a *multiple Bloom-filter* in this paper.

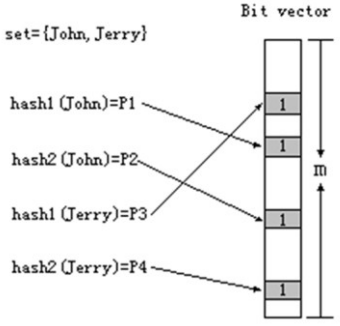

Fig. 4. A routing table for a path with two members, John and Jerry.

A vehicle can query the multiple Bloom-filter in a sensor node to determine which path to take. If any of the bits in a Bloom-filter are not set, the corresponding path definitely should not be taken. If all of the bits for a path are set, the vehicle *may* take the path. There is a non-zero probability that the decision is wrong because a bit could be set by other elements when they are hashed. This is known as a *false positive*. The advantage of using Bloom-filters as routing tables can be clarified by considering its compression capability, from n elements with any string length to m bits, and its high query efficiency, from an n elements search to a k hash functions check. However, there is a trade-off between the false positive rate and the compression rate, which is relevant to element number n, hash function number k and vector length m[18]:

$$p_{fp} \approx (1-e^{-nk/m})^k \tag{1}$$

The probability of false positives can be reduced by a multi-hop approach, where a query is forwarded to the next node in the path for double checking the membership.

In order to design L Bloom-filters, an intuitive approach is to select the table lengths $m(i)$, i=1…L, to meet a common false positive rate p_{fp} for all branch paths based on (1). However, it will result in different false positive errors for different edges. In a multiple Bloom-filter, the encoded numbers of nodes $n(i)$, i=1,…,L, for individual Bloom-filters could be quite different, e.g. some edges may lead to very few nodes but others may lead to a lot. However, all Bloom-filters or sub-tables have to pass equal numbers of queries, which could be legal queries or illegal queries for an edge, e.g. a query belonging to another table. Equal p_{fp} means equal possibility of errors for all paths, however many nodes there are in a path. As a result, a path with fewer nodes will have a higher chance of being wrong than a path with dense nodes. This can be verified by examining the relative error expectations. Consider when there are altogether N possible queries, e.g. a possible query consisting of 6 digits could have N reach 10^6, the relative error expectation for edge i due to N queries is

$$E_{EPN}(i) = \frac{(N-n(i))\cdot p_{fp}}{n(i)} \tag{2}$$

Therefore, we need to take the error expectation as the design criterion instead of the false positive rate in order to achieve a uniform relative error expectation for all paths. A biased false positive rate, $p_{fp}(i)$, i=1..L, is proposed for different edges i by taking into account $n(i)$:

$$p_{fp}(i) = p_{fp}/t \quad \text{with} \quad t = \frac{N-n(i)}{N-\bar{n}} \cdot \frac{\bar{n}}{n(i)} \tag{3}$$

where $\bar{n} = n/L$.

From (3), if $n(i) \le \bar{n}, t \ge 1$ and $p_{fp}(i) \le p_{fp}$, a lower false rate is set ; if $n(i) > \bar{n}, t < 1$ and $p_{fp}(i) > p_{fp}$, a higher false rate is set. Then all edges exhibit equal relative error expectation rates regardless of the number of nodes in an edge:

$$E'_{EPN}(i) = \frac{(N-n(i))\cdot p_{fp}(i)}{n(i)} = \frac{(N-n(i))}{n(i)} \cdot \frac{p_{fp}}{t} = \frac{N-\bar{n}}{\bar{n}} \cdot p_{fp}$$

It can be proved[26] that a multiple Bloom-filter using the error expectation based design will not only achieve uniform relative error but will also save memory space in comparison to the conventional design based on the false positive rate.

IV. Active Contour Based Path Planning and Control

After obtaining discrete routing information, an admissible and safe path has to be generated for a robot to travel from one sensor node to another, compliant with all kinematic and dynamic constraints.

