Robotica (2014) volume 32, pp. 477–499.
doi:10.1017/S0263574713000805

A snake-based scheme for path planning and control with constraints by distributed visual sensors

Y. Cheng†*, P. Jiang‡ and Y. F. Hu†

†*School of Engineering, Design and Technology, University of Bradford, Bradford BD7 1DP, UK*
‡*Department of Computer Science, University of Hull, Hull HU6 7RX, UK*

(Accepted July 11, 2013. First published online: August 9, 2013)

SUMMARY

This paper proposes a robot navigation scheme using wireless visual sensors deployed in an environment. Different from the conventional autonomous robot approaches, the scheme intends to relieve massive on-board information processing required by a robot to its environment so that a robot or a vehicle with less intelligence can exhibit sophisticated mobility. A three-state snake mechanism is developed for coordinating a series of sensors to form a reference path. Wireless visual sensors communicate internal forces with each other along the reference snake for dynamic adjustment, react to repulsive forces from obstacles, and activate a state change in the snake body from a flexible state to a rigid or even to a broken state due to kinematic or environmental constraints. A control snake is further proposed as a tracker of the reference path, taking into account the robot's non-holonomic constraint and limited steering power. A predictive control algorithm is developed to have an optimal velocity profile under robot dynamic constraints for the snake tracking. They together form a unified solution for robot navigation by distributed sensors to deal with the kinematic and dynamic constraints of a robot and to react to dynamic changes in advance. Simulations and experiments demonstrate the capability of a wireless sensor network to carry out low-level control activities for a vehicle.

KEYWORDS: Snake algorithm; Wireless visual sensors; Mobile robot navigation; Distributed robot system.

1. Introduction

A large body of robotics research to date has focused on the development of a large and smart "brain" to enable robot autonomy.[1–4] They are, however, facing a bottleneck of complexity due to the uncertainties in an unstructured environment. Although the techniques have been developed fruitfully,[5] they are still not robust enough to deliver autonomous robots for our daily life. Steering away from this autonomous intelligence approach, this paper investigates the collaborative control strategy of distributed sensors in an intelligent environment to alleviate the uncertainties faced by a robot. When sensors are distributed in an environment to provide pervasive intelligence,[6,7] a mobile robot with less on-board intelligence can exhibit a high degree of mobility. It relieves the massive requirement for centralized computation on-board the robot into a distributed sensor network where the sensor nodes can provide distributed information and processing capability to the robot.

Pervasive intelligence in an environment can significantly simplify decision making and control in many applications, which has led to a new wave of research on cyber-physical systems.[7,8] One example is the Ubiquitous Robotic Companion (URC), which was developed to enable automated integration of networked robots into ubiquitous computing environments.[9] Robotic ecology was introduced in the PEIS project[10] to enable robots to interact with distributed sensors and actuators in an intelligent environment. This approach simplified complex tasks of on-board functionalities by interactions with inter-connected PEIS components in the environment. Cloud robotics has recently emerged to expand a robot's knowledge beyond its physical body, so that a robot can become smaller,

* Corresponding author. E-mail: y.cheng4@bradford.ac.uk

cheaper, and smarter.[11] In these systems, pervasive intelligence provided semantically interpretable information to a robot for decision making at the symbolic level[9,12] but left low-level skills such as navigation to the robot. There is a trade-off between distributed intelligence and robot on-board intelligence. For the navigation of a simple and cheap robot, which has very limited on-board intelligence, a system with wireless visual sensors distributed in a building was developed in a project WiME (Wireless Mosaic Eyes)[13] by the authors. In this system, each sensor covers a small area in the environment and the wireless sensors were organized to a purposed network for navigation, where all navigation functions, including perception, localization, path planning, and motion control, were conducted by wireless sensors in the environment rather than on-board robots. A mobile robot controlled by such a visual sensor network is only equipped with a wireless receiver for command receiving and actuator driving but without a high computational capability on board. Deploying such a sensor network in an environment is economically feasible when a large number of vehicles need to be controlled in a certain area, such as navigation of wheelchairs in a care centre and Automated Guided Vehicles (AGVs) around a production line, trading environment intelligence for expensive on-board intelligence in every robot. This paper presented the algorithms developed in the WiME project to achieve pervasive intelligence guided navigation.

Autonomous navigation is an application which would benefit greatly from external intelligence. The pervasive intelligence provided by the sensors can greatly simplify the complexity faced by a robot in an environment, which causes the robust problem and hinders the broad application of autonomous robots. 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 localization using 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. Wireless sensors have been used to provide navigation services in several applications. Intelligent transport systems with distributed sensors can provide vehicles with information about the conditions in its surrounding environment,[14] and they can even provide vehicles with navigational services.[15] Wireless sensors can also actively locate and guide the visually impaired to avoid risk[16] and navigate people out of dangerous areas.[17] The 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,[18,19] routing,[20] or event-driven behaviour coordination.[21] A significant number of tasks of low-level perception and control were left to the robots, which have not been well studied. This paper is an attempt to develop techniques for low-level navigation control of a robot or a vehicle assisted by distributed environment intelligence.

To achieve autonomous navigation, a hierarchical architecture is often used to coordinate activities, for example the Nested Hierarchical Controller (NHC).[22] There are three main functions, i.e. SENSE, PLAN, and ACT, to be designed and implemented. A world model, for example the map of a building, has to be defined offline or leant through sensor fusion. Based on the world model, the PLAN carries out mission planning for setting a goal, navigation planning for generating discrete motion sequence and pilot planning for generating continuous trajectories to drive a robot using the ACT. All of them rely on real-time sensor readings in the SENSE. The SENSE block is used to perceive the environment and link the physical world with the virtual world model in a computer, which has to sample raw sensor data, extract low-level features, and retrieve high-level semantics about the world. If an environment is deployed with wireless sensors for navigation, the topology of the wireless sensor network forms a nature world model and a robot does not need to maintain it on-board. The mission planning in PLAN is a command to a destination, such as "go to John's office," sent wirelessly to the sensor network. The navigation planning task of PLAN can be accomplished by saving a routing table in every wireless sensor; for example a multiple Bloom filter[23] was used in the WiME project. This paper presents navigation techniques to implement visual detection in SENSE, pilot planning in PLAN and control in ACT in a wireless sensor network. This is a challenging task as navigation requires high real-time performance but wireless sensors are often resource-scarce, with limited memory, computational power, and communicational bandwidth. Therefore, a snake-based approach is proposed in this paper to accomplish path planning, trajectory generation, and motion control in a distributed wireless sensor network. Working in this way, a vehicle with less on-board intelligence can navigate under guidance from an intelligent environment.

Snakes, also termed as active contour models,[24] 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,[25] virtual springs,[26,27] and snake planner for redundant manipulators.[28] 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 of this research since it requires only neighbouring 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 has been ignored. For example, Zhou *et al.*[29] proposed a snake-based controller for the correction of the tracking paths of wheelchairs. It uses on-board sensors and therefore is a local motion controller. In fact, sensors distributed in an environment provide an infrastructure to carry out both global path planning and local motion control, which can deal with dynamic changes in the environment and predict future uncertainties more effectively.

The control technique proposed in this paper is a distributed solution for both path planning and motion control of a mobile robot possessing minimum on-board intelligence. Instead of relying on centralized control in the robot, the networked wireless visual sensors are deployed and used to perceive the environment. The perceived information is processed and fused through collaboration among the networked sensors to dynamically form an optimal and collision-free R-snake (Reference snake) serving as a reference path for the robot to track. An A-snake (Accompanied snake) algorithm is proposed as a feedback control mechanism to guide the mobile robot subject to constraints to follow the R-snake. A predictive control scheme is further proposed for the optimization of the A-snake tracking, considering dynamic constraints and future path shape on the way. The time optimal control is attempted in this paper; but the predictive control approach can be extended to other performance indices, such as energy optimization and smoothest driving. The novelty of this approach can be summarized as follows:

(1) The snake can be an efficient distributed implementation for robot navigation. Each resource-scarce wireless sensor is only in charge of a local area for local perception and path evolution to respond dynamic changes in the environment.
(2) The snake can be an efficient coordination mechanism to link local sensors together to form a global path in the environment, taking into account robot kinematic and dynamic constraints. Adjacent wireless sensors are cooperated together by the snake with less communication.
(3) Predictive control can be an effective method to overcome communication delay during robot real-time tracking. To overcome the slow communication rate when sending control commands from the wireless sensor network to the robot, the robot is driven to follow a predicted trajectory between any two communication frames. The predicted control is optimal based on the latest information received from the wireless network.

The rest of the paper is organized as follows. Section 2 presents the reference snake algorithm to form a collision-free path for a robot with a curvature constraint. The A-snake controller is discussed in Section 3 followed by predictive control for A-snake tracking in Section 4. Simulation and experiment results and conclusions are given in Sections 5 and 6, respectively.

2. R-snake as a Path Planner

When distributed visual sensors are deployed in an in-door environment along passages for robot navigation, every sensor observes a small area and can be linked via wireless communication to form a topological map for the navigation. It greatly confined the search space in individual sensors for path planning. For a given destination, a reference path, called reference snake (R-snake), is generated as a series of control points distributed in visual sensors or image planes to form a collision-free path to the destination. Bloom filter-based routing[23] links sensors into a chain from robot current position to the destination.

