9

A Distributed Mobile Robot Navigation by Snake Coordinated Vision Sensors

Yongqiang Cheng, Ping Jiang and Yim Fun Hu
University of Bradford
UK

1. Introduction

To date research in intelligent mobile agent have mainly focused on the development of a large and smart "brain" to enable robot autonomy (Arkin 2000; Murphy 2000). They are, however, facing a bottleneck of complexity due to the dynamics of the unstructured environments. Steering away from this smart brain approach, this chapter investigates the use of low level intelligence, such as insect eyes, and combines them to produce highly intelligent functions for autonomous robotic control by exploiting the creativity and diversity of insect eyes with small nervous systems (Land & Nilsson 2002) that mimics a mosaic eye.

A mosaic eye transmits information through the retina to the insect's brain where they are integrated to form a usable picture of the insect's environment in order to co-ordinate their activities in response to any changes in the environment. Applying this concept to robotic control, the mosaic eye is used to assist a robot to find the shortest and safest path to reach its final destination. This is achieved through path planning in a dynamic environment and trajectory generation and control under robot non-holonomic constraints with control input saturations, subject to pre-defined criteria or constraints such as time and energy. By utilising pervasive intelligence (Snoonian 2003) distributed in the environment, robots can still maintain a high degree of mobility while utilising little computational functions and power. However, navigation techniques assisted by an environment with distributed information intelligence are different from conventional ones that rely on centralised intelligence implemented in the robot itself. These distributed navigation techniques need to be reconsidered and developed.

The contour snake model (Kass et al. 1988) is broadly used and plays an important role in computer vision for image segmentation and contour tracking. Similar concepts have been applied to path planning with centralised robot navigation control using onboard sensors, such as elastic bands and bubbles (Quinlan & Khatib 1993; Quinlan 1994), connected splines (Mclean 1996) and redundant manipulators (Mclean & Cameron 1993; Cameron 1998). They all require a global planner to gather and process information. On the other hand, most of the existing work for robot navigation in an environment with small scale sensor networks considers only the high level path or discrete event planning(Sinopoli et al. 2003; Li & Rus 2005), ignoring issues related to low level trajectory planning and motion control due to sensor communication delay, timing skew, and discrete decision making. Low level

functions are usually given to robots to conduct local control relying on on-board sensors. Due to the limited information provided by local sensors, tracking speed and accuracy have to be compromised.

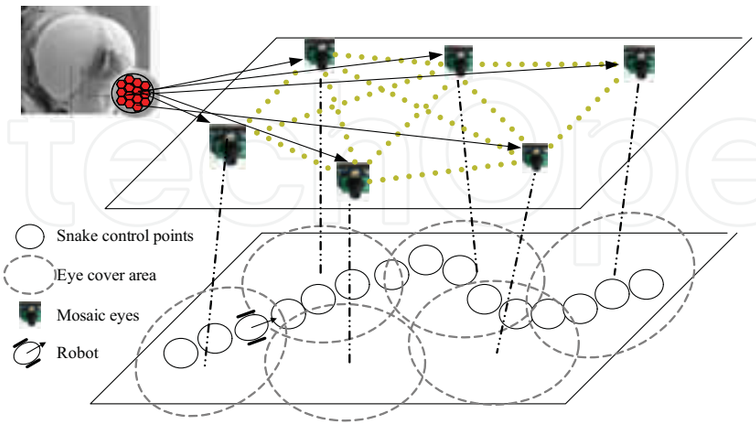


Fig. 1. Mosaic eye inspired intelligent environment

Building upon the WIreless Mosaic Eyes towards autonomous ambient assistant living for the ageing society (WiME) system(Website 2006; Cheng et al. 2008), this chapter aims at creating an intelligent environment with distributed visions for the navigation of unintelligence mobile robots in an indoor environment. The mosaic eye intelligent environment is illustrated in Fig. 1. It resembles that of insect eyes connected by neurons. The structure is formed using a set of distributed wireless vision sensors (mini-cameras, mosaic eye) mounted on a physical structure. Each vision sensor represents an eye in the mosaic eye structure. Neighbouring eyes are connected by wireless eye communications. Each eye covers a certain area of the workspace and holds a segment of the whole path. Thus, by considering a distributed sensor network, robots can be guided by a larger set of information concerning its surrounding environment to enable better predictive capability to be achieved. Predefined projection relations are calibrated in advance after mounting the mosaic eye in the structural environment. Once the start positions and target positions of a path are specified by an operator and written into the system profile files of all eyes, each eye will search its profile file and exchanges information with its neighbours for its role in the routing table, and generates a segment of the global distributed reference snake path as required. A mobile robot controlled by the mosaic eye only equips with wireless receiver and actuator without intelligent processing capability on-board. If an eye detects an abnormal situation within its coverage area, it will send a signal to inform its neighbouring eyes of this situation for their considerations in the path planning process. If an area is only covered by one eye, a robot is controlled only by this node. When a robot enters into an overlapping area covered by two neighbouring eyes, one of them will be elected to be the dominant eye to control the robot through negotiation between the two eyes based on some pre-defined criteria, for example, whether the robot is observed in the central or edge area of the eye coverage.

The rest of the chapter is organised as follows. Section two states the problem and gives a general idea of mosaic eye controlled snake based robot navigation; Section three summarises the generation of the Reference snake (R-snake) with curvature constraints; The

Accompany snake (A-snake) spatial position planning algorithm and time series trajectory based on R-snake are discussed in Section four and five respectively; Section six focuses on the control dedicated protocol that connects all distributed vision sensors; Experiment results and conclusions are given in section seven and section eight respectively.

2. Problem statement and overview of a snake based robot navigation by mosaic eye

In general, a snake is defined as a series of control points connected one by one to represent a collision free path. At initialisation, the start and target (or the end) control points of the snake will be input by an operator manually. Other control points between the start and target points are generated by a global search algorithm, such as Dijkstra's search algorithm (Cormen et al. 2001). The snake shape is dynamically deformed by two kinds of forces exerted on all the control points (Kass, Witkin et al. 1988). Internal forces are forces exerted by other control points within the snake body, such as the attraction force used to reduce the distance between two control points or the bending force used to reduce the sharpness of the snake body. External forces are actions generated from the environment rather than the control points. For example, the repulsive force from obstacles is an external force. The operation for deforming a snake body only requires interactions between adjacent joints (hereafter referred to as control points in this chapter) and reactions to surrounding obstacles in a certain range, e.g. one segment will move according to the locally observed obstacles approaching to it and this movement will affect its neighbouring segments, as a result, the whole snake will deform into a new optimised shape to avoid the obstacles. Therefore, it is suitable to apply the snake algorithm in distributed environment, such as distributed wireless sensor network, to plan a navigation path using only local information exchange between adjacent control points to alert a navigation robot in advance of any obstacles ahead of its navigation path with global effect.

The snake algorithm proposed in this chapter is designed for path planning as well as trajectory generating and robot motion control in an architectural environment installed with distributed vision sensors compliant to the non-holonomic constraints and control saturations. Each distributed vision sensor will maintain one part of the global path dynamically. Local environment information will be exchanged by communications between adjacent vision sensors. By negotiation, one of the vision sensors will plan the local trajectory and control the robot. This is rather different from previous applications where the generation of the snake path relies on centralised processing and global information. The snake will evolve and adapt dynamically to the local environment in an elastic manner, forming a collision free and constraint compliant reference global path, i.e. the R-snake path, for a navigation robot.

While the R-snake provides a reference path for a robot to travel, controlling the robot to follow the path is another difficult task. It involves trajectory generation and motion control subject to non-holonomic constraints and saturated torques, slippage speed etc. A proposed approach to take into account such constraints in this chapter is to develop a distributed control scheme, providing the environment with the ability to perceive information from a large area using distributed sensors to maintain the robot navigation performance up to its maximum driving speed limit. Therefore, an A-snake based on the R-snake is introduced to cope with all these constraints as well as to plan the shortest travel time trajectory. The A-snake is further divided into two phases: 1) A-snake spatial position planning; and 2) A-

snake time series trajectory generation and motion control. Phase one only deals with path geometry features without considering the time factor. The primary goal is to generate an associated path that converges to the R-snake path and at the same time complies with the dynamic and non-holonomic constraints, and control saturation. Phase two selects predictive control (Maciejowski 2001) to optimise the forthcoming tracking. The model based prediction also alleviates effects due to the slow feedback from vision sensors. A rolling window optimisation (Xi & Zhang 2002) is carried out to generate the optimum velocity profile of the mobile robot up to a certain distance from its current position.

3. Planning a R-snake under curvature constraints

Let p_i be a Euclidian coordinate representing the Cartesian configuration (x_i^{cp}, y_i^{cp}) in space R^2, where $i \in Z$ and Z is the set of integers. For a positive integer n, $\{p_0, p_1, ..., p_n\}$ denotes a sequence of configurations in space R^2. A snake is created by connecting adjacent coordinates sequentially. Each coordinate, p_i, represents a control point, which can be moved by exerted internal and external forces from obstacles and adjacent control points. An obstacle q_i is defined as a circle with a radius d_o, centred at (x_i^o, y_i^o). The total number of obstacles in a physical space covered by a vision sensor node is m. Then the objective of the snake algorithm is to adjust the n control points $\{p_0, p_1, ..., p_n\}$ dynamically for keeping the robot within a safe distance away from m obstacles $q_1, ..., q_m$, satisfying the given curvature constraint and maintaining the shortest path length from its start point to the goal point via intermediate control points.