Active contour models[19], also termed snakes, are techniques broadly used in computer vision for image segmentation and contour tracking. The determination of the presence of an object depends not only on the image details at a specific point, but also on the properties of an object's shape. Similar concepts have been applied to path planning, such as elastic bands[20] and virtual springs[21]. A snake is defined as a flexible entity that is deformable by applying internal and external forces, which can be represented in the configuration space of a robot as an admissible path. The deformation of a

snake body is caused by the interaction of adjacent joints, it appears to be suitable for distributed implementation since it requires only adjacent information exchange. The information flow along a snake evolves to make it energy-optimal as a whole and constraint compliant locally. However existing methods have been presented for centralized off-line planning and their potential for on-line distributed applications in an intelligent environment have been ignored. Paper[22] proposed a snake based controller for the correction of the tracking paths of wheelchairs. It uses only on-board sensors and therefore is a local motion controller. In fact, sensors distributed in an environment provide the infrastructure to consider both global path planning and local on-line control, which can deal with dynamic changes in the environment and predict future uncertainties more effectively.

A snake based path planning and predictive control scheme is proposed in this section as a unified mechanism, which dynamically evolves as a path distributed amongst the mosaic of wireless eyes for reaching a destination, for obstacle avoidance and for smooth navigation. Both kinematic and dynamic constraints are taken into account.

A. *Snake as a Path Planner*

Let $p_i=(x_i, y_i)$ be Euclidian coordinates of a point in a 2D configuration space. For a positive integer n, $\{p_0, p_1, \cdots, p_n\}$ denotes a sequence of configurations in the 2D space from p_0 to p_n. A snake is a curve connecting adjacent configurations p_i, $i=0..n$, sequentially. Each configuration p_i is called a control point, which can be moved by exerted internal and external forces from adjacent control points and obstacles.

The total energy of a snake can be expressed as

$$E_{snake} = E_{\text{int}ernal} + E_{external}, \tag{4}$$

where the $E_{internal}=E_{elastic}+E_{curvature}$ is concerned with the constraints of the snake; the external energy $E_{external}=E_{obstacle}$ is concerned with the obstacles on the way.

The snake should evolve to minimize its energy along the negative energy gradient direction with boundary conditions of

$$\begin{aligned} &\text{Initial condition}: p_0 = p_r(t); \theta_0 = \theta_r(t); \\ &\text{Terminal condition}: p_n = p_g. \end{aligned} \tag{5}$$

where $(p_r(t), \theta_r(t))$ is the robot position and orientation sampled at instant t by a visual sensor. The proposed snake based path planning is to evolve n control points p_i dynamically for maintaining clearance of m obstacles q_i, satisfying a curvature constraint and a minimal path-length. In addition, the initial boundary conditions in (5) must meet the nonholonomic constraint of a wheeled vehicle. The terminal condition ensures that it reaches its goal. Due to the multiple constraints involved, a flexible snake may change to a rigid or even to a broken state. A state machine is developed to coordinate the state switch[23].

Fig.5 (and the video of *snake.wmv*) demonstrates how a snake responds to a moving obstacle. When a dynamic intruder is detected by a visual sensor, control points start moving, where the yellow and red circles indicate the increased curvatures as shown in Fig.5b. The whole snake is deployed into several visual sensors as a global path planner; the control points interact with adjacent points across cameras via the wireless communication protocol. Therefore, a snake becomes a coordination mechanism, among distributed wireless sensors, for navigation.

B. *Predictive Control for Snake Tracking*

The snake provides a reference path for a robot to travel along, but the actual control of the robot to follow the path is another difficult task, which involves trajectory generation and motion control. Due to the limited field of view of the on-board sensors for the autonomous robots, the tracking speed has to be controlled conservatively in order not to compromise safety. The long range sight provided by a visual sensor network provides the potential for optimal control of vehicles, responding to distant changes long in advance and, therefore,