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

Wavefront expansion in a cell decomposition map and potential field in a continuous map are two typical methods used in autonomous robots.[1] The methods focus on collision-free path finding. One of the basic yet important requirements of mobile robot path planning is to satisfy the mobile robot kinematic constraints during the planning phase.[30] Research efforts have been focused on satisfying the robot non-holonomic and curvature constraints. For example, Liang *et al.*[31] presented a non-holonomic path planning method considering curvature constraint and length minimization for a car-like robot based on cubic spirals. Nelson[32] presented two types of continuous steering functions to generate continuous curvature curves: (a) Cartesian quintics for lane changes and (b) polar splines for symmetric turns of arbitrary angle. Based on the latter, Ge *et al.*[33] investigated the use of a polar polynomial curve to construct a path that changes continuously in curvature and satisfies the dynamic constraints. Recently, stochastic trajectory planning algorithms such as CHOMP[3] and STOMP[4] were proposed as efficient algorithms for planning in a high-dimensional and complex space. CHOMP used gradient method to search for optimal solution considering the smoothness of a trajectory over a segment of the trajectory. It has been successfully applied in motion planning of high-dimensional manipulation systems. The STOMP algorithm is a very general planning algorithm and can take into account system dynamics and constraints. These algorithms are more suitable to planning in a complex environment for a high degree of freedom (DOF) robot. They often need several hundred iterations to find a path, which hinders their applications in distributed sensors having limited computational and communicational capability. In fact, deploying sensors in an environment has greatly reduced the uncertainties and confined the search space for the planning. Computation for global optimization can be avoided. Sampling-based planning algorithms[34] are more appealing to real-time applications, using random sampling and graph growing. They can deal with various constraints and can be adopted as a global planner to generate initial R-snake if the environment has complex obstacles, although non-holonomic constraints of mobile robots have not been considered. Laumond *et al.*[35] presented a planning algorithm for non-holonomic robots, which includes collision-free path generation without considering non-holonomic constraints, linking two configurations to transfer a robot to the path, and optimization to have near-minimal length. The approach was efficient because it separated constrained planning from collision-free path planning. However, it used a centralized implementation to implement all the three steps and did not consider dynamic constraints for robot control. For distributed path planning using wireless sensors, especially in a dynamically changing environment, the lack of centralized information makes it much harder to plan a path to reach a global goal in real time, which conforms to various constraints. This paper takes a similar approach to separate constrained planning from collision-free path planning. The R-snake generates a collision-free path first and then an A-snake is generated to approach the R-snake with non-holonomic constraints. Dynamic constrained prediction control is further applied to control robot movement. The proposed snake approach is in fact a cooperation mechanism to support distributed planning and takes into account kinematic and dynamic constraints, including non-holonomic constraints.

2.1. Snake model

Let p_i be a point representing Cartesian configuration (x_i^{cp}, y_i^{cp}) in spaceR^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 spaceR^2. A snake is created by connecting adjacent coordinates sequentially. Each p_i is a control point, which can be moved by exerted internal and external forces from obstacles and other 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 with a safe distance from m obstacles $q_1, ..., q_m$, satisfying a given curvature constraint and maintaining the shortest path length from its start point to the goal point via intermediate control points.

Define 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; the total energy, E^{snake}, of a snake can be expressed as

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

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

where the internal energy, $E^{\text{internal}} = E^{elastic} + E^{curvature}$, is concerned with the intrinsic actions of the snake, such as its shape and length, while the external energy, $E^{\text{external}} = E^{obstacle}$, is concerned with the effect from the environment, such as obstacles.

The first term in the internal energy is defined as elastic energy. The associated elastic force attracts control points each other in order to minimize the energy. This force causes contraction of the snake to reduce the length. The elastic energy is expressed as

$$E_{\text{elastic}} = k_e \cdot \sum_{i=1}^{n-2} (\|p_i - p_{i-1}\|^2 + \|p_{i+1} - p_i\|^2). \tag{2}$$

To impose the curvature constraint on snake control point i, its preceding node and succeeding node exert pulling forces, t_{i-1} and t_{i+1}, on it to reduce the bending of two line segments. When the bending angle, i.e. the curvature, is small, $-\delta < \theta < \delta$, the forces are defined to be zero. Thus, the corresponding bending energy can be defined as below with angle θ in the interval $(-\pi, \pi]$:

$$E_{\text{curvature}} = \begin{cases} k_c \cdot (|\theta| - \delta)^2, |\theta| > \delta \\ 0, \quad \text{others} \end{cases} \tag{3}$$

where $\delta \in R^+$ represents the maximum angle to have zero potential; k_c is a positive coefficient.

The external energy is proportional to the distance between obstacles and the control points. When the distance is more than a barrier d_0, the energy reduces to zero. The repulsive external energy is defined as

$$E_{\text{obstacle}} = \begin{cases} k_o \cdot \sum_{t=0}^{m} (||p_i - q_t|| - d_0)^2, d < d_0 \\ 0, \quad \text{others} \end{cases} \tag{4}$$

where m is the number of obstacles.

To minimize the energy of a snake, p_i needs to move along the negative energy gradient direction. The total force, F^{snake}, exerted on it, in terms of E^{internal} and E^{external} can be expressed as

$$F^{\text{snake}} = F^{\text{internal}} + F^{\text{external}} = (-\nabla \eta_{\text{int}} E^{\text{internal}}) + (-\nabla \eta_{\text{ext}} E^{\text{external}}),$$

where η_{int} and η_{ext} are positive gains to represent the force strength and ∇ is the gradient operator.

Let f^o be the obstacle force, f^e be the elastic force and $f^{\text{curvature}}$ be the curvature constraints force. We have the total resultant force:

$$F_i^{\text{snake}} = f_i^o + f_i^e + f_i^{\text{curvature}} \tag{5}$$

The detailed obstacle, elastic and curvature forces can be found in ref. [36].

2.2. Three states of a snake

In a distributed environment, a snake is segmented into small portions among wireless visual sensors and deformed by exerted local forces. Although dedicated curvature forces are designed to reduce its bending, the curvature constrains of a robot can be violated if segments of the snake in different sensors are not coordinated properly. In comparison with the global coordination in the centralized snakes,[25,29] a distributed coordination mechanism is essential for the applications in wireless sensor networks. The mechanism can be referred to the vertebras of a snake. Different from a rubber band, a snake body can become rigid when it bends to a certain limit. Further bending will cause the vertebra to break. Hence, three states are defined for a distributed snake.

Flexible state, in which a control point can be moved freely by the forces in (5), from the obstacles and the neighbouring nodes. This state occurs when the curvature of a control point is less than the maximum allowed curvature threshold or the resultant force is trying to reduce the curvature.

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

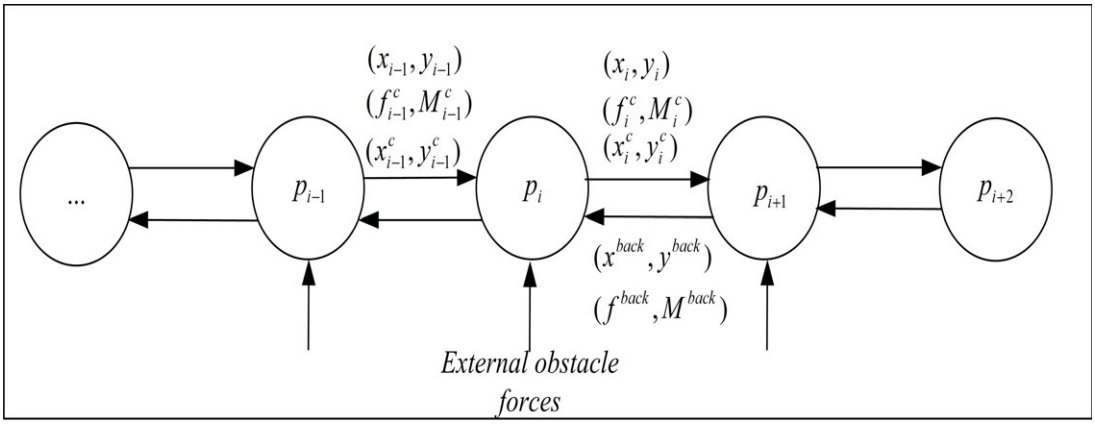

Fig. 1. Information exchange flow between control points.

Rigid state, in which a control point and its neighbours move as a rigid body under the forces from surrounding obstacles and the neighbouring nodes. The relative positions between control points of a rigid body remain unchanged during its moving in order not to violate the curvature constraint.

Broken state, in which a snake segment in the rigid state is broken under a big force from obstacles due, for example, a moving obstacle has pushed away the snake as a whole too much from its initial path or the curvature constraint has to be completely violated for the robot to traverse. After the segment is broken, the snake will be recovered by searching an alternative safe path in the local area. It is concerned with both global path cost and local curvature constraint.

In order to coordinate distributed control points for state transition, every control point is encapsulated into a software component which can exchange information with other control points through communication. Intuitively, the interactions between neighbouring segments of a nature snake body through the inner force determined by their relative positions. This is the case under the flexible state, where control points are adjusted freely without violating any constraints by exchanging coordinates or inner forces with other control points. However, more data exchanges are required when the control points are under the rigid state. Because these control points are linked rigidly, their motions have to be synchronized and, as a result, signals need to be passed to all connected control points which are in the rigid state. The information exchange flow between control points p_i is shown in Fig. 1.

Let (x_{i-1}, y_{i-1}), (x_i, y_i), and (x_{i+1}, y_{i+1}) be the coordinates of control points p_{i-1} p_i and p_{i+1} respectively. From the definition of the energy and force functions in Section 2.1, the coordinates are the information required by control point p_i from p_{i-1} and p_{i+1} when it is in the flexible state. If only elastic force exists, the snake will shrink to its minimum length. If external obstacle forces exist, the snake will avoid obstacles and evolve to equilibrium according to the exerted forces. If there is an intruder detected by a wireless sensor, the snake will dynamically deform in such a way that each control point will exert a force to its neighbours until the force flow passes through all nodes, to reach equilibrium again.

Points p_{i-1}, p_i, and p_{i+1} in the rigid state imply that these three control points have reached or exceeded the curvature constraint. They will behave collectively as a single rigid body with unchanged relative positions. Points p_{i-2} and p_{i+2} will also be part of this rigid body in order to maintain the curvature constraints for p_{i-1} and p_{i+1}. For a rigid body, applied forces can be equivalent to a single resultant force and a resultant moment at a point. Thus, each control point p_i in the rigid body will add up coordinates and force/moment received from its preceding node p_{i-1} to have (x_i^c, y_i^c) and (f_i^c, M_i^c), and then pass them to its succeeding node p_{i+1}. When the last node in the rigid body is reached, it will pass back the final centroid coordinates $(x^{\text{back}}, y^{\text{back}})$ and the resultant forces and moment $(f^{\text{back}}, M^{\text{back}})$ to the preceding nodes in order to generate movements for all control points as a whole rigid body.

A state machine is developed to manage the state transition, as shown in Fig. 2.