As discussed above, the safe path for a robot is the R-snake path maintained by the distributed mosaic eye. Define the internal and external energy functions in space R^2 as the energy caused by attractive actions from adjacent control points and repulsive actions from obstacles respectively, then the total energy, E^{snake}, of the snake can be expressed as(Quinlan & Khatib 1993),

$$E^{snake} = E^{internal} + E^{external} \tag{1}$$

where the internal energy, $E^{internal}$, is only concerned with the intrinsic actions of the snake, such as its shape and length while the external energy, $E^{external}$, is concerned with the effect from the environment, such as obstacles.

To minimize the energy exerted on a snake, p_i should move along the negative energy gradient direction so that the total energy of the snake decreases. The total force, F^{snake}, exerted on the snake, in terms of $E^{internal}$ and $E^{external}$ can be expressed as:

$$\begin{aligned} F^{snake} &= F^{internal} + F^{external} \\ &= (-\nabla \eta_{\text{int}} E^{internal}) + (-\nabla \eta_{ext} E^{external}) \end{aligned} \tag{2}$$

where, η_{int} and η_{ext} are positive coefficients representing the force strength, ∇ is the gradient operator.

In Eq. (1), different forces serve different purposes. The aim of path planning is to find an obstacle free and the shortest length path that satisfies the curvature constraints. Thus, the elastic contraction energy and the curvature bending energy are considered as internal

energy and the obstacles repulsive energy is considered as external energy. Let f^o be the obstacle force, f^e be the elastic force and $f^{curvature}$ be the curvature constraints force. By adding all internal and external forces, one gets the resultant force f_i^r on a control point i where $f_i^r = \sqrt{f_{i,x}^r + f_{i,y}^r}$ and

$$\begin{aligned} f_{i,x}^r &= f_{i,x}^o + f_{i,x}^e + f_{i,x}^{curvature} \\ f_{i,y}^r &= f_{i,y}^o + f_{i,y}^e + f_{i,y}^{curvature} \end{aligned} \tag{3}$$

Readers can refer to (Cheng, Hu et al. 2008) for more detailed descriptions on obstacle, elastic and curvature forces. A three-state R-snake for guaranteed curvature constraints can also be found in (Cheng, Hu et al. 2008).

4. A-snake spatial position planning

Feedback control of distributed vision sensors is adopted for the dynamic generation of an A-snake spatial position. The A-snake extends from the robot's current position and orientation to approach the R-snake path by taking into account the limited steering torque under the non-holonomic constraints. This will correct the deviation of the robot's position from the reference path. A key issue in this phase is whether the A-snake can converge to the R-snake path without violating the steering torque and driving force limitations.

Fig. 2 denotes a coordinate system with the doted circles representing the R-snake control points and the square box indicating the robot. θ_d is the desired direction which the robot should follow at each iterative step; θ is the robot's current direction; $\vec{i}$ and $\vec{j}$ are the unit tangential vector and normal vector of the robot's local coordinates along its direction of movement respectively; $\vec{i}_0$ is the tangential vector on the nearest reference control point; $\vec{r}$ is the robot's location vector, $\vec{r}_0$ is the vector of the nearest reference control point of the snake; θ_e is the direction error between the desired direction and the robot's current direction; ξ is a positive coefficient. The A-snake will always start from the robot current position.

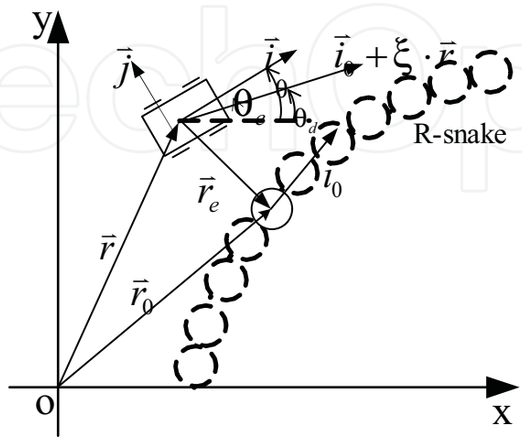


Fig. 2. Coordinates definitions

From (Cheng et al. 2010), a proper desired direction θ_d which can lead the A-snake to converge to the R-snake is,

$$\theta_d = \theta_0 + \arctan\left(\frac{\Delta r_{\bar{j}}}{\Delta r_{\bar{i}}}\right) \tag{4}$$

where,

$$\begin{cases} \Delta r_{\bar{i}} = 1 \\ \Delta r_{\bar{j}} = \xi_1 \cdot r_e \end{cases} \tag{5}$$

ξ_1 is the positive gain along the vector $\vec{r}_e$ and,

$$r_e = sign(\vec{r}_e) \cdot \left\| \vec{r}_0 - \vec{r} \right\|, \tag{6}$$

$$sign(\vec{r}_e) = \begin{cases} -1, & \textit{robot is on the left side of the R - snake} \\ 1, & \textit{robot is on the right side of the R - snake} \end{cases}$$

Eq. (4) gives the desired direction for the A-snake to follow so as to converge to the R-snake. In practice, the robot's moving direction cannot always reach the desired value due to the non-holonomic constraints (maximum steering torque) and dynamic constraints (maximum driving force and frictions). The maximum robot velocity $v_{\max}$ should be obtained as,

$$v_{\max} = \min\left(v_{\tau\max}, v_{f\max}\right) \tag{7}$$

where,

$$\begin{aligned} v_{\tau\max}^2 &= \frac{\tau_{\max}}{J\left|\partial k / \partial s\right|} \\ v_{f\max}^2 &= \frac{(\mu N)_{\max}}{M\left|k\right|} = \frac{g\mu_{\max}}{\left|k\right|} \end{aligned} \tag{8}$$

M and J are the mass and the inertia of the robot respectively; g is the Gravity factor; μ is the friction coefficient; N is the normal force with the ground; k is the trajectory curvature; s is the arc length along the snake and the saturated torque of a robot $\tau_{\max}$.
Thus, the control points of A-snake can grow iteratively according to Eq. (9) ,

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \int \begin{pmatrix} \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial s} \end{pmatrix} ds + \begin{pmatrix} x \\ y \end{pmatrix} \tag{9}$$

where, $\begin{pmatrix} x \\ y \end{pmatrix}$ and $\begin{pmatrix} x' \\ y' \end{pmatrix}$ are the current and newly calculated control point positions respectively, and ,

$$\begin{pmatrix} \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial s} \end{pmatrix} = \begin{pmatrix} \cos(\int \frac{\partial \theta}{\partial s} ds) \\ \sin(\int \frac{\partial \theta}{\partial s} ds) \end{pmatrix} \tag{10}$$

5. A-snake time series trajectory generation and motion control

Although the A-snake spatial position provides a reference path for a robot to travel, due to the limited field of view of the onboard sensors, the tracking speed has to compromise with safety. In fact, the far sight provided by a large scale distributed sensor network is developing the foundation and potential for the optimal control of vehicles. A vehicle is envisaged to be able to respond to changes on its way ahead and be driven with optimal time or energy in a dynamic environment. Distributed control to achieve high performance tracking up to the driving limit becomes a promising technique.

In (Cheng, Jiang et al. 2010), a predictive control scheme combining trajectory planning and dynamic control is developed to achieve optimal time tracking, taking into account the future path that needs to be tracked, subject to non-holonomic constraints, robot kinematic and dynamic constraints, the maximum velocity without slippage, driving force and steering torque saturation.

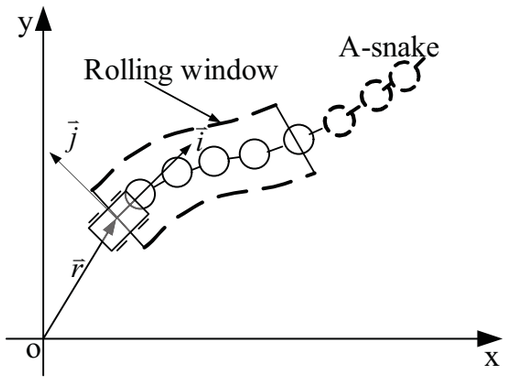


Fig. 3. Robot, A-snake and rolling window

Define a rolling window with length l along a snake as shown in Fig. 3, which could be distributed in several wireless sensors and evolved by individual vision sensors asynchronously. The optimal control for a robot can be achieved by optimizing,

$$\min_{F^d,\tau}(\Gamma) = \min_{F^d,\tau}(\int_0^l 1/\ v(s)ds\big|_{s\in A}) \tag{11}$$

with boundary conditions: $v(0) = v_{r0}, v(l) = 0$; subject to

$$\begin{cases} \left|f^{friction}\right| < (\mu N)_{\max}, & for\ non-slippage \\ \left|F^d\right| < F^d_{\max}, & for\ limited\ driving\ force \\ \left|\tau\right| < \tau_{\max}, & for\ limited\ steering\ torque \end{cases} \tag{12}$$

where A is the A-snake; $v(s)$ is the velocity profile of the robot; v_{r0} is the sampled velocity of the robot at any instant; f is the friction of tires with coefficient μ and the normal force N; F^d and τ are the driving force and steering torque of the robot with upper bounds of $F^d_{\max}$ and $\tau_{\max}$ respectively. The objective equation Eq.(11) for predictive control is to minimise the robot's travelling time Γ which is also called a servo period along the snake from its current location until the end of the l-window. However the robot should be able to stop at the end such that $v(l)=0$ in order to respond to the worst possible circumstances which are not covered by the current rolling window.

In order to optimise Eq.(11), the area under the velocity profile $v(s)$ will be maximised so that the area of its reciprocal can be minimised according to the maximum uniform velocity obtained in Eq.(7), which is the maximum allowable velocity for the robot to travel along the A-snake without any effort for acceleration or deceleration, this section will obtain the optimal force and torque approaching this maximum allowable velocity as much as possible, considering the dynamic constraints and satisfying the boundary conditions.