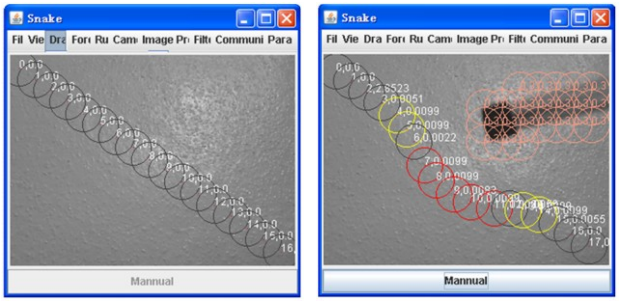

Fig.5a. Initial snake Fig.5b. Responding to an obstacle

making it possible to drive with optimal speed or energy use in a dynamic environment. Model Predictive Control is a technique to achieve optimal control by predicting future system behaviours[24]. One advantage of model predictive control is its ability to handle constraints[25]. Its open-loop format also provides the ability to control a system with a longer sample period. Navigation using a wireless sensor network needs to deal with motion constraints and slower sampling rates due to the limitations of wireless communication. As a result, predictive control becomes a powerful solution.

Define a rolling window of length l along a snake path for a robot. For every sample period, optimal control is calculated by taking into account the dynamic constraints and the geographic features of the snake path in the window. The l-window rolls forward one step at a time and makes the next set of predictions. Working in this way repeatedly, a vehicle can react to possible hazards on its route long in advance and use its driving capacity effectively. This is formulated as the following optimization problem for predictive control:

$$\begin{aligned} &\min_{F,\tau}(T) = \min_{F,\tau}\left(\int_0^l 1/v(s)ds \Big|_{s\in A} \right); \\ &\text{boundary conditions: } v(0) = v_{r0}, v(l) = 0; \\ &\text{subject to } \begin{cases} |f| < (\mu N)_{\max} & \text{(non-slippage)} \\ |F| < F_{\max} & \text{(limited driving force)} \\ |\tau| < \tau_{\max} & \text{(limited steering torque)} \end{cases} \end{aligned} \tag{6}$$

where A is the snake path to be followed; $v(s)$ is the velocity profile of the robot to be optimized; v_{r0} is the sampled velocity

of the robot at an instant; f is the friction of tires with coefficient μ and normal force N; F and τ are the driving force and steering torque of the robot with upper bounds of F_{max} and τ_{max}, respectively. Optimization of the objective function (6) minimizes the robot travelling time T along the snake from its current location to the end of the l-window. The robot should be able to stop at the end, $v(l)=0$, to cope with the worst circumstances not seen in the rolling window. An optimal velocity profile $v(s)$ along the snake will be obtained and sent to the vehicle to follow for this sampling period, which may vary due to ad-hoc communications.

V. Experiments

We have developed a complete solution for robot navigation using distributed visual sensors or a mosaic of eyes. It includes a multiple Bloom-filter for routing, a snake algorithm for path planning, and a predictive control algorithm for trajectory generation and motion control. This is different to the mainstream of robot navigation research that uses on-board sensors and computation; this solution aims to effectively utilize intelligence placed in the environment for the navigation of low intelligence robots.

A. Multiple Bloom Filter for Routing

In order to evaluate the performance of the proposed error expectation based multiple Bloom-filter design in comparison with the conventional false positive rate based design, we randomly generate a topological map with altogether n=1000 nodes. For k=4 hash functions and a desired false positive rate p_{fp}=0.01, false positive rate based and error expectation based multiple Bloom-filters are developed. For a node with four branches(Path1, Path2, Path3 and Path4), which have 23, 193, 332 and 452 nodes, respectively, the two multiple Bloom-filters are queried by four groups of 10^6 random strings. Table 1 and Table 2 show the experimental results.

Table 1: Errors Using The False Positive Rate Based Design

Group	*1*	*2*	*3*	*4*	*Average Errors*	*Relative Errors*
Path 1	9987	10013	9979	10025	10001	434.83
Path 2	9976	9984	9981	10012	9988	51.75
Path 3	10014	9985	9998	10002	9999	30.12
Path 4	9982	9986	10011	9993	9993	22.11

Table 2: Errors Using The Error Expectation Based Design

Group	*1*	*2*	*3*	*4*	*Average Errors*	*Relative Errors*
Path 1	937	945	914	909	926	40.26
Path 2	7702	7734	7724	7727	7722	40.00
Path 3	13178	13297	13206	13308	13247	39.90
Path 4	17882	18123	18105	17934	18011	39.84

From Table 1, the conventional false positive rate based design results in many more relative errors for a path with fewer nodes than a path with more nodes, for example, path 1 exhibits a 434.83 relative error. From the working process of a multiple Bloom-filter proposed in III, any query needs to be checked by all the Bloom-filters. Therefore, a path with fewer nodes will suffer more errors from illegal queries. The errors will cause more than one Bloom-filter to pass the hash check. The multi-hop checks have to be carried out for further confirmation as described in III.B, which increases the communication cost in the wireless network. The proposed expectation based design guarantees a uniform distribution of relative errors. In Table 2, all paths show similar numbers of relative errors, about 40, that generates equal risks for a query.