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

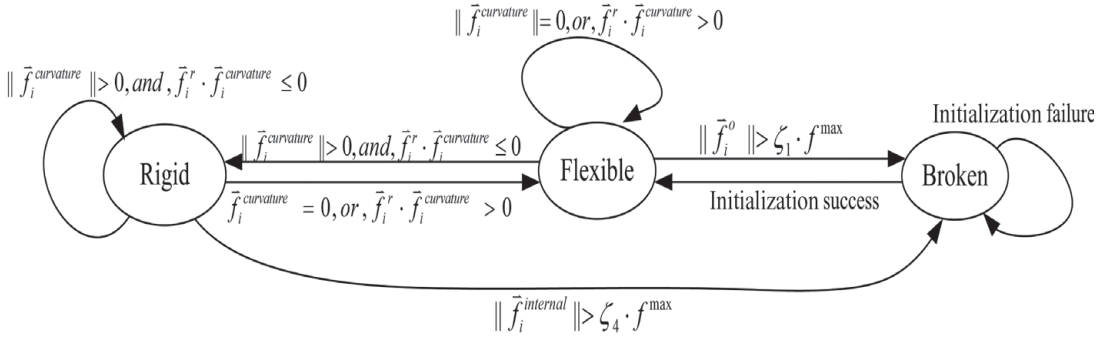

Fig. 2. State machine for state transition.

2.2.1. Flexible state to rigid state. The curvature force is designed to resist bending. If a control point moves within a limited range, as defined in Equation (3), the curvature force is 0. If the curvature force $\vec{f}_i^{\text{curvature}}$ becomes nonzero and the resultant force vector $\vec{f}_i^{r}$ of p_i is in the opposite direction of the curvature vector, i.e. to bend more, this control point will change from its current flexible state to the rigid state. The transition condition can be expressed as

$$\left|\left| \vec{f}_i^{\text{curvature}} \right|\right| > 0 \quad \text{and} \quad \vec{f}_i^{r} \cdot \vec{f}_i^{\text{curvature}} \le 0. \tag{6}$$

2.2.2. Rigid state to flexible state. If the curvature force $\vec{f}_i^{\text{curvature}}$ is zero or the resultant force vector $\vec{f}_i^{r}$ of the control point p_i is in the same direction of the curvature vector, the control point will be in the flexible state with the following transition condition:

$$|| \vec{f}_i^{\text{curvature}} || = 0 \quad \text{or} \quad \vec{f}_i^{r} \cdot \vec{f}_i^{\text{curvature}} > 0. \tag{7}$$

2.2.3. Flexible state to broken state. Regardless of the reason, if the distance between an obstacle and the control point is less than the sum of the control point's radius and the obstacle's radius, the control point is at the risk of colliding with the obstacle. The snake path should enter into the broken state to search a new path. The precondition for this change is as follows:

$$\left|\left| \vec{f}_i^{o} \right|\right| > \zeta_1 \cdot f^{\max}, \tag{8}$$

where ζ_1 is a threshold constant and $f^{\max} = \zeta_2 \cdot (d_{cp} + d_o - \zeta_3)$, ζ_2 and ζ_3 are positive constants, d_{cp} is the radius of the control point, and d_o is the radius of an obstacle.

2.2.4. Rigid state to broken state. In the rigid state, the resultant force exerted on an individual control point is synchronized to $f^{\text{back}} + M^{\text{back}}(p_i - p^{\text{back}})$, where p^{back} is the centroid of the rigid body. Thus, the internal force of p_i, including the elastic force and the curvature force, can be calculated as

$$f_i^{\text{internal}} = \left(f_i^{\text{back}} + M^{\text{back}}(p_i - p^{\text{back}})\right) - f_i^{o}. \tag{9}$$

When the total internal force increases to or above a threshold, due to a very long path or an excessive bending, the control point will enter into the broken state for searching an alternative path:

$$\left|\left| \vec{f}_i^{\text{internal}} \right|\right| > \zeta_4 \cdot f^{\max}, \tag{10}$$

where ζ_4 is a positive constant.

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

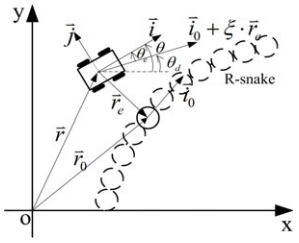

Fig. 3. Coordinate definitions.

2.2.5. Broken state to flexible state. In the broken state, the whole snake should be re-initiated by a global path search algorithm. In our WiME system, Dijkstra search[37] is performed in the visual node on its image plane where the snake is broken to regenerate an R-snake from robot current position leading to the destination. The sampling-based algorithm[34] can be an alternative approach to improve the initialization efficiency, if each visual sensor has to control a large area and the environment is complex. If the re-initiation is successful, a new path is obtained, and all control points will return to the flexible state. Otherwise, another attempt to re-initiate a new path will be made. There is no direct state transfer from the broken state to the rigid state because a re-initiated snake is always in its flexible state.

The existence of the broken state allows a new snake to be searched if the current snake has evolved to be too poor in terms of path length or bending. It avoids the problem of the conventional snake approaches that the global cost could be very high because of the continuity of a snake with only local gradient search.

3. A-Snake as a Tracker

In order to follow the R-snake, distributed visual sensors need to perceive robot's position and heading direction and guide the robot to correct any tracking error subject to various constraints, such as non-holonomic constraint, limited driving force, and steering torque. The control of a non-holonomic system has been a challenge to control system design, where the system suffers a constraint on its instant velocity, such as the restriction on the lateral velocity of a moving wheel.[35] It was proved that point stabilization cannot be achieved with a smooth and time-invariant state-feedback control law.[38] Therefore, time-varying control laws[39,40] and discontinuous feedback laws[41–43] were developed. The difficulty can be alleviated by generating a reference path for robot to track,[44,45] where both kinematic and dynamic constraints can be taken into account during the reference path planning. The current research usually considers a pre-determined path, although dynamic modification of the path can be carried out to deal with a dynamic environment.[46] However, in a distributed sensor environment, the local views of individual sensors and the asynchrony among them make a sensor node difficult to have steady information about the whole R-snake. A local path growing mechanism, i.e. the A-snake, is proposed to track the R-snake, using less information from other sensors and satisfying kinematic and dynamic constraints during the course. For each sampled image, an A-snake will grow from the current sensor node, observing the robot along the R-snake with robot's maximum driving power.

3.1. Desired direction to approach the R-snake

Define a coordinate system as shown in Fig. 3. The doted circles represent the R-snake control points to be tracked by the robot. Here, θ_d is a desired direction which the robot should follow for reducing any deviation from the R-snake, θ is the robot's current direction, $\vec{i}$ and $\vec{j}$ are the unit tangential vector and normal vector of the robot, $\vec{i}_0$ is the tangential direction on the nearest reference control point, $\vec{r}$ is the robot's location vector, $\vec{r}_0$ is the vector of the nearest control point on the R-snake, θ_e is the direction error between the desired direction and the robot's current direction, and ξ is a proportional gain for the deviation control. An A-snake is a local path from the robot current position and direction converging to the R-snake, subject to robot's dynamic and kinematic constraints.

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

To determine a proper desired direction θ_d, which can lead a robot to track the R-snake, assume the position error vector between the robot current position and the R-snake to be

$$\vec{r}_e = \vec{r}_0 - \vec{r}. \tag{11}$$

Construct a Lyapunov function V such that

$$V = \vec{r}_e^{\,T} \cdot \vec{r}_e \geq 0. \tag{12}$$

The derivative of the above function with respect to the arc length s can be obtained as

$$\frac{\partial V}{\partial s} = 2 \cdot \vec{r}_e^{\,T} \cdot \frac{\partial \vec{r}_e}{\partial s}. \tag{13}$$

Substitute (11) into (13), one can get

$$\frac{\partial V}{\partial s} = 2 \cdot \vec{r}_e^{\,T} \cdot \left(\frac{\partial \vec{r}_0}{\partial s} - \frac{\partial \vec{r}}{\partial s} \right) = 2 \cdot \vec{r}_e^{\,T} \cdot (\vec{i}_0 - \vec{i}).$$

If the robot's direction can be controlled according to

$$\vec{i} = \vec{i}_0 + \xi_1 \cdot \vec{r}_e \tag{14}$$

then

$$\frac{\partial V}{\partial s} = -2 \cdot \vec{r}_e^{\,T} \cdot \xi_1 \cdot \vec{r}_e \leq 0. \tag{15}$$

According to Lyapunov theorem and from (12) and (15), one can draw the conclusion that if the robot's heading towards the direction of (14), i.e. $\theta_d = \angle \vec{i}$, the position error between the robot and the R-snake will be reduced to zero asymptotically.

3.2. Dynamic model and the constraint to the A-snake

However, due to the limited steering torque and non-holonomic constraint of a wheeled robot, the robot cannot change its heading direction instantly. An A-snake is introduced as a guide path for the robot to approach the desired direction θ_d defined by the vector $\vec{i}$ in (14). The motion of a wheeled robot can be modelled as

$$\begin{cases} \dot{\vec{r}} = v \cdot \vec{i}, \\ \omega = v \cdot k, \end{cases} \tag{16}$$

where $\vec{r} = (x, y)^T$ is the robot position vector, $\vec{i}$ is the unit vector of its tangential direction, v is the robot velocity, ω is the robot angular velocity, and k is the trajectory curvature.

After differentiating the above equation, the acceleration and the angular acceleration of the robot are given as follows:

$$\begin{cases} \ddot{\vec{r}} = \dot{v} \cdot \vec{i} + v\omega\vec{j}, \\ \dot{\omega} = \dot{v}k + v\dot{k}, \end{cases} \tag{17}$$

where $\vec{j}$ is the unit normal vector of the robot.

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

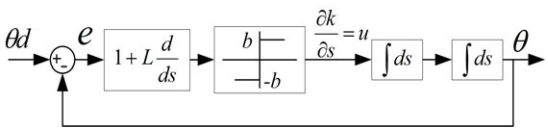

Fig. 4. Heading direction generation of an A-snake.

According to Newton's law, the dynamic equation of a wheeled robot can be obtained as

$$\begin{cases} F^d = M\dot{v}, \\ f^{\text{friction}} = Mv\omega = Mv^2k, \\ \tau = J(\dot{v}k + v\dot{k}) = J\left(\dot{v}k + v^2\frac{\partial k}{\partial s}\right), \end{cases} \tag{18}$$

where F^d is the driving force of the robot and τ is the steering torque; f^{friction} is the lateral friction on the wheels; s is the arc length along the snake, M and J are the mass and the inertia of the robot.