In order to satisfy the two boundary conditions in Eq. (11), one needs to design optimal control inputs such that the robot can safely travel a path and is able to stop at the end of the rolling window, considering the dynamic constraints. A sharp bend on the way requires the robot to decelerate in advance due to the limited driving force and steering torque. The optimisation of Eq. (11) can be solved by maximising the area of $v(s)$, which can be achieved by squeezing the velocity profile from the two ends towards the middle using the maximum force and torque.

The maximum allowable positive acceleration can be obtained as

$$\begin{aligned} a_{F\max+} &= F^d_{\max} / M \\ a_{\tau\max+} &= \frac{\tau_{\max}}{J|k|} - \frac{v^2}{k}\frac{\partial k}{\partial s} \\ a_{\max+} &= \min(a_{F\max+}, a_{\tau\max+}) \end{aligned} \tag{13}$$

If $v \le v_{\max}$, then $a_{\max+} \ge 0$, this implies that a positive acceleration exists. Therefore, the acceleration process has to be bounded by $v_{\max}$.

Similarly, the maximum allowed negative acceleration can be obtained as

$$\begin{aligned} a_{F\max-} &= -F^d_{\max} / M \\ a_{\tau\max-} &= -\frac{\tau_{\max}}{J|k|} - \frac{v^2}{k}\frac{\partial k}{\partial s} \\ a_{\max-} &= \max(a_{F\max-}, a_{\tau\max-}) \end{aligned} \tag{14}$$

A negative acceleration exists if the velocity is bounded by $v_{\max}$.

The squeezing process is approximated by segments of uniform acceleration/ deceleration movements from two boundary velocities with an incremental step δ:

For acceleration at s_+, forward planning is carried out

$$v^2_+(s_+) = v^2_+(s_+ - \delta) + 2\alpha_{\max+}\delta \tag{15}$$

For deceleration at s_-, backward planning is carried out

$$v_-^2(s_-) = v_-^2(s_- + \delta) + 2\alpha_{\max -}\delta \tag{16}$$

During the squeezing process, it is needed to ensure that the values of $v_+^2(s_+)$ and $v_-^2(s_-)$ in Eq. (15) and Eq. (16) do not exceed $v_{\max}$ for the segment in between s_+ and s_-. If this happens at any point $s_\#$, the velocity profile can be squeezed for the segment from $v_+(s_+)$ to $v_-(s_\#) = v_{\max}(s_\#)$. The process continues until the acceleration segment and the deceleration segment encounter each other. The velocity profile will be obtained by repeating this squeezing process for the remaining segments. Working in this way, the area of $v(s)$ is maximised and therefore the travelling time is minimised. The generated velocity profile informs the robot when to accelerate or decelerate in advance in order to safely track the dynamic snake in a predictive l-window. The algorithm is summarised as below and is shown in Fig. 4:

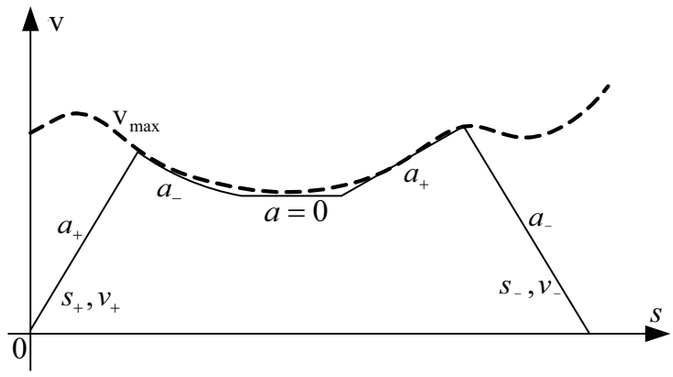


Fig. 4. Rolling window optimisation for trajectory generation (assume $v_{r0} = 0$)

1. According to the snake, obtain the maximum allowable velocities $v_{\max}$ from Eq.(7) in rolling window l;
2. Initialise the squeezing process with the following boundary conditions: initial state $s_+ = 0, v_+(0) = v_{r0}$ and terminate state $s_- = l, v_-(l) = 0$;
3. Plan/ increase v_+ and v_- in parallel: if $v_+ \le v_-$, increase v_+ by Eq. (15) and $s_+ = s_+ + \delta$; If $v_+ > v_-$, increase v_- by Eq. (16) and $s_- = s_- - \delta$;
4. If $s_+ = s_-$ and there is any unplanned segment, set s_+ and s_- to be the unplanned segment and go to 3); if the maximum allowable velocity is found at $s_\#$ between $s_+ \sim s_-$, create two new segments, $s_+ \sim s_\#$ and $s_\# \sim s_-$ then go to 3), otherwise go to 5);
5. Calculate the driving force and steering torque for $s = 0$ as control signals for time t:

$$\begin{aligned} F^d(t) &= Ma_{\max}(0) \\ \tau(t) &= Ja_{\max}(0)k(0) + Jv^2(0)\frac{\partial k}{\partial s}\Big|_{s=0} \end{aligned} \tag{17}$$

where $a_{\max} = a_{\max +}$ or $a_{\max -}$ obtained in step 3);

6. Send $F^d(t)$ and $\tau(t)$ to the robot for control and shift the rolling window one step forward;
7. For every servo period Γ, $F(t+n\Gamma)$ and $\tau(t+n\Gamma)$ will be continuously generated from the obtained velocity profile to control the vehicle until a new profile is received from a vision sensor;
8. For every image sampling period, the mosaic eye updates the A-snake and then repeats (1).

6. Communication empowered distributed mobile robot navigation

The purpose of this section is to design a control dedicated communication protocol connecting all vision sensors and mobile robot thus enabling the mobile robot navigation functionality in the intelligent environment. Meanwhile the protocol will establish a reliable and precise data exchange among the intelligent environment in order to support the path planning and motion control.

6.1 Overview

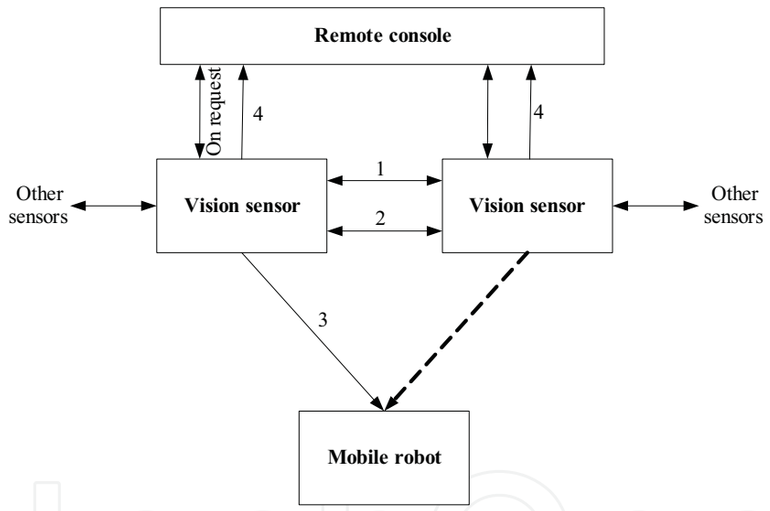


Fig. 5. Communication protocols overview

Fig. 5 is an outline of all the communication parties and the links between them. As the key intelligent component, vision sensors are involved in all communications, either as a sender or as a receiver. Except the '*on request*' commands originated by an operator in the remote console and the subsequent responses to these requests, commands are invoked periodically in the following sequence:

1. **Exchanging of control points and/or obstacles**: Control points coordinates are used to calculate internal forces to deform a snake. Snake segments in two neighbouring vision sensors can interact with each other by adjusting their positions according to the forces generated by the received control points coordinates from other vision sensors. Obstacles observed by one vision sensor will be sent to its neighbouring sensors if localisation fusion is required. After snake deformation, the vision sensor will send the planned control points to its neighbouring sensors;

2. **Control token negotiation**: At a specific time, one robot should be controlled by only one vision sensor. All vision sensors which have the robot in view will compete for the token. The vision sensor with a control token will become the dominant vision sensor and broadcast its ownership of the token periodically or initiate a token handover procedure if required;
3. **Mobile robot control**: The dominant vision sensor sends control to the robot; the one without a control token will skip this step;
4. **Monitoring purpose reporting**: If a vision sensor is marked by an operator to send monitoring related information, such as control points, it will send out the corresponding information to the remote console.

6.2 Protocol stack structure

The proposed control protocol is built on top of the IEEE 802.15.4 protocol which has the following data structure (Zhang 2008),

```
typedef struct __TOS_Msg
{
        __u8 length;     // data length of payload
        __u8 fcfhi;              // Frame control field higher byte
        __u8 fcflo;              // Frame control field lower byte
        __u8 dsn;                // sequence number
        __u16 destpan; // destination PAN
        __u16 addr;      // destination Address
        __u8 type;               // type id for Active Message Model handler
        __u8 group;      // group id
        __s8 data[TOSH_DATA_LENGTH];  // payload
    __u8 strength;//signal strength
    __u8 lqi;
    __u8 crc;
    __u8 ack;
    __u16 time;
   } TOS_Msg;
```

As seen in the TOS_Msg structure, 16 bytes are used as headers, the maximum payload length, TOSH_DATA_LENGTH, should be 112 bytes. The control protocol packets are encapsulated and carried in the payload. The protocol stacks at different interfaces are discussed in the following subsections.