In terms of overall memory usage, we randomly generate 1000 topological maps with 1000 nodes each. The table lengths by using the false positive rate based design and the error expectation based design are shown in Figs. 6.

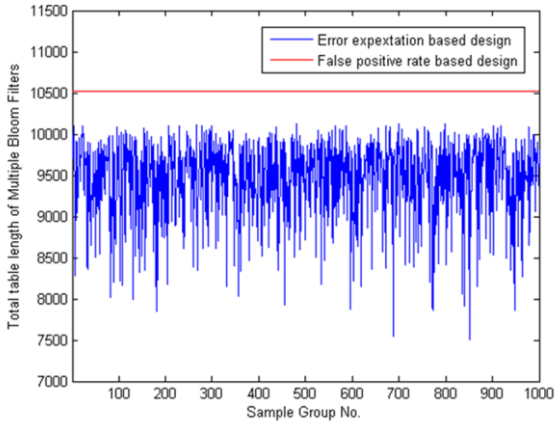

Fig. 6. Total table lengths of the two methods

It is clear that the proposed method outperforms the traditional design. In order to have a quantitative view, we list the memory usage of the Bloom-filter based routing tables comparing with the routing tables without using Bloom-filters for the 4 groups of results in Table 3.

Table 3: Memory Usage of Routing Tables(Kbits)

Group	*1*	*2*	*3*	*4*	*Percentage*
No Bloom-filter	48	48	48	48	100%
FPB Bloom-filters	10.52	10.52	10.52	10.52	21.9%
EEB Bloom-filters	9.449	9.452	9.483	9.462	19.7%

From Table 3, the memory usage is reduced to 21.9% by introducing Bloom-filters as routing tables, where the Bloom-filters are designed to achieve equal false positive rates. The relative memory usage is further reduced to about 19.7% by using the new error expectation based design.

B. Path Planning and Predictive Control

Trajectory tracking of a robot controlled by the mosaic of eyes is investigated with an experiment. The robot is a wireless controlled model car. Four eyes are mounted on the ceiling in a manner that each has two neighboring eyes to form a closed running loop. The predictive control is designed with a rolling window of l=20(control points). The maximum speed is 0.8(m/s). The robot with a mass of 0.56(kg) has its maximum

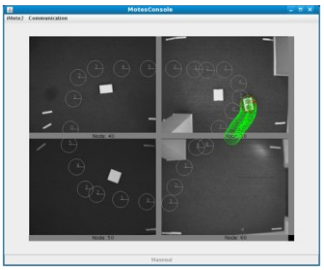

Fig. 7a. Snake without obstacle

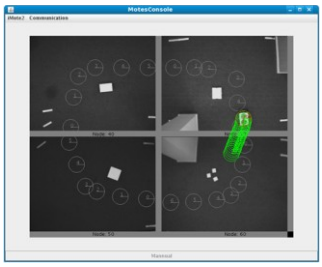

Fig. 7b. Obstacles appear

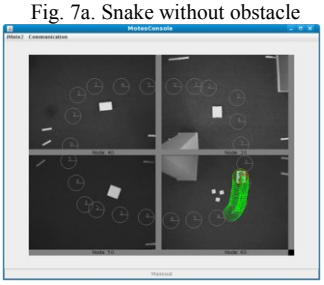

Fig. 7c. Passing obstacle area

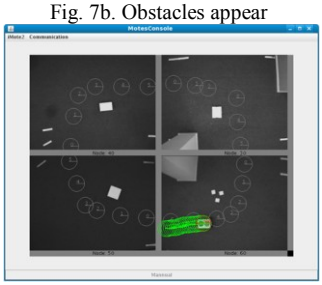

Fig.7d. Passed obstacle area

driving force F_{max}=4.4(N), τ_{max}=2.0(N.m) and friction coefficient μ_{max}=0.6. The parameters are selected empirically. In general, we adjust the maximum speed, force and torque to produce fast and smooth tracking.