From (18), one can see that if τ is bounded, there is a trade-off between the driving velocity v and the derivative of curvature $\partial k/\partial s$. If $\partial k/\partial s$ is very large, the driving speed has to be very low. This is equivalent to the situation that a robot can make a sharp turn only if it is very slow. For a smooth and fast driving, considering a saturated torque $\tau_{\max}$, a limitation on $\partial k/\partial s$ is imposed as follows:

$$\left|\frac{\partial k}{\partial s}\right| < b, \tag{19}$$

where b is a positive constant.

3.3. A-snake generation

A nonlinear control algorithm as shown in Fig. 4 is designed to generate the heading direction θ to follow the desired θ_d for a robot with a limited steering torque. Let $u = \partial k/\partial s$, the generated direction θ can always satisfy constraint (19) because $|u| < b$.

THEOREM 1. *Let L be a constant coefficient factor, $L \in R^+$, e be the orientation error between θ_d and the robot direction θ, b be the maximum curvature derivative a robot can follow. If the given desired direction θ_d is constant, then e and $\partial e/\partial s$ converge to 0 with oscillation by the control scheme defined in Fig. 4.*

Proof. The orientation error between θ_d and the robot direction θ is defined as □

$$e = \theta_d - \theta. \tag{20}$$

If θ_d is constant, then

$$\frac{\partial^2 e}{\partial s^2} = -\frac{\partial^2 \theta}{\partial s^2} = -\frac{\partial k}{\partial s} = -u. \tag{21}$$

From Fig. 4, let

$$Q = L \cdot \frac{\partial e}{\partial s} + e. \tag{22}$$

Then, we can generate the curvature derivative u for the robot to track as

$$\frac{\partial^2 \theta}{\partial s^2} = u = b \cdot \text{sign}(Q), \tag{23}$$

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

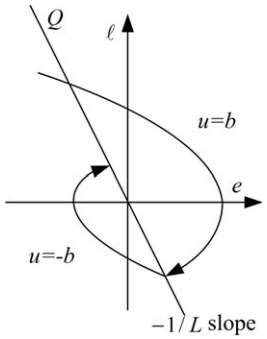

Fig. 5. Phase track of e and ℓ.

where

$$\text{sign}(Q) = \begin{cases} -1, & Q < 0 \\ 1, & Q \geq 0 \end{cases}$$

so that constraint (19) is satisfied.

Considering the second-order system (21), the phase plane trajectory can be derived as

$$\frac{d\ell}{de} = -\frac{u}{\ell},$$

where $\ell = \frac{\partial e}{\partial s}$. Rewriting the above equation gives

$$\ell d\ell = -ude.$$

Integrate it to obtain

$$\tfrac{1}{2}\ell^2 = -u\,(e + C),$$

where C is a constant. The trajectories in the phase plane ee–ℓ shown in Fig. 5 are parabolas intersected with the switch line Q, and the direction of the trajectory is clockwise. If $L > 0$, then e and $\ell = \partial e/\partial s$ converge to 0.

Theorem 1 provides a mechanism to generate robot heading direction to follow the desired θ_d under constraint (19). However, the result is for a constant θ_d which is not the case in general. The following theorem states that the convergence can still be guaranteed if the change of θ_d is limited in a range.

THEOREM 2. *If a change of direction θ_d satisfies the condition of $|L\frac{\partial^2\theta_d}{\partial s^2} + \frac{\partial e}{\partial s}| < L \cdot b$, sliding mode happens on the switch line Q so that e and $\partial e/\partial s$ converge to 0 along the sliding plane by the control scheme in Fig. 4.*

Proof. Let (22) be the sliding plane of sliding model control.[47] □

The sliding mode condition can be verified as

$$Q \cdot \frac{\partial Q}{\partial s} = Q \cdot \left(L \cdot \frac{\partial^2 e}{\partial s^2} + \frac{\partial e}{\partial s} \right). \tag{24}$$

From (20) and (23), one has

$$\frac{\partial^2 e}{\partial s^2} = \frac{\partial^2 \theta_d}{ds^2} - b \cdot \text{sign}\,(Q)\,.$$

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

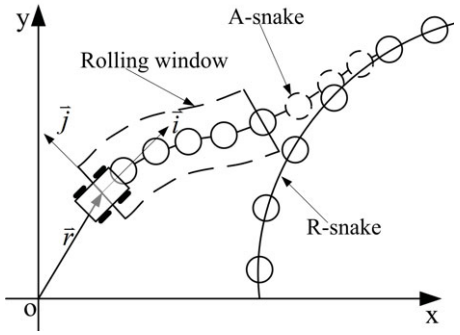

Fig. 6. Robot, A-snake and rolling window.

Thus,

$$Q \cdot \frac{\partial Q}{\partial s} = Q \cdot \left(-L \cdot b \cdot \text{sign}(Q) + L \frac{\partial^2 \theta_d}{\partial s^2} + \frac{\partial e}{\partial s} \right) = -L \cdot b\,|Q| + Q \cdot \left(L \frac{\partial^2 \theta_d}{\partial s^2} + \frac{\partial e}{\partial s} \right). \tag{25}$$

$$\text{If } \left| L \frac{\partial^2 \theta_d}{\partial s^2} + \frac{\partial e}{\partial s} \right| < L \cdot b, \text{ one can get } Q \cdot \frac{\partial Q}{\partial s} \leq 0.$$

The sliding mode happens with a sliding plane $Q = 0$, which leads e and $\partial e/\partial s$ converge to 0.

Therefore, the proposed control scheme in Fig. 4 can be a tracker of θ_d for a robot to track the R-snake. The generated heading direction θ can always satisfy the constrained derivative of curvature. However, in a wireless visual sensor, the sampling rate could be low due to limited computational capacity and *ad hoc* communication. The robot has to carry out an inner model[48] based tracking if the new sensing information is not available. The internal simulation is conducted to generate an A-snake in order to track the stored R-snake:

(1) capture an image and extract the robot's current position $\vec{r}(0)$ and $\vec{i}(0)$ from the image;
(2) determine the robot's deviation $\vec{r}_e$ from the nearest control point on the R-snake and the corresponding direction $\vec{i}_0$ of the R-snake. The desired θ_d is obtained by (14) for R-snake tracking;
(3) feed θ_d into the controller in Fig. 4 to generate the heading direction θ and the associated k for the robot to track;
(4) using the robot kinematics in (16) with $v = \dot{s}$ to grow the A-snake:

$$\frac{d\vec{r}}{ds} = \vec{i} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}. \tag{26}$$

The robot's new position can be calculated with a step Δ: $\vec{r}(s + \Delta) = \vec{r}(s) + \int_s^{s+\Delta} \vec{i}\,ds$;
(5) go to step (2) to grow the A-snake for n steps $\{\vec{r}(0), ..., \vec{r}(n\Delta)\}$, as shown in Fig. 6.

When the next image is sampled, a new A-snake will be generated from (1) to (5) again. This process is repeated and can be extended to the successive sensor nodes if needed. Therefore, the A-snake can be considered as an inner controller for the R-snake tracking to cope with a slow sampling rate of wireless sensor networks, satisfying the non-holonomic constraint of a wheeled robot with a bounded steering torque.

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

4. Predictive Control for Optimal A-Snake Tracking

The A-snake provides a constraint-compliant path for a robot to track. The path needs to be provided with explicit time scale, i.e. temporal planning,[46] subject to various robot dynamic constraints. The forward velocity provides an additional degree of freedom for the possible optimization of the tracking. However, for conventional autonomous robots, the tracking speed has to compromise with safety due to the limitedly local view of on-board sensors. Optimal tracking is not easy to be practically implemented. In our system, the sensors are pervasive in the environment. The far-sight provided by a distributed sensor network opens up the opportunity for the optimal control of vehicles. A vehicle can respond to a change on its way in advance and be driven with optimal time or energy in a dynamic environment. A predictive control approach is developed in this section to achieve time-optimal tracking, taking into account the geometric features of the future path to be tracked, subject to various dynamic constraints. Predictive control is a technique to achieve optimal control by predicting future system behaviours.[49] One advantage of model predictive control is its ability to handle constraints.[50] For an autonomous vehicle, predictive capability is essential and therefore model-based predictive control was applied to achieve optimal path tracking subject to vehicle constraints,[51–53] where the approach of model predictive control is applied to a linearized robot model for the optimization of a quadratic objective function. For an uncertain and dynamic environment, predictive capability is more important for a mobile robot. A rolling optimization algorithm was thus proposed in ref. [54]. In our project, a dynamic environment is observed by a group of visual sensors, which can have a slow feedback rate. Thanks to the open-loop optimization of predictive control, we can carry on tracking if new update is not available. A rolling window optimization is proposed in this paper for application to a sensor network controlled robot with dynamic constraints.

Define a rolling window with length l along the A-snake as shown in Fig. 6, which could be distributed in several wireless sensors and evolved asynchronously. In every sampling period, the optimal driving force and steering torque are calculated in the window. The l-window will roll forward one step for the next sampled image. Working in this way repeatedly, a vehicle can react to possible risks on its way in advance and use its driving capacity sufficiently.

In a rolling window l, the time-optimal control for a wheeled robot (18) can be formulated as

$$\min_{F^d,\tau}(\Gamma) = \min_{F^d,\tau}\left(\int_0^l 1/v(s)ds\ |_{s\in A}\right) \tag{27}$$

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

$$\begin{cases} |f^{\text{friction}}| < (\mu N)_{\max}, & \text{for non-slippage} \\ |F^d| < F^d_{\max}, & \text{for bounded driving force} \\ |\tau| < \tau_{\max}, & \text{for bounded steering torque} \end{cases} \tag{28}$$

where A is the A-snake, $v(s)$ is the velocity profile of the robot along the A-snake, v_{r0} is the sampled velocity of the robot, f^{friction} is the friction of tires with coefficient μ and normal force N, F^d and τ are the driving force and steering torque of the robot with the upper bounds of $F^d_{\max}$ and $\tau_{\max}$, respectively. The objective function in (27) for predictive control is to minimize the robot's travelling time Γ along the snake from its current location to the end of the l-window. Equation $v(l) = 0$ implies that the robot needs to have the capability to stop at the end of the rolling window, in order to respond to the worst possible circumstance which is not observable through the current rolling window. Other performance indices can be applied for the optimization in the rolling window, for example equation $\min_{F^d,\tau}(\Gamma) = \min_{F^d,\tau}(\int_0^l L(|F|, |\tau|)ds|_{s\in A})$ for a minimum-energy problem, where $L(|F|, |\tau|)$ is the battery consumption function of the robot. A numerical algorithm is developed for efficiently solving the time-optimal problem in this paper.