6.2.1 Protocol stack between vision sensors

As shown in Fig. 6, the control protocol layer is built on top of the physical lay and MAC layer of the 802.15.4 protocol stack to enable vision sensors to communicate with each other. The information processing and robot navigation control algorithms are resided within the control protocol layer.

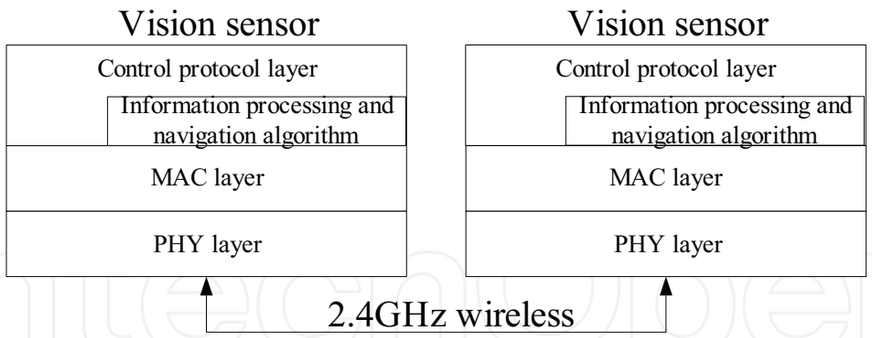


Fig. 6. Protocol stack between vision sensors

6.2.2 Protocol stack between vision sensor and mobile robot

Similar to the protocol stack between visions, the control protocol stack between vision sensor and mobile robot is shown in Fig. 7.

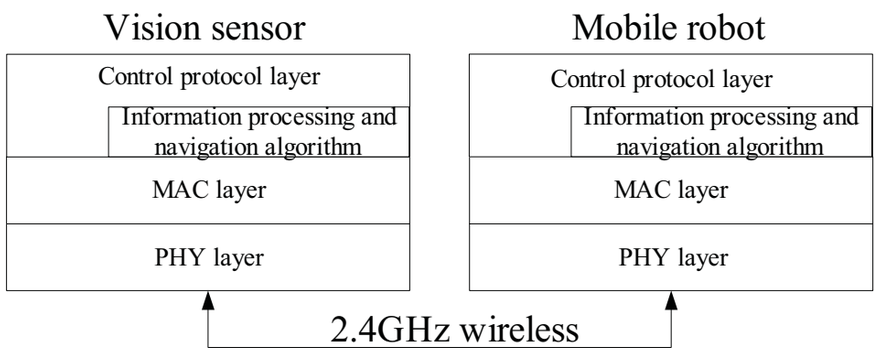


Fig. 7. Protocol stack between vision sensor and mobile robot

6.2.3 Protocol stack between vision sensor and remote console

To enable the communication between a normal PC and the vision sensor, a wireless adaptor is used to make conversion between 2.4GHz wireless signal and USB wire connection. The GUI application in the remote console PC will act as a TCP server which listens to the connection request from the wireless adaptor. The protocol stack is shown in Fig. 8.

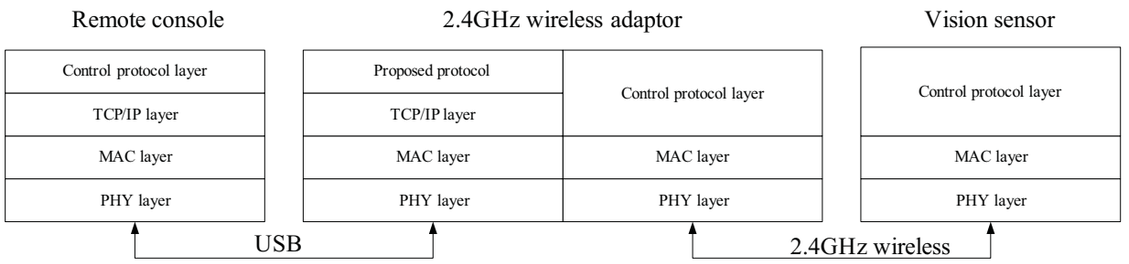


Fig. 8. Protocol stack between vision sensor and remote console

6.3 Generic packet structure

As mentioned above, the control protocol will be based on the TOS_Msg data structure. All packets are carried within the *data* area of the TOS_Msg structure. The generic packet format is defined as below Table 1.

Byte 0 0 1 2 3 4 5 6 7	Byte 1 0 1 2 3 4 5 6 7	Byte 2 0 1 2 3 4 5 6 7	Byte 3 0 1 2 3 4 5 6 7
CHK	CMD	SrcAddr	
SN		TotalNum	
User payload……			

Table 1. Generic packet structure

The fields are,

- CHK: check sum which is the remainder of the addition of all fields except the CHK itself being divided by 256;
- CMD: type of commands which identifies different control protocol payloads;
- SrcAddr: Sender short address from 1 to 65535 (0 is broadcast address);
- SN: Packet sequence number;
- TotalNum: Total number of packets to be transmitted;
- User payload: the length varies from 0 to 104 bytes depending on the CMD value; the structures of different payloads will be discussed in the next subsection.

6.4 Detailed design of the proposed control protocol

There are basically 5 commands designed to meet the data exchange and control requirements. Their descriptions are listed in Table 2.

CMD	Description	Message direction
1	Control points	Vision sensor ←→ vision sensor
2	Obstacles	Vision sensor ←→ vision sensor
3	Token negotiation	Vision sensor ←→ vision sensor
4	Mobile robot control commands	Vision sensor ←→ mobile robot
5	Monitoring purpose	Vision sensor ←→ remote console

Table 2. Command lists

The following subsections will discuss the detailed usage and packet structure of each command. It is organized according to the command sequence.

6.4.1 Control points

This is a vision sensor to vision sensor command. The purpose of this command is to transmit the planned control points from one vision sensor to another. To reduce the communication burden and save frequency resource, only the preceding vision sensors send border control points to the succeeding ones, as shown in Fig. 9.

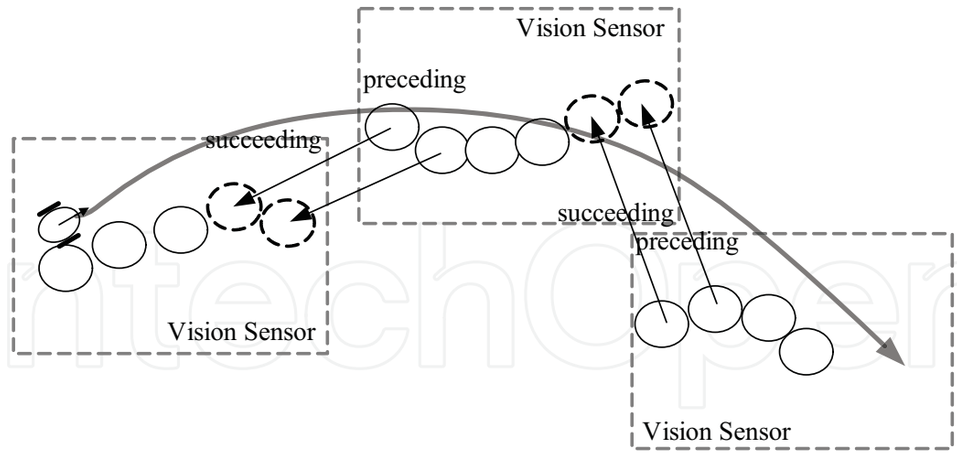


Fig. 9. Sending border control points from preceding vision sensor to succeeding ones

The signal flow is shown in Fig. 10. Border control point coordinates are transmitted periodically by all the vision sensors to their succeeding vision sensors if they exist. Destination address is specified in the TOS_Msg header.

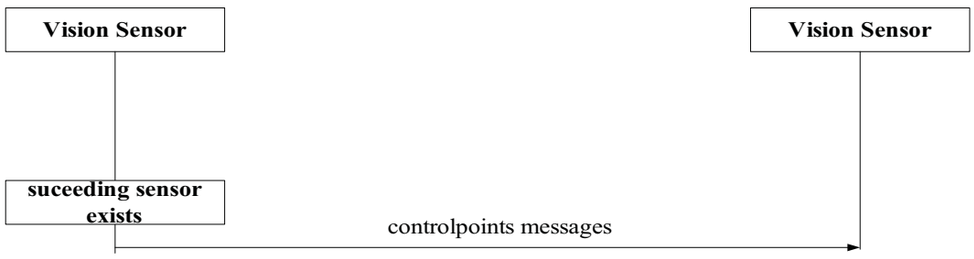


Fig. 10. Exchange border control points signal flow

The corresponding packet format is shown in Table 3,

Byte 0 0 1 2 3 4 5 6 7	Byte 1 0 1 2 3 4 5 6 7	Byte 2 0 1 2 3 4 5 6 7	Byte 3 0 1 2 3 4 5 6 7
CHK	CMD	SrcAddr	
SN		TotalNum	
NCP			
Series of control point coordinates ……			

Table 3. Control point packet format

where,

- CHK, SrcAddr, SN and TotalNum are referred to section 6.3
- CMD = 1
- NCP: Total number of control points to be sent, maximum 25 (103/ 4) control points can be sent within one packet
- Control point coordinates (x, y) are followed by format below,

Byte 0 0 1 2 3 4 5 6 7	Byte 1 0 1 2 3 4 5 6 7	Byte 2 0 1 2 3 4 5 6 7	Byte 3 0 1 2 3 4 5 6 7
x		*y*	

6.4.2 Obstacles

This is a vision sensor to vision sensor command. It is created to provide information for multiple geometry obstacle localisation. If obstacles are observed by one vision sensor, and this vision sensor has overlapping areas with the dominant one, it will transmit the observed obstacles to the dominant sensor. This function can be disabled to reduce the communication burden in the program.
The data format is shown in Table 4.