Fig.7 shows four real-time images captured by the four eyes during the experiment. The four eyes arranged in a square have IDs of 30, 40, 50 and 60 from the top-right anti-clockwise. In Fig.7a, the robot is controlled by eye 30(top right) and is heading towards the control area of eye 60(bottom right). The light grey circles represent a dynamic snake-track for the robot to follow. The green circles indicate the *l*-window used for predictive control. The white rectangular blobs are dynamic obstacles. As one can see in Fig.7a, dynamic obstacles exist in the view of eyes 30, 40 and 50 but not in eye 60. After 0.5 seconds shown in Fig.7b, a triple obstacle is detected by eye 60. Although the robot is controlled by eye 30, which cannot see the new obstacle, the snake is updated to an obstacle-free path through internal data exchange with eye 60. Robot control privilege is handed over from eye 30 to eye 60 in Fig.7c, where the robot is passing the triple obstacle area. Fig.7d shows the robot after it has passed the triple obstacle successfully returning to its original snake track.

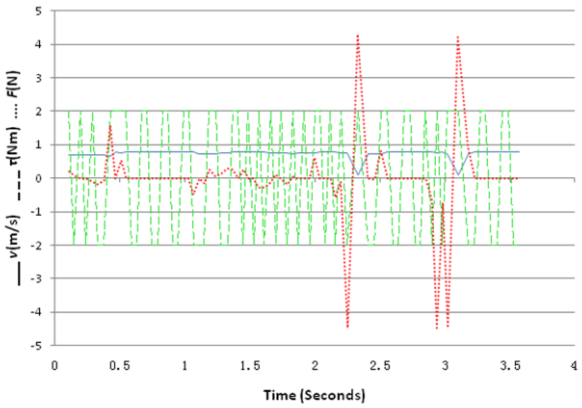

Fig.8. Robot control and velocity

The corresponding control signals and robot velocity are shown in Fig.8. It shows that predictive control keeps the driving force and steering torque within the admissible ranges. The alternating signs of the steering torque indicate the feedback regulation of the predictive control required to follow the snake track, even though the predictive control is designed in an open-loop format for each single prediction in the *l*-window. The robot speed is about 0.76(m/s) in Fig.7.a. Because there is no obstacle in the view of eye 60, the path generated is a straight line. When the obstacle is detected by eye 60, the snake track changes as in Fig.7b. The robot maintains its high speed when crossing the overlap area between eye 30 and eye 60. When the robot approaches the triple obstacle after 2.3 seconds, it decelerates to avoid skidding(Fig.7c). It accelerates again after passing the triple obstacle in Fig.7d.

C. *Integrated Experiments*

A practical WiME system has been implemented. Altogether 30 wireless visual sensors are mounted on the ceiling in a building with a topology shown in Fig. 2. Software packages and communication protocols were developed for wheelchair navigation using the proposed algorithms. The only processor on-board the wheelchair is an 8-bit Atmega 128L, which is used to link the WiME network wirelessly through the IEEE 802.15.4 protocol and to drive the two differential wheels via two PWM signals. With such a low-performance processor on-board, the high level functions, such as localization and path planning in the tiered architecture[1] cannot be implemented. However, the distributed environment intelligence developed in this paper can control such a low intelligence wheelchair with superior mobility, see the video *MovieWiME.wmv*. A communication layer was developed on the top of the IEEE 802.15.4 short frame protocol for the coordination of wireless sensors and wheelchairs. There are five sets of commands defined, as shown in Fig. 9:

1) Control points exchanging commands: coordinates of control points are exchanged between adjacent wireless sensors for force calculation to deform the snake as in IV.A.

2) Control token commands: at a specific time, only a single sensor node with the token can control a robot as the coordinator. The commands are used to broadcast the ownership of the token periodically or initiate a token handover procedure.