In order to optimize (27), the area under the velocity profile $v(s)$ needs to be maximized, subject to robot dynamics (18) and constraints (28), which include non-slippage, bounded driving force, and bounded steering torque. A numerical solution is developed for the optimization, which is achieved by finding the maximum uniform velocity in the l-window first and then accelerating/decelerating with the highest driving power to approach the maximum velocity.

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

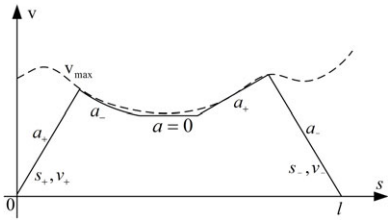

Fig. 7. Rolling window optimization for trajectory generation (assume $v_{r0} = 0$).

4.1. The maximum uniform velocity

From the dynamic model in (18), the maximum speed v of a robot is constrained by the limited friction $|f^{\text{friction}}| < (\mu N)_{\max}$:

$$v_{f\max}^2 = \frac{(\mu N)_{\max}}{M|k|} = \frac{g\mu_{\max}}{|k|} \tag{29}$$

and the bounded steering torque $\tau < \tau_{\max}$:

$$v_{\tau\max}^2 = \frac{\tau_{\max}}{J\,|\partial k/\partial s|}, \tag{30}$$

where $v_{\tau\max}$ is the maximum uniform velocity with $\dot{v} = 0$ substituted into (18). This is the maximum reachable velocity for a robot to travel without any cost on acceleration or deceleration.

Thus, the maximum uniform velocity can be obtained as

$$v_{\max} = \min(v_{\tau\max}, v_{f\max}). \tag{31}$$

In order to optimize the two-boundary-value problem in (27), the optimal $v(s)$ in the l-window is calculated by squeezing the velocity profile from the two boundaries using the maximum acceleration/deceleration, so that the area under $v(s)$ can be maximized, as shown in Fig. 7.

4.2. The maximum acceleration

From the dynamic model in (18), the maximum positive acceleration can be obtained with the force and torque constraints:

$$\begin{aligned} a_{F\max+} &= F_{\max}^d/M, \\ a_{\tau\max+} &= \frac{\tau_{\max}}{J|k|} - \frac{v^2}{k}\frac{\partial k}{\partial s}, \\ a_{\max+} &= \min(a_{F\max+}, a_{\tau\max+}). \end{aligned} \tag{32}$$

If $v \le v_{\max}$, then $a_{\tau\max+} \ge 0$, this implies that a positive acceleration exists. Therefore, the acceleration process has to be bounded by $v_{\max}$ in (31). We have

$$a_{\max+} = \min(a_{F\max+}, a_{\tau\max+}). \tag{33}$$

4.3. The maximum deceleration

Similarly, the maximum negative deceleration can be obtained as

$$\begin{aligned} a_{F\max-} &= -F_{\max}^d/M, \\ a_{\tau\max-} &= -\frac{\tau_{\max}}{J|k|} - \frac{v^2}{k}\frac{\partial k}{\partial s}, \\ a_{\max-} &= \max(a_{F\max-}, a_{\tau\max-}). \end{aligned} \tag{34}$$

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

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

The squeezing optimization is numerically implemented by segmented uniform acceleration/deceleration from the two boundary velocities with an incremental step δ.

- For the acceleration at s_+, forward planning is carried out

$$v_+^2(s_+) = v_+^2(s_+ - \delta) + 2\alpha_{\max +}\delta. \tag{35}$$

- For the deceleration at s_-, backward planning is carried out

$$v_-^2(s_-) = v_-^2(s_- + \delta) + 2\alpha_{\max -}\delta. \tag{36}$$

During the squeezing process, it is needed to ensure that $v_+(s_+)$ and $v_-(s_-)$ in (35) and (36) do not exceed $v_{\max}$ for the segment in between. If this happens at any point $s\#$, the velocity profile for the segment from $v_+(s_+)$ to $v_-(s\#) = v_{\max}(s\#)$ has to be calculated first. The process continues until the acceleration segment and the deceleration segment encounter. The whole velocity profile can be obtained by repeating this squeezing process for the remaining segments. Working in this way, the area of $v(s)$ is maximized and therefore the travelling time is minimized. The generated velocity profile tells the robot when to accelerate or decelerate in advance in order to safely track the dynamic snake in the predictive l-window. The algorithm can be summarized as follows and is shown in Fig. 7:

(1) According to the current A-snake, obtain the maximum uniform velocities $v_{\max}$ from (31) in the rolling window l.
(2) Initialize the squeezing process with the boundary conditions: initial state $s_+ = 0$, $v_+(0) = v_{r0}$ and terminal state $s_- = l$, $v_-(l) = 0$.
(3) Forward/backward planning of v_+/v_- in parallel. If $v_+ \le v_-$, increase v_+ by (35) and $s_+ = s_+ + \delta$. If $v_+ > v_-$, increase v_- by (36) and $s_- = s_- - \delta$.
(4) If $s_+ \neq s_-$ and $v(s) < v_{max}(s)$ for any $s \in [s_-, s_+]$, go to (3).
(5) If $v(s\#) \ge v_{\max}$ at $s\#$ between $s_+ \sim s_-$, create two new segments, $s_+ \sim s_\#$ and $s_\# \sim s_-$ and go to (3) for their planning.
(6) If $s_+ = s_-$ but there are any unplanned segment, go to (3) for their planning;
(7) The full velocity profile $v(s)$, $s \in [0, l]$, is sent to the robot for its control at time t:

$$\begin{aligned} F^d(t) &= Ma_{\max}(0), \\ \tau(t) &= Ja_{\max}(0)k(0) + Jv^2(0)\frac{\partial k}{\partial s}\bigg|_{s=0} \end{aligned} \tag{37}$$

(8) Shift the rolling window one step forward.
(9) For every servo period Γ,$F^d(t + n\Gamma)$ and $\tau(t + n\Gamma)$ will be continuously generated by the robot from the obtained velocity profile $v(s)$, until a new v(s) is received from a vision sensor.
(10) For every image-sampling period, the visual sensor will update the A-snake and go to (1).

The rolling window optimization utilizes information of the A-snake l-distance in advance to achieve time-optimal tracking. The open-loop format of the predictive control also alleviates the difficulty of control latency, which is common in wireless sensor-based applications due to their *ad hoc* communication and limited on-board capacity.

5. Simulation and Experimental Results

5.1. Overview of the experiment components and functional modules

Two key components were implemented in the mosaic eyes supported robot navigation experiment system as shown in Fig. 8: networked *visual sensors* and the *mobile robot*. There are two processing loops in each visual sensor: the SENSE loop is dedicated for image processing to capture the

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

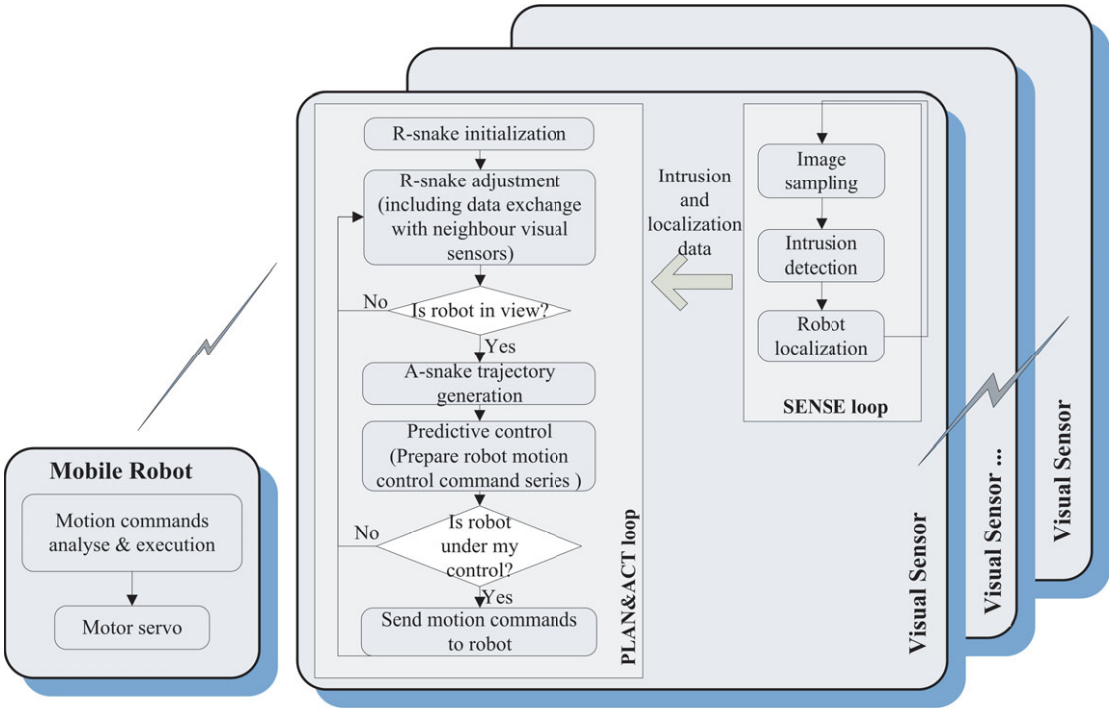

Fig. 8. (Colour online) Experiment system components and functional modules.

foreground images, to detect intrusion by comparing the foreground images with background one and to fulfil the robot localization; the PLAN & ACT loop utilizes the intrusion and robot localization data shared by the SENSE loop and snake segments information exchanged with neighbouring visual sensors to deform the R-snake in real time to maintain a collision-free reference snake to destination. Taking input from the R-snake adjustment module and determined by whether the robot is in its coverage area, the A-snake trajectory generation and predictive control algorithms will selectively run. Only when a robot is observed by a visual sensor, the visual sensor plans a series of motion commands; and only when the robot is under the control of a visual sensor, the visual sensor sends the motion control command series to the robot wirelessly. The mobile robot is responsible for receiving motion control commands from the visual sensors and executing the commands to drive and steer according to the commands.