Byte 0 0 1 2 3 4 5 6 7	Byte 1 0 1 2 3 4 5 6 7	Byte 2 0 1 2 3 4 5 6 7	Byte 3 0 1 2 3 4 5 6 7
CHK	CMD	SrcAddr	
SN		TotalNum	
NOB			
Series of obstacle coordinates ……			

Table 4. Obstacle packet format

where,

- CHK, SrcAddr, SN and TotalNum are referred to section 6.3
- CMD = 2
- NOB: Total number of obstacles to be sent
- The obstacles coordinates are as the same format as control points in section 6.4.1

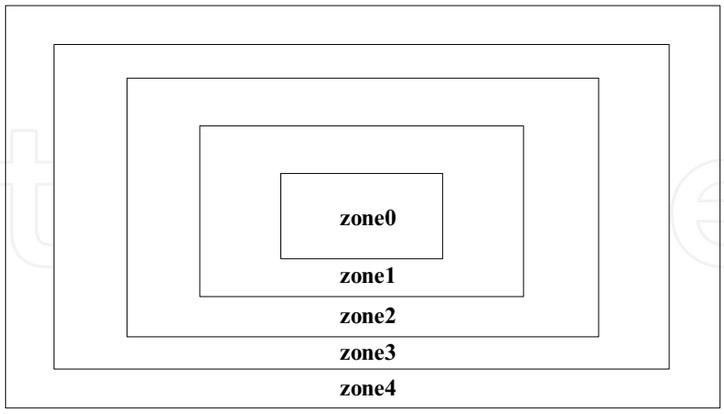


Fig. 11. Division of the observation area into zones in one vision sensor

6.4.3 Token negotiation

At a specific time, only the dominant vision sensor can send control command to the mobile robot. In the proposed distributed environment, there is no control centre to assign the control token among vision sensors. Therefore all vision sensors have to compete for the

control token. By default, vision sensors with the mobile robot in view will check whether other visions broadcast the token ownership messages. If there is no broadcast messages received within a certain period of time, it will try to compete for the control token based on two criteria: 1) the quality of the mobile robot being observed by the vision sensors and 2) a random number generated by taking into account the vision sensor short address as the seed. The quality of the mobile robot being observed is identified by different zones shown in Fig. 11. Zone0 is in the inner area which denotes the best view and zone4 is in the outer area which represents the worst view. Different zones are not overlapped and divided evenly based on the length and width of the view area.

The control token negotiation procedures are interpreted as following four cases.

Case 1: One vision sensor sends request to compete for the token and there is no other request found at the same time. A timer is set up once the command is broadcasted. If there is no other token request messages received after timeout, the vision sensor takes the token and broadcast its ownership of the token immediately. Fig. 12 shows the signal flow.

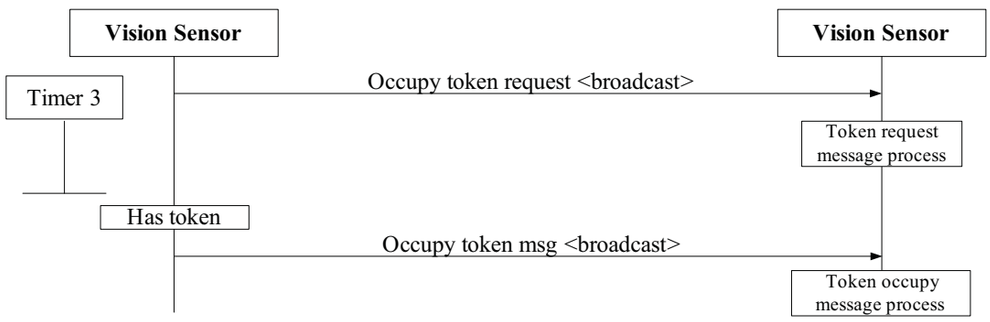


Fig. 12. Control token init signal flow, case 1

Case 2: If a control token request message is received before timeout, the vision sensors will compare its observation quality with the one carried in the broadcast message. The one with the less zone number will have the token. It might be a possibility that the zone numbers are the same, then the values of their short addresses are used to determine the token ownership, i.e. smaller value of the address will be the winner. Fig. 13 depicts the signal flow.

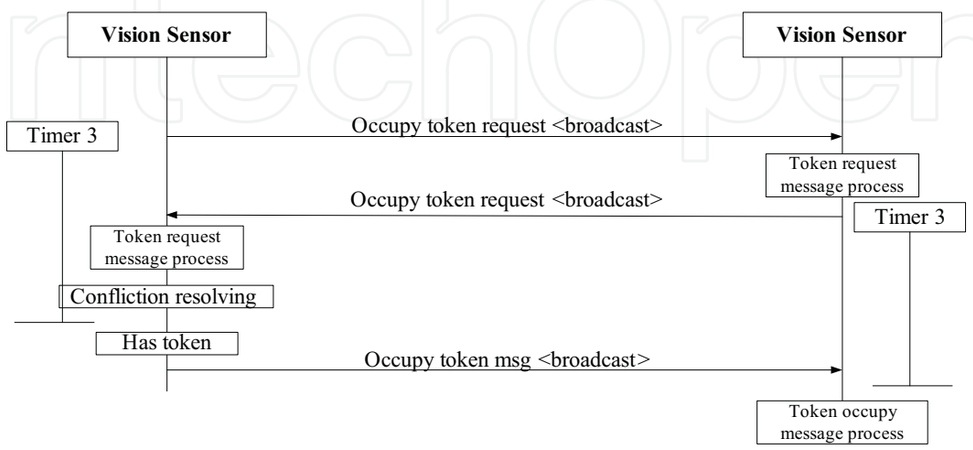


Fig. 13. Control token init signal flow, case 2

Case 3: Once a vision sensor has the control token, it will broadcast its ownership periodically. Upon receipt of this message, other vision sensors will set up a timer which should be greater than the time for a complete processing loop (image processing, path planning and trajectory generation). During the lifetime of this timer, it assumes that the ownership is still occupied by others and will not send request message during this time. If the dominant vision sensor receives a token request message, it will reply with an token already being occupied message immediately to stop other vision sensor from competing for the token. Fig. 14 shows the signal flow.

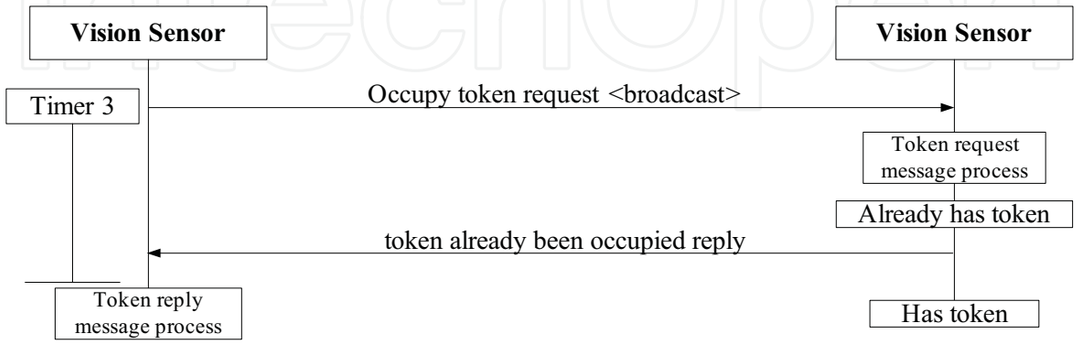


Fig. 14. Control token init signal flow, case 3

Case 4: When the mobile robot moves from an inner area to an outer area in the vision, the dominant vision sensor will try to initiate a procedure to handover the token to other vision sensors. First it broadcasts a token handover request with its view zone value and setup a timer (Timer 1). Upon receipt of the handover message, other vision sensors will check whether they have a better view on the robot. Vision sensors with better views will send token handover reply messages back to the dominant vision sensor and setup a timer (Timer 2). If the dominant vision sensor receives the response messages before the Timer 1 expires, it will choose the vision sensor as the target and send token handover confirmation message to that target vision sensor to hand over its ownership. If there is more than one vision sensors reply the handover request message, the dominant one will compare their view zone values and preferably send the handover confirmation message to the vision sensor with less zone value. If they have the same view quality, vision sensor short address will be used to decide the right one. If token handover confirmation message is received, the target vision sensor will have the token, as shown in Fig. 15. However if no handover confirmation messages received before the Timer 2 expires, i.e. the handover confirmation message does not reach the recipient, a token init procedure will be invoked as no other sensors apart from the dominant vision sensor has the token to broadcast the occupy token message which is shown in Fig. 16.

The packet format is listed in Table 5.

Byte 0 0 1 2 3 4 5 6 7	Byte 1 0 1 2 3 4 5 6 7	Byte 2 0 1 2 3 4 5 6 7	Byte 3 0 1 2 3 4 5 6 7
CHK	CMD	SrcAddr	
SN		TotalNum	
type	zone		

Table 5. Token packet format

where,

- CMD = 3
- CHK, SrcAddr, SN and TotalNum are referred to section 6.3
- type: Token message types.