3) Control commands: the vision sensor with the control token will send the velocity profile obtained from the predictive control to the robot for tracking, as shown in IV.B.

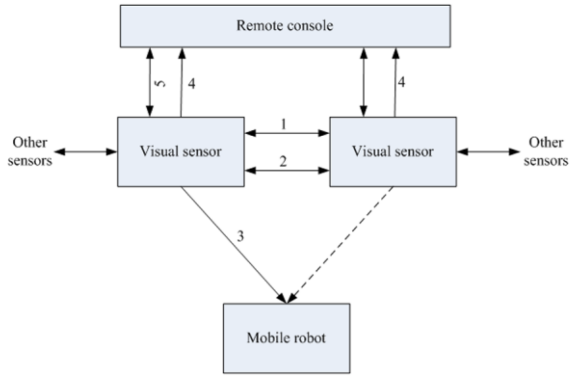

Fig. 9. Communication protocol

4) Monitoring commands: the commands are used to send out current information, for example control points and detected obstacles, to a remote console for the purpose of monitoring.

5) Service request commands: these commands are from a remote console, such as a PDA, to request a navigation service to a destination. The route to the destination can be obtained from the multiple Bloom-filters as presented in III.

In the experiments, the system achieves a performance of request -response time of less than 0.5 seconds, average moving speed of about 0.3m/s, and a maximum control error of less than 0.2m. The control error considered here is the deviation of the robot from the snake which produces a dynamic path responding to unknown moving obstacles. This is higher than the conventional tracking error with a static path because of unexpected intruders and a longer control latency. The wheelchair is driven by following a velocity profile obtained from the predictive control for each sample period. If an intruder is detected by a visual sensor during this period, the snake will be updated and may produce a bigger error in the next control period. As a result, the maximum error is dependent on the control latency and the robot speed. For behaviors requiring higher accuracy, such as passing a door, the snake will be constrained with less flexibility in that area and the predictive control will automatically generate a lower speed to guarantee a safe passage, see the generated velocity profiles at about 2.4 seconds and 3.1 seconds in Fig.8 for passing obstacles.

The time spent on individual tasks by a visual sensor module was recorded in Fig.10. The sampling period can be varied between 260 and 400 ms, depending on whether this visual sensor is controlling the robot. The visual sensor in control of the robot does not increase the processing time for reading a frame, extracting obstacles and adjusting the snake but increases the time for trajectory generation using predictive control, communication token processing and sending the snake to the other nodes. The time for extracting the foreground also increases with more time needed to distinguish the robot from the background.

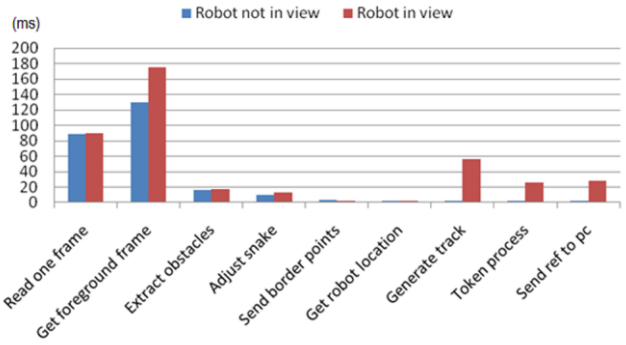

Fig.10. Time spent on tasks

VI. CONCLUSIONS

This paper has presented the WiME architecture and an effective solution for navigation using pervasive intelligence. The wireless visual sensors in the WiME were provided with unambiguous semantics for routing, control and image processing to support the navigation of "non-intelligent robots". This paper includes a multiple Bloom-filter technique for storing large amounts of routing information in a wireless sensor node where memory is a scarce resource. The multiple Bloom-filter is based on an error expectation design, instead of the conventional false positive rate based design, it achieves a uniform relative error distribution and uses less memory. The second contribution is to use snake based predictive control for robot navigation with distributed vision, where a snake is a concise and unified mechanism to embrace all the navigation components: from path planning, trajectory generation to motion control, dealing with kinematic and dynamic constraints. In addition, a snake is an efficient way to self-organize distributed visual sensors for realizing both global behavior and local reactive behavior. The method is presented as a general approach for navigation using a wireless sensor network, considering limited on-board memory space and computational power, although they are less critical in the prototype system. Experiments proved that the proposed distributed cameras can be used to enhance the mobility of service robots, such as indoor wheelchairs for aged and disabled people.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and editors for their constructive and helpful comments.