5.2. Control scheme simulations

In order to verify the effectiveness of the proposed snake-based predictive control, simulations have been carried out, in response to dynamic obstacles in the environment. Figure 9 shows a result for a 0.56 (kg) robot with the constraints of limited curvature derivative $|u| < 2$, the friction coefficient $\mu_{\max} = 0.6$, $F_{\max} = 4.4$ (N), $\tau_{\max} = 2.0$ (N m), and the maximum speed 1.5 (m/s).

The environment is monitored by a mosaic of visual sensors. Through internal force communication among the sensors, an R-snake forms a safe path to a destination. When an obstacle is detected by a visual sensor (shown in Fig. 9a) on the way, the R-snake starts to bend for the avoidance. A rolling window ($l = 50$) is opened for the predictive control. First, the A-snake is generated to guide the robot's tracking with the constrained curvature derivative $|u| < 2$, as shown in Fig. 9(a). The velocity profile is then optimized with $\mu_{\max}$, $F_{\max}$, and $\tau_{\max}$. The solid line in Fig. 9(b) shows the maximum uniform velocity of (31) for the A-snake path. Because of the position error between the robot and the R-snake, the A-snake tries to approach the R-snake with the maximum curvature derivative at the beginning. Due to this high curvature derivative, the maximum velocity in part A is reduced to 0.9 m/s, although the highest speed of the robot is 1.5 m/s. Near the obstacle area, part B of Fig. 9(b), the higher curvature limits the maximum speed to avoid wheel slippage. The rolling window-based prediction thus allows the robot to take actions in advance to avoid the obstacle. After the obstacle

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

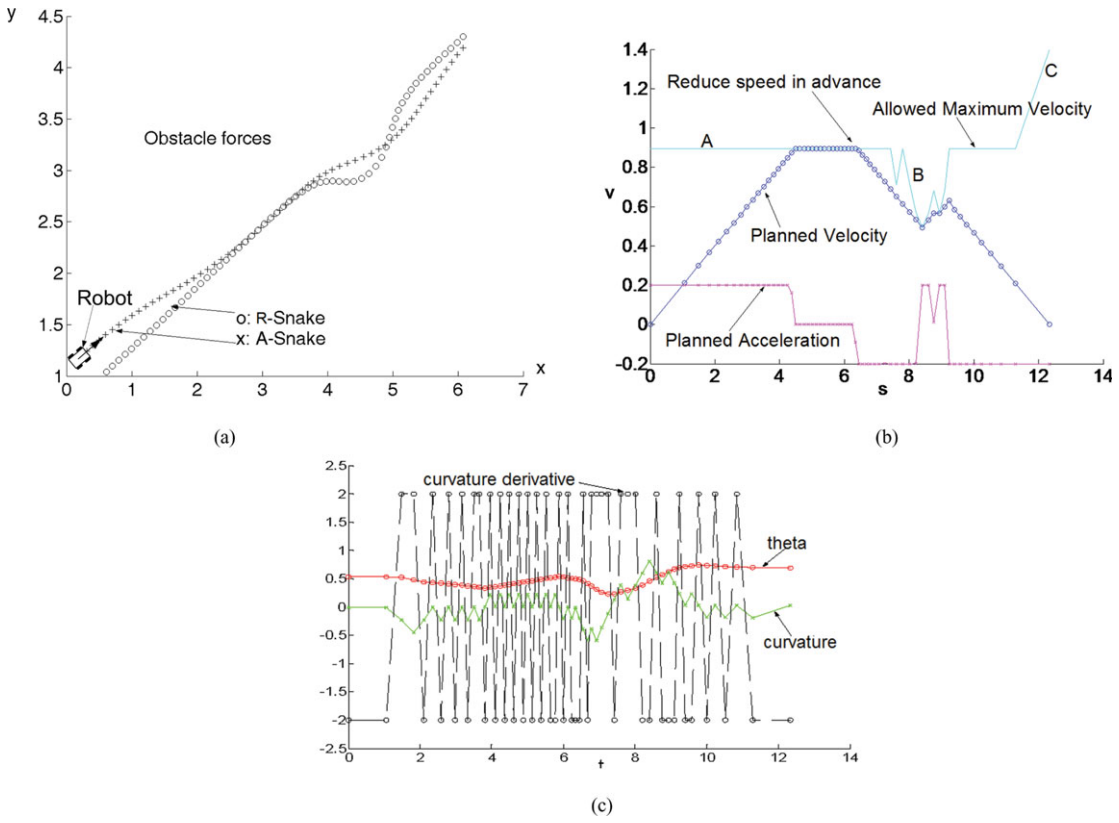

Fig. 9. (Colour online) Trajectory of A-snake. (a) Deformable snake with A-snake. (b) Maximum velocity, planned velocity and acceleration. (c) Planned curvature derivative, curvature and orientation.

area has been passed in part C of Fig. 9(b), the A-snake converges to the R-snake. The smoother path makes the maximum velocity go up to 1.5 m/s.

From Fig. 9(b), one can see that the planned velocity has a similar trend as the maximum velocity, with a zero initial speed for the robot. At the end of the rolling window, the robot is supposed to be stoppable to cope with the worst situation unseen through the rolling window. From the planned acceleration, one can see that the generated acceleration is always within the limitation of 0.2m/s^2.

Figure 9(c) shows the planned curvature derivative, curvature, and robot heading orientation. The curvature derivative switches between -2 and 2 through the bang–bang control scheme as shown in Fig. 4 to correct any directional error.

5.3. Experiments of robot navigation by wireless visual sensors

Trajectory tracking of a model car under control of the WiME network is experimented. Four wireless visual sensors mounted on ceiling are used to form a closed and continuously running circle so that each sensor has a neighbouring sensor on each side, one on the left and another on the right as shown in Fig. 10. The visual sensor is developed based on iMote2 microcontroller board[55] where all path planning and trajectory generation are implemented and all computation tasks are performed. Each visual sensor has a 100-MHz PXA271 XScale processor, 256-kB SRAM, 32-MB flash and 32-MB SDRAM on-board. It is also equipped with OV7620 vision sensor module,[56] which has a maximum resolution of 640×480 and 30 frames per second capturing capability. With the cc2420 transceiver, it is IEEE 802.15.4 communication enabled.

An independent remote control console is set up to communicate with the visual sensors for navigation request and working status monitoring. The model car (mobile robot) is controlled by a Motorola MC9S12DT128B CPU, which is used to execute the received commands from the wireless sensors and control the actuator. The control console and the model car are IEEE 802.15.4 communication enabled too. A communication protocol is developed for R-snake evolving, A-snake

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

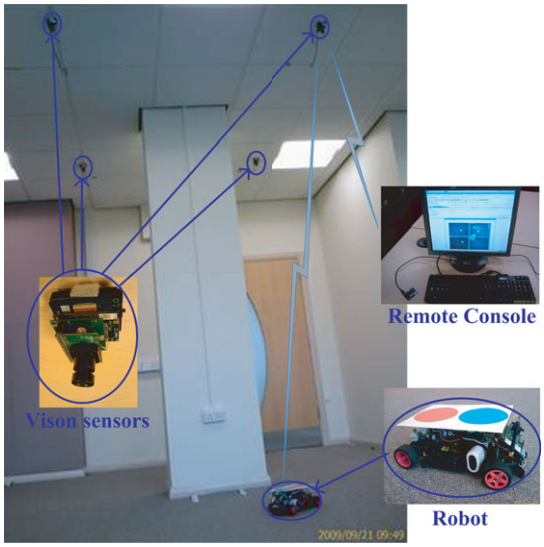

Fig. 10. (Colour online) WiME intelligent environment.

generation, and predictive control between wireless sensors and a robot. The model car is colour marked for facilitating image-based measurement of the robot's position and orientation.

Figure 11 shows the experiments of the proposed planning and control scheme. Each figure displays four images captured by the four sensors, with their identities 30, 40, 50, and 60 from top-right one to count anticlockwise. In Fig. 11(a), 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 R-snake that the robot should follow; the green circles are the generated A-snakes. The white rectangle blobs represent dynamic obstacles. As one can see in Fig. 11(a), the dynamic obstacles are within the views of eyes 30, 40, and 50 but out of the sight of eye 60. In Fig. 11(b), an obstacle appears within the sight of eye (60). At this point, the robot is under the control of eye (30) but eye (30) is not aware of the existence of the new obstacle. With the information exchange between eye (60) and eye (30), the R-snake is updated to avoid the obstacle. In Fig. 11(c), the robot control is handed over from eye (30) to eye (60). The figures show that, with the predictive path updated by eye (30) and with the control of eye (60), the robot has successfully avoided the obstacle (Fig. 11d) and continued to move along the updated R-snake.

The corresponding control and driving velocity are shown in Fig. 12. It shows that the driving force and the steering torque are both kept within the allowable ranges. The sign changes in the steering torque indicate the feedback regulation of the predictive control in order to track the snake, although the predictive optimization is carried out in an open-loop manner for each single prediction in the l-window. The robot speed was 0.76 m/s when data collection started as shown in Fig. 11(a). Since there were no dynamic obstacles in the view of eye (60), the path was stabilized at a straight line to minimize the distance. When the obstacle was detected by eye (60), it propagated this information by deform the path in eye (60) and then further to the path in eye (30) as shown in Fig. 11(b). As far as the robot was concerned, it can maintain a high speed to go through the overlapped area of eye (30) and eye (60). The robot decelerated when it approached the obstacle in order to avoid slippage (Fig. 11c) and resumed a high speed again after passing the obstacle (Fig. 11d).

The average CPU processing times for the software modules in individual visual sensor are listed in Table I. The appearance of the robot does not increase the processing time for image processing modules in SENSE loop whilst A-snake trajectory generation and predictive control modules have significant increments of the processing time.

As seen from Table I, a complete cycle, when the robot is in the coverage area of a visual sensor (with robot in view), takes about 380 ms CPU processing time of the iMote2 PXA271 XScale processor. The total time of a cycle when the robot is out of the visual sensor's coverage (without robot in view) is around 260 ms without robot in view. Given the maximum speed 1 m/s of the model car and set the granularity of each step as 40 ms, one can obtain 0.04 m maximum travel distance

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

Table I. Module processing time in a visual sensor.