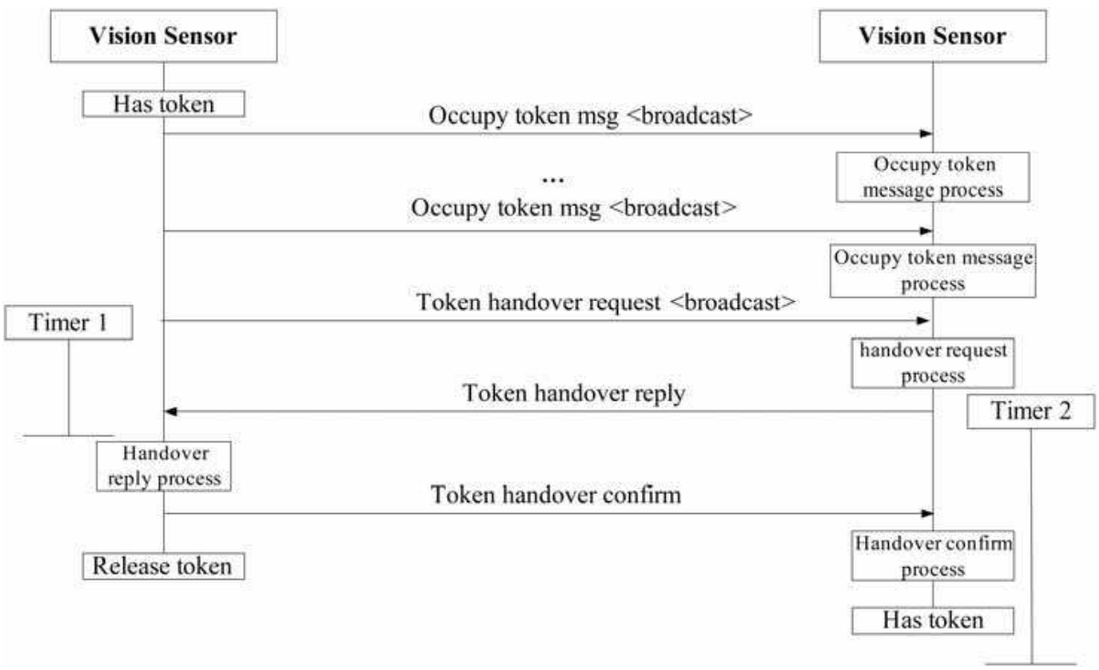


Fig. 15. Control token handover signal flow - successful

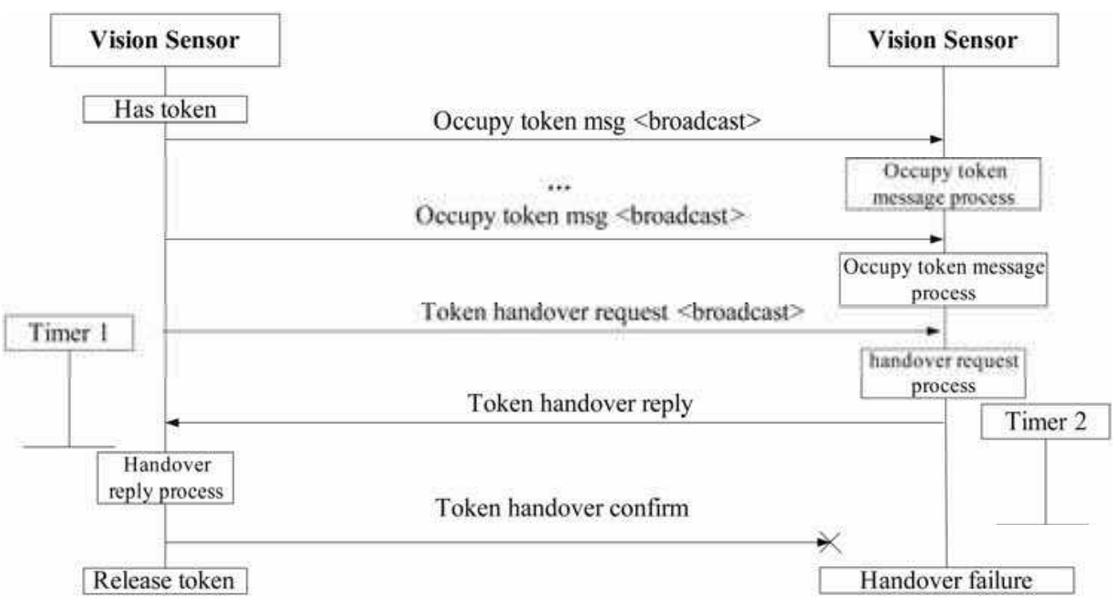


Fig. 16. Control token handover signal flow - failure

The descriptions and possible values for type is listed in Table 6,

type value	Description
0	Init token request
1	Occupy token msg
2	token already occupied reply
3	Token handover request
4	Token handover reply
5	Token handover confirmation

Table 6. Token messages

- zone: view zones. It is used to indicate the quality of mobile robot being observed in one vision sensor. The zone0, zone1, zone2, zone3 and zone4 are represented by 0, 1, 2, 3 and 4 respectively.

6.4.4 Mobile robot control

This is a vision sensor to mobile robot command. After planning, the dominant vision sensor will send a series of commands to the robot with time tags. The signal flow is shown in Fig. 17.

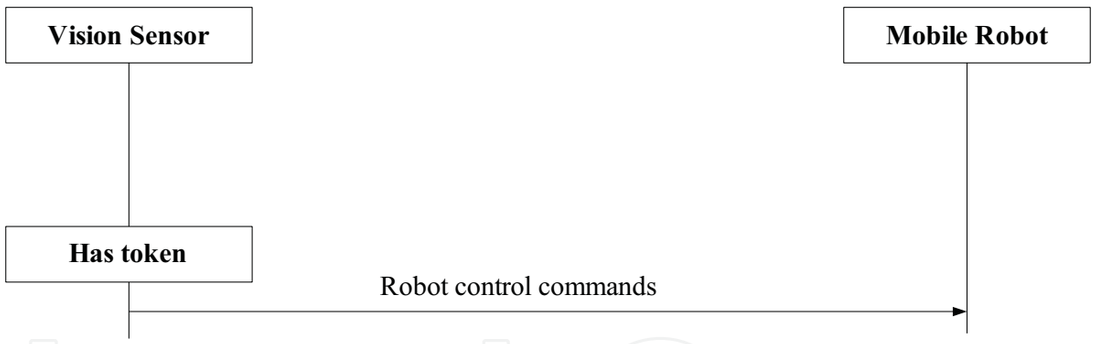


Fig. 17. Robot control signal flow

The packet format is shown in Table 7,

Byte 0 0 1 2 3 4 5 6 7	Byte 1 0 1 2 3 4 5 6 7	Byte 2 0 1 2 3 4 5 6 7	Byte 3 0 1 2 3 4 5 6 7
CHK	CMD	SrcAddr	
SN		TotalNum	
Num of steps	Control parameters ……		

Table 7. Robot control commands packet format

where,

- CMD = 4

- CHK, SrcAddr, SN and TotalNum are referred to section 6.3
- Num of steps: Number of control commands in one packet. It could be one set of command or multiple set of commands for the mobile robot to execute
- Control parameters: one set of control parameter includes five values as below,

Byte 0 0 1 2 3 4 5 6 7	Byte 1 0 1 2 3 4 5 6 7	Byte 2 0 1 2 3 4 5 6 7	Byte 3 0 1 2 3 4 5 6 7
timet	Vvalue	Vsign	Dvalue
Dsign			

Timet is an offset value from the previous one with the unit millisecond. The velocity *Vvalue* is the absolute value of the speed with the unit ms^{-1}. The *Dvalue* is the angle from the current direction. The *Timet* and *Vvalue* are multiplied by 100 before they are put in the packet to convert float numbers into integers. The value ranges are listed in Table 8,

Field	**Value**
Dsign	0: left or centre, 2: right
Dvalue	0~45 degree
Vsign	0: forward or stop, 2: backward
Vvalue	0~255 cm/ s

Table 8. Robot control parameter values

6.4.5 Remote console

This is a vision sensor to remote console PC command. The remote console is responsible for system parameters setting, status monitoring, vision sensor node controlling and etc.. The communication protocol between vision sensors and console is designed to provide the foundations of these functions. After configuration of all the parameters, the system should be able to run without the remote console.

As a transparent wireless adaptor for the remote console, the wireless peripheral will always try to initiate and maintain a TCP connection with the remote console PC to establish a data exchange tunnel when it starts.

6.4.5.1 Unreliable signal flow

On the one hand, the operator can initiate requests from the remote console PC to vision sensors, e.g. restart the sensor application, set the flags in the vision sensor to send real time image and/ or control points information, instruct vision sensor to sample background frame and etc.. The wireless module attached with the remote console will be responsible for unpacking IP and sending them wirelessly to vision sensors; On the other hand, vision sensors will periodically send control points, real time images, path information, robot location etc. to the remote console according to the flags set by the operator. The loss of messages is allowed. It is illustrated as Fig. 18.

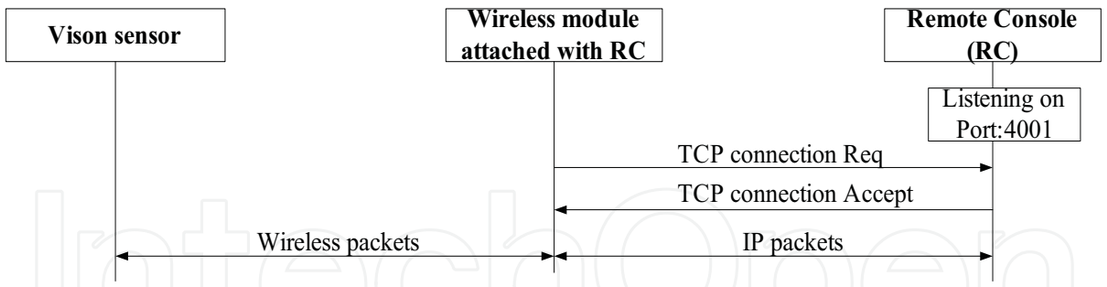


Fig. 18. Unreliable signal flow between remote console and vision sensor

6.4.5.2 Reliable signals

Reliable transmission is also provided in case packet loss is not tolerant, e.g. downloading vision sensor system drivers, updating user programs, transferring setting profile files and etc.. All data packets are required to be acknowledged by the recipient. Re-transmission will be invoked if no confirmation messages received within a given time. The signal flow is illustrated in Fig. 19. The wireless module will buffer the packets and make sure all the packets sent to the vision sensor are acknowledged to ensure a reliable transmission.