REFERENCES

[1] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge: USA: MIT Press, 2004.
[2] R. C. Arkin, *Behavior-based Robotics*. Cambridge: MIT Press, 2000.
[3] J. Neumann, C. Fermuller, and Y. Aloimonos, "Eyes from eyes: new cameras for structure from motion," presented at IEEE Workshop on Omnidirectional Vision, pp. 19-26, 2002.
[4] G. N. DeSouza and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 237-267, 2002.
[5] F. Chaumette and S. Hutchinson, "Visual servo control, part 1:Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 82-90, 2006.
[6] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM), Part I: The essential algorithms," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 99-108, 2006.
[7] S. I. Roumeliotis and G. A. Bekey, "Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization," presented at Proc. IEEE International Conference on Robotics and Automation (ICRA-2000), pp. 2985 - 2992, San Francisco, CA, 2000.
[8] S. Thrunn, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, pp. 99-141, 2001.
[9] D. Estrin, D. Culler, and K. Pister, "Connecting the physical world with pervasive networks," *IEEE Pervasive Computing*, vol. 1, pp. 59-69, 2002.
[10] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communication Magazine*, vol. 40, pp. 102-114, 2002.
[11] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Relaxation on a mesh: A formalism for generalized localization," presented at Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), pp.1055-1060, Wailea, Hawaii, 2001.
[12] Q. Li and D. Rus, "Navigation protocols in sensor networks," *ACM Trans. on Sensor Networks*, vol. 1, pp. 3-35, 2005.

[13] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. S. Sastry, "Distributed control applications within sensor networks," *Proceedings of the IEEE*, vol. 91, pp. 1235-1246, 2003.
[14] Z. Feng, N. Lu, and P. Jiang, "Posterior probability measure for image matching," *Pattern Recognition*, vol. 41, pp. 2422-2433, 2008.
[15] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, pp. 422-426, 1970.
[16] M. D. McIlroy, "Development of a Spelling List," *IEEE Transactions on Communications*, vol. 30, pp. 91-99, 1982.
[17] E. H. Spafford, "Opus: preventing weak password choices," *Computer and Security*, vol. 11, pp. 273-278, 1992.
[18] M. Mitzenmacher, "Compressed bloom filters," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 604-612, 2002.
[19] M. Kass, A. Witkin, and D. Terzopoulos, "Snake: Active contour models," *International Journal of Computer Vision*, vol. 1, pp. 321-331, 1988.
[20] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," presented at IEEE Int. Conf. Robotics and Automation, pp. 802-807, Atlanta, USA, 1993.
[21] S. Cameron, "Dealing with geometric complexity in motion planning," in: K. Gupta and A. P. del Pobil (eds), Practical Motion Planning in Robotics, Wiley, New York, pp. 259-274, 1998.
[22] L. J. Zhou, C. L. Teo, and E. Burdet, "A nonlinear elastic path controller for a robotic wheelchair," presented at 3rd IEEE Conference on Industrial Electronics and Applications, pp.142-147, Singapore, 2008.
[23] Y. Cheng, P. Jiang, and Y. F. Hu, "A distributed snake algorithm for mobile robots path planning with curvature constraints," presented at IEEE International Conference on Systems, Man and Cybernetics, pp.2056-2062, Singapore, 2008.
[24] C. E. Gacia, D. M. Prett, and M. Morari, "Model predictive control: theory and practice-a survey," *Automatica*, vol. 25, pp. 335-348, 1989.
[25] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: stability and optimality," *Automatica*, vol. 36, pp. 789-814, 2000.
[26] P. Jiang, Y. Ji, X. Wang, J. Zhu, "Design of a multiple bloom-filter for distributed navigation routing", submitted to *IEEE Transactions on Systems, Man, and Cybernetics--Part A: Systems and Humans*.