Processing module	Without robot in view (ms)	With robot in view (ms)
Image sampling	89.203	89.469
Intrusion detection	146.323	192.927
Robot localization	2.454	2.235
R-snake adjustment (communication time excluded)	10.220	12.990
A-snake trajectory generation	1.797	56.080
Predictive control	1.800	26.111
Communications (sending motion commands to robot and exchanging information with neighbouring visual sensors)	3.803	18.411

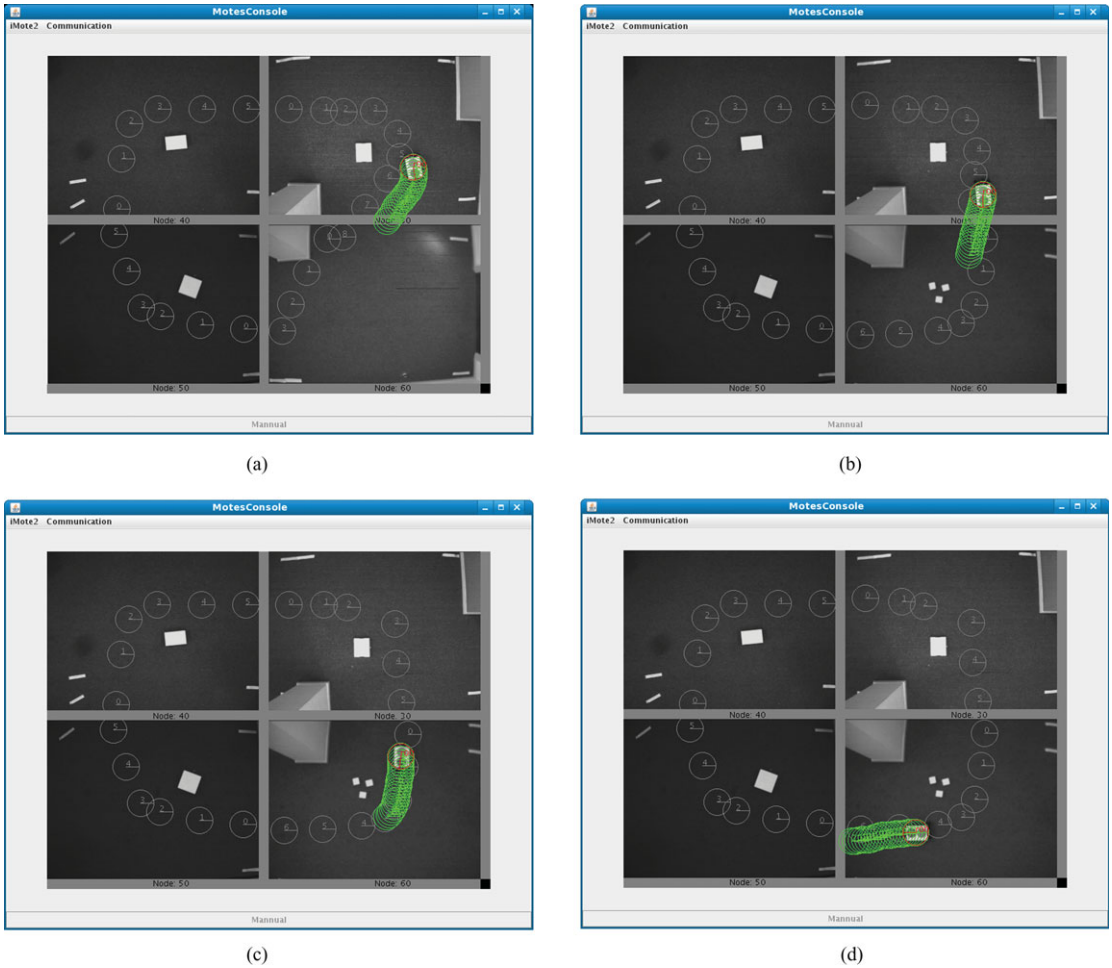

Fig. 11. (Colour online) The real-time experiments of robot control by a visual sensor network. (a) Snake without obstacles. (b) Obstacles appear. (c) Passing obstacle area. (d) Passed obstacle area.

by the robot in each step. The processing time was calculated based on a 15 predicted step series of motion command planned in one cycle. These 15 steps will last about 0.6 seconds, which is about one and half the image processing time and ensures a smooth motion of the robot. Since the PLAN & ACT is a separate processing loop from the SENSE one, and the total processing time of the PLAN & ACT loop is less than 120 ms in one cycle, the Kalman filter algorithm has been implemented to fuse the captured visions with the predicted ones to complement the slow image processing process.

This experiment confirms the effectiveness and capability of the snake algorithm to be implemented on distributed resource-scarce wireless visual sensors to coordinate and achieve a successful global

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

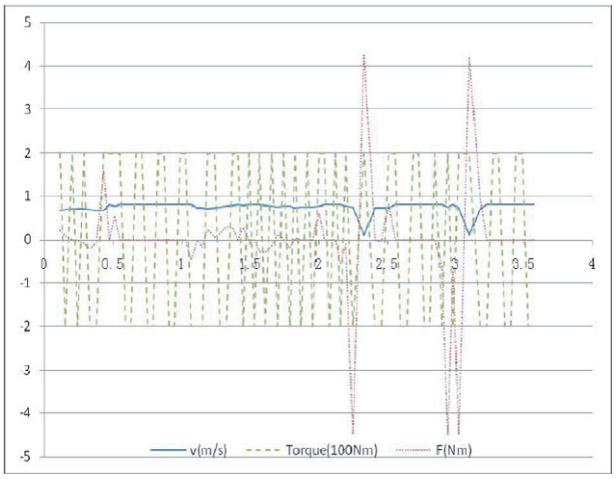

Fig. 12. (Colour online) Robot control and velocity (driving force F_d: dot; steering torque τ: dashed; velocity v: solid).

navigation of a low intelligent robot. In addition, the slow communication rate was overcome by sending a series of motion control commands based on a predicted trajectory.

Theoretically, there is no limit on the number of visual sensors in the system since one visual sensor only processes its own coverage area, communicates with its neighbour nodes, and controls the robot in its local view. By increasing the number of the visual sensor, the total computing capability on the planning coordinated by the deployed snake algorithm could compete with any existing supercomputer for navigation in a large area.

The algorithms presented in this paper were practically implemented with wireless visual sensors mounted on the ceiling for wheelchair navigation in a building. The only processor on-board the wheelchair was an 8-bit Atmega 128L, which was used to link the visual sensor network wirelessly through the IEEE 802.15.4 protocol and to drive the two differential wheels of the wheelchair. Such a low-performance processor is not powerful enough to carry out global navigation. However, the snake distributed in visual sensor network can provide enough distributed intelligence to control the wheelchair. A demonstration video of wheelchair navigation in the building can be found in ref. [13].

5.4. Complexity comparison with centralized algorithms

The snake-based path-planning scheme takes a different approach from the well-known centralized algorithms such as potential field path planning[57] or Laumond's motion planner.[35] It is proposed for the real-time navigation of a mobile robot using a wireless sensor network rather than a centralized controller. Thanks to the communications between neighbouring nodes, a whole path managed by a sequence of sensors can become collision free and satisfy robot constraints in response to dynamic obstacles, where each sensor only handles one segment of the whole path. Similar to the Laumond's motion planner, the proposed scheme generates an R-snake to be collision free from dynamic obstacles and further an A-snake to be non-holonomic.

At first, the R-snake method is compared with the potential field method, which is a typical method for avoiding dynamic obstacles with a set goal. Assume that there are n sensors in an R-snake and each sensor perceives m obstacles at most. Because the R-snake is evolved in n sensors simultaneously, the number of operations for the whole path is $O(m)$. This is the complexity for the obstacles detection and the R-snake deformation. As a comparison with the potential field method, the robot is exerted with repulsion forces from obstacles that are nm in total. Hence, the figure for the potential field method is $O(mn)$, n times higher than the snake-based approach, due to its centralized nature.

To satisfy non-holonomic and curvature constraints, Laumond's method subdivides the initial collision-free path, so that the two configurations of any subdivided path segments are linked by the minimal-length feasible trajectories that must be collision free and respect the constraints. Let ρ be the arclength of the initial path and Δs be the smallest size of path division; the complexity of the algorithm to find a collision-free minimal length curve could reach $O(\rho/\Delta s + K)$, where K is the number of original subpaths. Therefore, it involves a whole path scan and is suitable for off-line

https://doi.org/10.1017/S0263574713000805 Published online by Cambridge University Press

planning in a known environment. The proposed A-snake algorithm is a feedback controller to track the R-snake and to satisfy non-holonomic and limited torque constraints. The implementation can be very efficient for real-time tracking, as shown in Fig. 4. If a single step is Δt, generating n steps along the A-snake takes a fixed time of $O(n\Delta t)$ and uses only local information. Therefore, it is more suitable to distributed implementation.

6. Conclusions

This paper has proposed a distributed predictive control scheme for the navigation of a robot with a low degree of intelligence, utilizing a wireless visual sensor network in an environment. The scheme is based on a reference snake and a control snake maintained in the wireless network, taking into account robot constraints and a dynamic environment. The reference snake has three states in order to facilitate collaborated planning of a collision-free and constraint-compliant path by distributed sensors. A tracking controller is further developed, which consists of the control snake to correct tracking error and a predictive control mechanism to optimize tracking speed under dynamic and kinematic constraints of a wheeled robot. The proposed path planning and control are suitable for distributed implementation and can deal with the communication latency of wireless sensor networks. Both the simulation and the experiment have been carried out to verify that the proposed method offers an effective distributed solution for integrating robot navigation, path planning, trajectory generation, and motion control into a unified snake-based mechanism. While the current research seldom focuses on low-level motion control using pervasive intelligence in an environment, the results of this paper demonstrate that the distributed sensors are able to provide automatic navigation service; for example, greatly growing CCTV cameras can be used to enhance the mobility of vehicles with less on-board intelligence, such as indoor wheelchairs for aging and disabled people. In the future, research on sampling-based planning for initializing the reference snake will be carried out to improve the efficiency of its initialization and recovering from a broken state, where two issues need to be addressed: (1) communication protocol to support sampling across several wireless sensors and (2) sampling-based planning for a robot with non-holonomic constraints.

Acknowledgements

This research work was partially supported by the Royal Society and the NSFC joint project WiME.

References

1. R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots* (MIT Press, Cambridge, MA, 2004).
2. R. R. Murphy, *Introduction to AI Robotics* (MIT Press, Cambridge, MA, 2000).
3. N. Ratliff, M. Zucker, J. A. Bagnell and S. Srinivasa, "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning," *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan (2009) pp. 489–494.
4. M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning, Robotics and Automation," *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China (2011) pp. 4569–4574.
5. G. N. DeSouze and A. C. Kak, "Vision for mobile robot navigation: A survey," *IEEE Trans. Pattern Anal. Machine Intell*. **24**, 237–267 (Feb 2002).
6. D. Estrin, D. Culler and K. Pister, "Connecting the physical world with pervasive networks," *IEEE Pervasive Comput*. **1**, 59–69 (2002).
7. R. Poovendran, "Cyber-physical systems: Close encounters between two parallel worlds," *Proc. IEEE* **98**, 1363–1366 (2010).
8. M. D. Ilic, L. Xie, U. A. Khan and J. M. F. Moura, "Modeling of future cyber-physical energy systems for distributed sensing and control," *IEEE Trans. Syst. Man Cybern.* A **40**, 825–838 (2010).
9. Y. G. Ha, J. C. Sohn, Y. J. Cho and H. Yoon, "Towards a ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework," *ETRI J*. **27**, 666–676 (2005).
10. A. Saffiotti, M. Broxvall, M. Gritti, K. LeBlanc, R. Lundh, J. Rashid, "The PEIS-ecology project: Vision and results," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France (2008) pp. 2329–2335.
11. E. Guizzo, "Robots with their heads in the clouds," *IEEE Spectr.* **48**, 16–18 (2011).

12. L. Sabri, A. Chibani, Y. Amirat and G. P. Zarri, "Narrative Reasoning for Cognitive Ubiquitous Robots," *Presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, San Francisco, CA (2011).
13. P. Jiang, Z. Feng, Y. Cheng, Y. Ji, J. Zhu, X. Wang, F. Tian, J. Baruch and F. Hu, "A mosaic of eyes," *IEEE Robot. Autom. Mag*. **14**, 104–113 (2011).
14. D. Tacconi, D. Miorandi, L. Carreras, F. Chiti and R. Fantacci, "Using wireless sensor networks to support intelligent transportation systems," *Ad Hoc Netw*. **8**, 462–473 (2010).
15. V. P. Srini, "A vision for supporting autonomous navigation in urban environments," *Computer* **39**, 68–77 (2006).
16. A. Mpitziopoulos, C. Konstantopoulos, D. Gavalas and G. Pantziou, "A pervasive assistive environment for visually impaired people using wireless sensor network infrastructure," *J. Netw. Comput. Appl*. **34**, 194–206 (2011).
17. M. Li, Y. Liu, J. Wang and Z. Yang, "Sensor Network Navigation Without Locations," *Proceedings 28th IEEE International Conference on Computer Communications*, Rio de Janeiro, Brazil (2009) pp. 2419–2427.
18. A. Howard, M. J. Mataric and G. S. Sukhatme, "Relaxation on a Mesh: A Formalism for Generalized Localization," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Wailea, Hawaii (2001).
19. C. M. Huang and L. C. Fu, "Multitarget visual tracking based effective surveillance with cooperation of multiple active cameras," *IEEE Trans. Syst. Man Cybern.* B **41**, 234–247 (Feb 2011).
20. Q. Li and D. Rus, "Navigation protocols in sensor networks," *ACM Trans. Sensor Netw*. **1**, 3–35 (2005).
21. B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert and S. S. Sastry, "Distributed control applications within sensor networks," *Proc. IEEE* **91**, 1235–1246 (2003).
22. A. Meystel (Ed.), *Nested Hierarchical Control: An Introduction to Intelligent and Autonomous Control* (Kluwer, Boston, MA, 1992).
23. P. Jiang, Y. Ji, X. Wang, J. Zhu and Y. Cheng, "Design of a multiple bloom-filter for distributed navigation routing," *IEEE Trans. Syst. Man Cybern. Syst.* Early online access articles (2013).
24. M. Kass, A. Witkin and D. Terzopoulos, "Snake: Active contour models," *Int. J. Comput. Vis*. **1**, 321–331 (1988).
25. S. Quinlan and O. Khatib, "Elastic Bands: Connecting Path Planning and Control," *IEEE International Conference on Robotics and Automation* Atlanta, GA, USA (1993) pp. 802–807.
26. A. McLean and S. Cameron, "The virtual springs method: Path planning and collision avoidance for redundant manipulators," *Int. J. Robot. Res.* **15**, 300–319 (1996).
27. S. Cameron (Ed.), *Dealing with Geometric Complexity in Motion Planning* (Practical Motion Planning in Robotics) (Wiley, New York, 1998) pp. 259–274.
28. A. Mclean and S. Cameron, "Snake-Based Path Planning for Redundant Manipulators," *Robotics and Automation, IEEE International Conference on,* Atlanta, GA, USA (1993) pp. 275–282.
29. L. J. Zhou, C. L. Teo and E. Burdet, "A Nonlinear Elastic Path Controller for a Robotic Wheelchair," *Third IEEE Conference on Industrial Electronics and Applications*, Singapore (2008) pp. 142–147.
30. S. M. LaValle, *Planning Algorithms* (Cambridge University Press, Cambridge, 2006).
31. T. Liang, J. Liu, G. Hung and Y. Chang, "Practical and flexible path planning for car-like mobile robot using maximal-curvature cubic spiral," *Robot. Auton. Syst.* **52**, 312–325 (2005).
32. W. L. Nelson, "Continuous steering-function control of robot carts," *IEEE Trans. Ind. Electron*. **36**, 330–337 (1989).
33. S. S. Ge, X. Lai and A. A. Mamun, "Sensor-based path planning for nonholonomic mobile robots subject to dynamic constraints," *Robot. Auton. Syst*. **55**, 513–526 (2007).
34. S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res*. **30**, 846–894 (2011).
35. J.-P. Laumond, P. E. Jacobs, M. Taix and R. M. Murray, "A motion planner for nonholonomic mobile robots," *IEEE Trans. Robot. Autom.* **10**, 577–593 (1994).
36. Y. Cheng, P. Jiang and Y. F. Hu, "A Distributed Snake Algorithm for Mobile Robots Path Planning with Curvature Constraints," *IEEE International Conference on Systems, Man and Cybernetics*, Singapore (2008) pp. 2056–2062.
37. T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 2nd ed. (MIT Press, Cambridge, MA, McGraw-Hill, New York, 2001).
38. R. Brockett, "Asymptotic stability and feedback stabilization," **In**: *Differential Geometric Control Theory* (R. W. Brockett, R. S. Millman and H. J. Sussmann, eds.) (Birkhauser, Boston, MA, 1983) pp. 181–191.
39. C. C. D. Wit, H. Khennouf, C. Samson and O. J. Sordalen, "Nonlinear control design for mobile robots," *World Sci. Ser. Robot. Autom. Syst.* **11**, 121–156 (1993).
40. J. Godhavn and O. Egeland, "A Lyapunov approach to exponential stabilization of nonholonomic systems in power form," *IEEE Trans. Autom. Control*. **42**, 1028–1032 (1997).
41. A. Aguiar, A. Atassi and A. Pascoal, "Regulation of a Nonholonomic Dynamic Wheeled Mobile Robot with Parametric Modeling Uncertainty using Lyapunov Function," *Proceedings of the 39th IEEE Conference on Decision and Control*, Sidney, Australia (Dec 2000).
42. C. C. De Witt and O. J. Sordalen, "Exponential stabilization of mobile robots with nonholonomic constraints," *IEEE Trans. Autom. Control* **37**, 1791–1797 (1992).

43. J. Hespanha, "Stabilization of Nonholonomic Integrators via Logic-Based Switching," *Proceedings of the 13th World Congress of IFAC*, Francisco, CA, USA (1996) 467–472.
44. K. Koh and H. Cho, "A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints," *J. Intell. Robot. Syst.: Theory Appl*. **24**, 367–385 (1999).
45. C. Wit and C. Samson, "Path Following of a 2-DOF Wheeled Mobile Robot Under Path and Input Torque Constraints," *IEEE International Conference on Robotics and Automation*, Sacramento, CA (1991) pp. 1142–1147.
46. M. Prado, A. Simo, E. Carabias, A. Perez and F. Ezquerro, "Optimal velocity planning of wheeled mobile robots on specific paths in static and dynamic environments," *J. Robot. Syst*. **20**, 737–754 (2003).
47. J. J. E. Slotine and W. Li, *Applied Nonlinear Control* (Prentice-Hall, Englewood Cliffs, NJ, 1991).
48. D. A. Jirenhed, G. Hesslow and T. Ziemke, "Exploring Internal Simulation of Perception in Mobile Robots," *The Fourth European Workshop on Advanced Mobile Robots,* Lund, Sweden (2001) pp. 107–113.
49. C. E. Gacia, D. M. Prett and M. Morari, "Model predictive control: Theory and practice – A survey," *Automatica*, **25**, 335–348 (1989).
50. D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, **36**, 789–814 (2000).
51. G. Klancar and I. Skrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robot. Auton. Syst*. **55**, 460–469 (2007).
52. F. Kuhne, W. F. Lages and J. M. G. da Silva Jr, "Model Predictive Control of a Mobile Robot Using Linearization," *Proceedings of Mechatronics and Robotics*, Aachen, Germany (2004).
53. K. Kanjanawanishkul, M. Hofmeister and A. Zell, "Path Following with an Optimal Forward Velocity for a Mobile Robot," *7th IFAC Symposium on Intelligent Autonomous Vehicles*, Lecce, Italy (2010) pp. 462–467.
54. Y. G. Xi and C. G. Zhang, "Rolling path planning of mobile robot in a kind of dynamic uncertain environment," *Acta Autom. Sin.* **28**, 161–175 (2013).
55. Intel Corporation, "Intel mote 2 engineering platform data sheet," Rev2.0, 2006 (online document, link: http://wsn.cse.wustl.edu/images/c/cb/Imote2-ds-rev2_2.pdf)
56. Omnivision Technologies Inc., "OV7620 single-chip CMOS VGA color digital camera," Rev1.3, 2000 (online product specification, link: http://mxhaard.free.fr/spca50x/Doc/Omnivision/OV7620.pdf)
57. J. H. Chuang and N. Ahuja, "An analytically tractable potential field model of free space and its application in obstacle avoidance," *IEEE Trans. Syst. Man Cybern.* B **28**, 729–736 (1998).