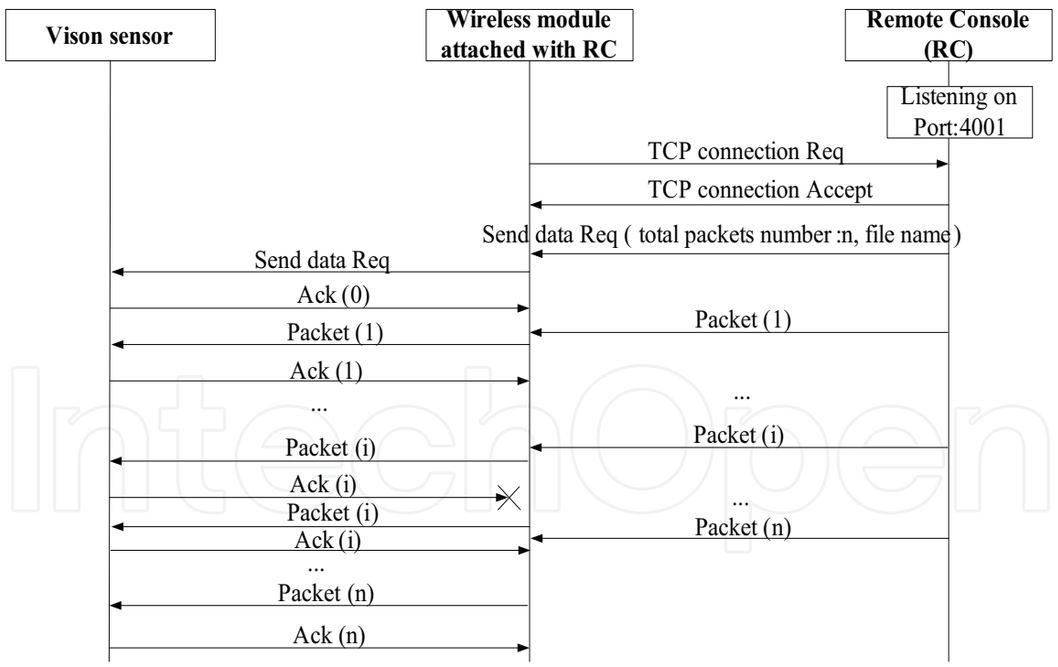


Fig. 19. Signal flow between remote console and vision sensor

7. Simulation and experiment results

Trajectory tracking of a car like robot using the mosaic eye is experimented. Four eyes are mounted on a room ceiling forming a closed continuously running room such that each eye

will have a neighbouring eye at each side, one on the left and another on the right. An independent remote monitor terminal is setup to capture the mosaic eye working status on demand and to maintain mosaic eye when needed. The main processor of the car like robot is a Motorola MC9S12DT128B CPU which is used to execute the received commands from mosaic eye. The mosaic eye, the remote monitor terminal and the robot are all 802.15.4 communication enabled. The car like robot is marked by a rectangle with red and blue blobs on top of it which is used to locate the robot's position and to distinguish the robot from its obstacles, as shown in Fig. 20.

Fig. 20. Car like robot marked by a rectangle with red and blue blobs on it

The predictive control has a rolling window with $l = 20$ (control points). The maximum travelling speed is 0.8 *m/s*, the maximum driving force is $F_{\max}^{d} = 4.4(N)$ with a 0.56(kg) robot mass and the friction factor $\mu_{\max} = 0.6$ and $\tau_{\max} = 2.0(N \cdot m)$.

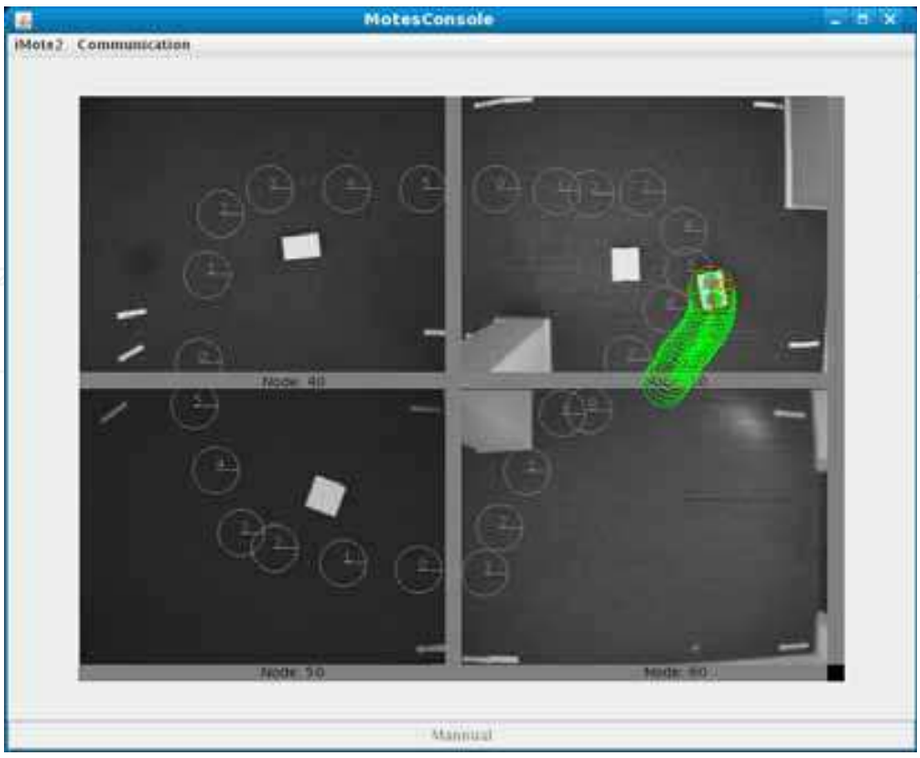


Fig. 21. Robot moving from eye-30 to obstacles free eye-60

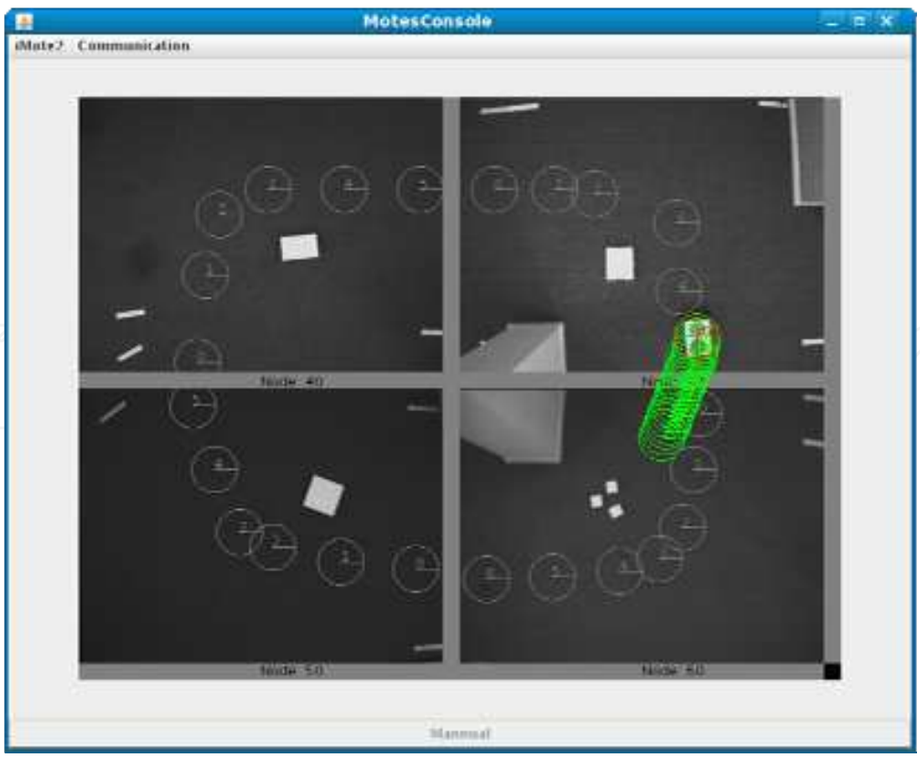


Fig. 22. Obstacles appear in eye-60

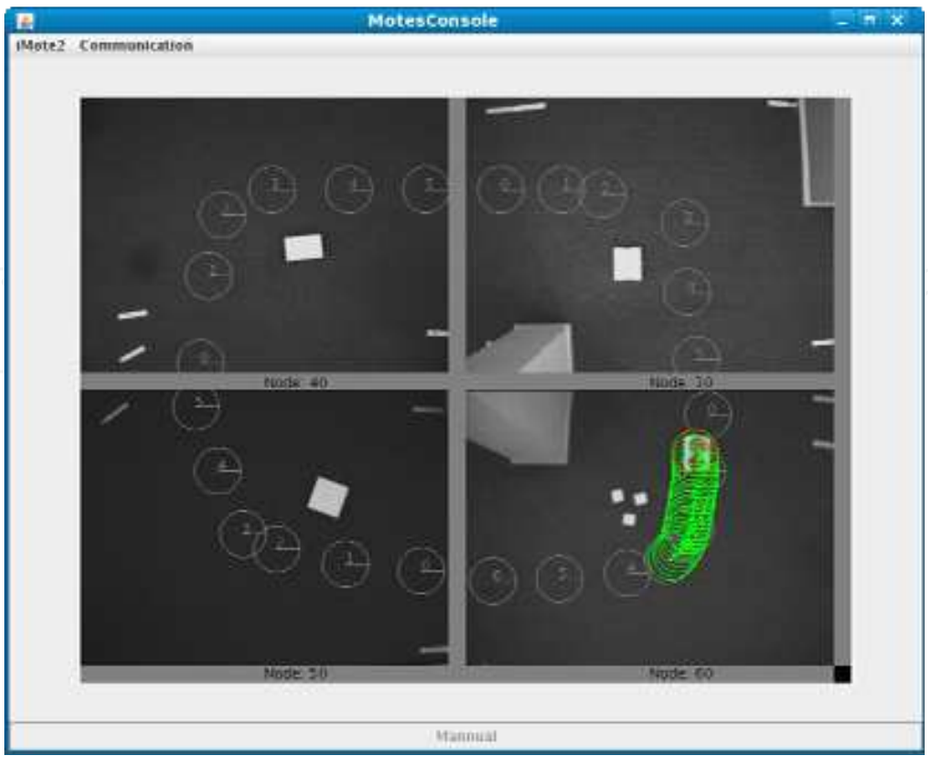


Fig. 23. Robot passing obstacle area in eye-60

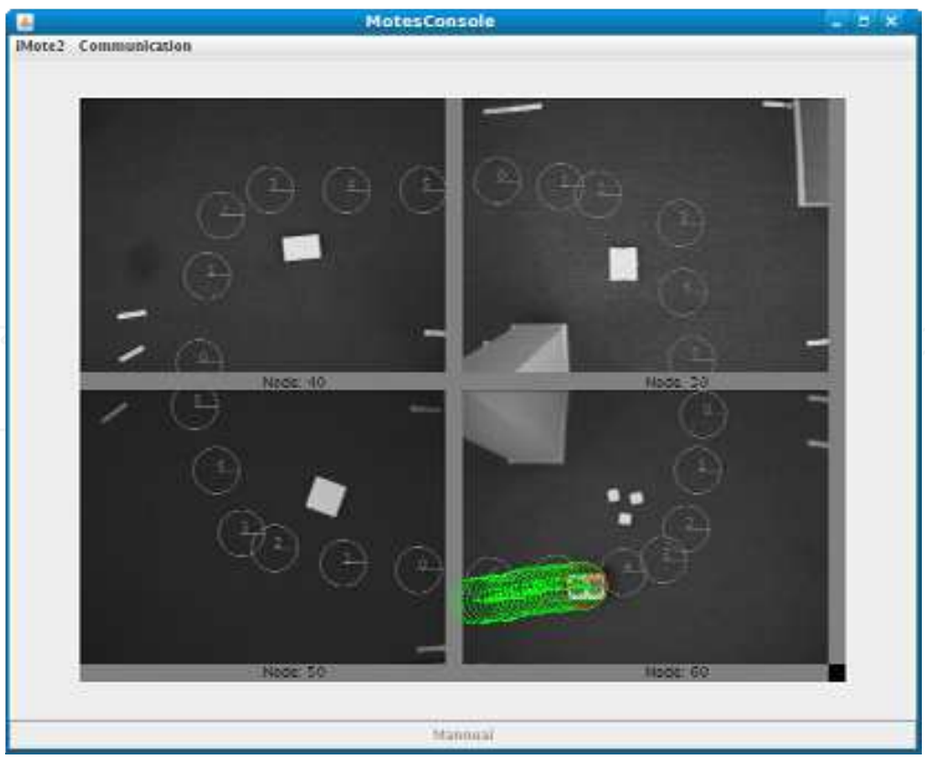


Fig. 24. Robot passed obstacles area in eye-60 and move to eye-50

Fig. 21, Fig. 22, Fig. 23 and Fig. 24 show the real time experiments of robot control by the mosaic eye. Each figure displays four views from each of the four eyes. Let eye-30, eye-40, eye-50 and eye-60 be the names of the mosaic eye starting from top-right one and counting anti-clockwise. In Fig. 21, the robot is controlled by eye-30 heading to the control area of eye-60. The sparse white circles with numbers in the centre represent the desired path that the robot should follow. The white rectangle blobs represent dynamic obstacles. As one can see in Fig. 21, the dynamic obstacles are within the views of eye-30, eye-40 and eye-50 but out of sight of eye-60. In Fig. 22, an obstacle appears within the sight of eye-60. At this point, the robot is under the control of eye-30 and eye-30 does not know the existence of the new obstacle. With the information sent from eye-60 notifying eye-30 of the obstacle, the predictive path is updated to avoid the obstacle. In Fig. , the robot control is handed over from eye-30 to eye-60. The figure shows that with the predictive path updated by eye-30 and with the control of eye-60, the robot has successfully avoided the obstacle and continued to move along the updated predictive path.

8. Conclusion

As an attempt to steer away from developing an autonomous robot with complex centralised intelligence, this chapter proposes a scheme offering a complete solution for integrating communication with path planning, trajectory generation and motion control of mobile robots in an intelligent environment infrastructure where intelligence are distributed in the environment through collaborative vision sensors mounted in a physical architecture, forming a wireless vision sensor network, to enable the navigation of unintelligent robots within that physical architecture through distributed intelligence. A bio-mimetic snake algorithm is proposed to coordinate the distributed vision sensors for the generation of a collision free R-snake path during the path planning process. Segments of a path distributed in individual sensors from a start position to a goal position are described as an elastic band emulating a snake. By following the R-snake path, an A-snake method that complies with the robot's nonholonomic constraints for trajectory generation and motion control is introduced to generate real time robot motion commands to navigate the robot from its current position to the target position. A rolling window optimisation mechanism subject to control input saturation constraints is carried out for time-optimal control along the A-snake.

The scheme has been verified by the development of a complete test bed with vision sensors mounted on a building ceiling. Results obtained from the experiments have demonstrated the efficiency of the distributed intelligent environment infrastructure for robot navigation.

9. References

Arkin (2000). Behavior-based Robotics. Cambridge, MIT Press.

Cameron (1998). Dealing with Geometric Complexity in Motion Planning. New York, Wiley.

Cheng, Hu, et al. (2008). A distributed snake algorithm for mobile robots path planning with curvature constraints. *IEEE Int. Conf. on SMC*, Singapore.

Cheng, Jiang, et al. (2010). A-Snake: Integration of Path Planning with Control for Mobile Robots with Dynamic Constraints. ICCAE.

Cormen, Leiserson, et al. (2001). Introduction to Algorithms, MIT Press and McGraw-Hill.

Kass, Witkin, et al. (1988). Snake: Active contour models. *International Journal of Computer Vision* 1(4): 321-331.

Land and Nilsson (2002). Animal Eyes, Oxford University Press.

Li and Rus (2005). Navigation protocols in sensor networks. A*CM Trans. on Sensor Networks* 1(1): 3-35.

Maciejowski (2001). Predictive control with constraints.

Mclean (1996). Dealing with geometric complexity in motion planning. *Int. Conf., IEEE in Robotics and Automation.*

Mclean and Cameron (1993). Snake-based path planning for redundant manipulators. *Robotics and Automation, IEEE International Conference on*, Atlanta, USA.

Murphy (2000). Introduction to AI robotics. Cambridge, MIT Press.

Quinlan (1994). Real-time modification of collision-free paths. Department of Computer Science, Stanford. PhD.

Quinlan and Khatib (1993). Elastic bands: connecting path planning and control. I*EEE Int. Conf. Robotics and Automation*, Atlanta, USA.

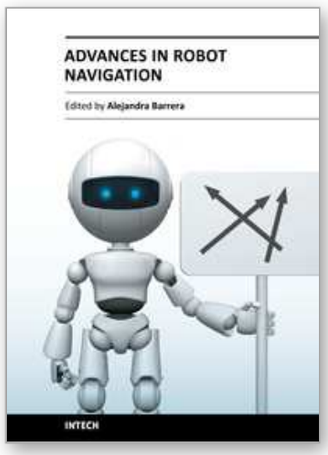
Sinopoli, Sharp, et al. (2003). Distributed control applications within sensor networks. *Proceedings of IEEE.*

Snoonian (2003). Smart buildings. *IEEE Spectrum* 40(8).

Website (2006). http:/ / www.inf.brad.ac.uk/ ~pjiang/ wime/ .

Xi and Zhang (2002). Rolling path planning of mobile robot in a kind of dynamic uncertain environment. *Acta Automatica Sinica* 28(2): 161-175.

Zhang (2008). TOS_MAC Driver based on CC2420 radio chip.



Advances in Robot Navigation

Edited by Prof. Alejandra Barrera

ISBN 978-953-307-346-0
Hard cover, 238 pages
Publisher InTech
Published online 15, June, 2011
Published in print edition June, 2011

Robot navigation includes different interrelated activities such as perception - obtaining and interpreting sensory information; exploration - the strategy that guides the robot to select the next direction to go; mapping - the construction of a spatial representation by using the sensory information perceived; localization - the strategy to estimate the robot position within the spatial map; path planning - the strategy to find a path towards a goal location being optimal or not; and path execution, where motor actions are determined and adapted to environmental changes. This book integrates results from the research work of authors all over the world, addressing the abovementioned activities and analyzing the critical implications of dealing with dynamic environments. Different solutions providing adaptive navigation are taken from nature inspiration, and diverse applications are described in the context of an important field of study: social robotics.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen