Design and implementation of a navigation system for autonomous mobile robots

Jang-Ping Sheu*

Department of Computer Science,
National Tsing Hua University,
Hsinchu, Taiwan 30013
Fax: +886-3-572-3694
E-mail: sheujp@cs.nthu.edu.tw
*Corresponding author

Chia-Chi Chang and Kai-Wen Lo

Department of Computer Science and Information Engineering,
National Central University,
Chung-Li, Taiwan 32001
E-mail: dyson0703@gmail.com
E-mail: kevinlo@axp1.csie.ncu.edu.tw

Chi-Wen Deng

Networks & Multimedia Institute,
Institute for Information Industry, Taiwan
E-mail: cwdeng@nmi.iii.org.tw

Abstract: In this paper, a navigation system for autonomous mobile robots is proposed. Our navigation system is a hybrid of behaviour-based and model-based navigation systems. In our system, a behaviour-based subsystem is in charge of low-level reactive actions, and a model-based subsystem is responsible for high-level planned actions. If there are obstacles in the way, the navigation system will use our obstacle avoidance algorithm to navigate around these obstacles and keep the robot moving towards the destination. On the basis of our experimental results, our navigation system can navigate the robot to the destination effectively.

Keywords: hybrid architecture; localisation; navigation system; obstacle avoidance; WSNs; wireless sensor networks.

Reference to this paper should be made as follows: Sheu, J-P., Chang, C-C., Lo, K-W. and Deng, C-W. (2010) 'Design and implementation of a navigation system for autonomous mobile robots', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 6, No. 3, pp.129–139.

Biographical notes: Jang-Ping Sheu received the BS in Computer Science from Tamkang University, Taiwan, Republic of China, in 1981, and the MS and PhD in Computer Science from National Tsing Hua University, Taiwan, in 1983 and 1987, respectively. Currently, he is a Chair Professor of the Department of Computer Science, National Tsing Hua University. His current research interests include wireless sensor networks, vehicular ad hoc networks and mobile computing. He was an Associate Editor of *the IEEE Transactions on Parallel and Distributed Systems*. He is an Associate Editor of *International Journal of Ad Hoc and Ubiquitous Computing and International Journal of Sensor Networks*. He is an IEEE Fellow, a member of the ACM and Phi Tau Phi Society.

Chia-Chi Chang received the BS in Mechanical Engineering from Chung Yuan Christian University in 1999, the MS in Mechanical Engineering from National Yunlin University of Science & Technology in 2001. Currently, he is a PhD student in the Department of Computer Science and Information Engineering of National Central University. His current research interests include wireless sensor networks and embedded system design.

Kai-Wen Lo received his BS in Electrical and Control Engineering from Chiao Tung University in 2006, and his MS in Computer Science and Information Engineering from the National Central University, Taiwan, in 2008. His current research interests include wireless sensor networks and mobile computing.

Chi-Wen Deng received his MSEE from National Chung Cheng University, Chiayi County, Taiwan, ROC, in 2003. Currently, he is a Project Manager at the Department of Networks & Multimedia Institute in Taiwan Institute for Information Industry. His areas of interest are wireless sensor networks, intelligent system applications, embedded system and power-line communication protocol.

1 Introduction

Nowadays, robotics has been growing vigorously, and it is the science of robots that covers many subjects, such as electronics, mechanics and software. Most robots are mobile that can be operated in the real-world environment without any form of external control. Hence, they are called "autonomous mobile robots". One of the most fundamental and important issues is to let these robots be able to reach the scheduled position. The movement of robots is controlled by their own navigation system, which is the key to the right movement action. Thus, in this paper, we focus on design and implementation of a navigation system for autonomous mobile robots. We present an effective navigation system that permits robot to arrive in the correct position.

Robot navigation systems can be classified into behaviour-based and model-based systems. Behaviour-based navigation systems (Grush, 2004; Sheu et al., 2008; Simpson et al., 2006) are composed of a layered set of task-achieving modules. It implements a specific robotic behaviour in each module, which can only solve the portion of the required navigation problem. However, a set of modules can cooperate to mimic more complex behaviours. Behaviour-based navigation systems, which neither need precise locations nor miss their own positions, have better flexibility. Besides, they can be implemented quickly in a simple environment. They do not need a map of the environment and therefore they cannot do high-level path planning. In other words, they just simply and directly return action feedback with regard to their sensing environment, so they cannot handle complex situations or environments.

Model-based navigation systems (Luke et al., 2005) consist of four phases, namely perception, localisation, planning and motion control. In the first phase, the navigation systems collect the environmental information such as the location of walls, doors, obstacles and people. In the second phase, they use the current and historical information to estimate their locations in the maps. In the third phase, they plan the path from their own robot location to their own destination. Furthermore, they guide their robots along that path to destination. Thus, the robots are able to accomplish their navigation missions. Because of using map-based concept of position explicitly, model-based navigation systems are suitable for any simple or complex environment with its correct and precise map. Navigation systems can do more highly intelligent path planning to let the robot move more efficiently by utilising the maps. Furthermore, people can also utilise the maps conveniently to command the robot to move. However, model-based navigation systems depend on the internal stored map and historical information of the environment. If there is great difference between the map and the stored information of the environment, the robot will get lost owing to being incapable of estimating its own position. If we got the more precise map and information that sensors get, the model-based navigation systems operate constantly without fail. However, higher costs for implementing the map and sensors required are the disadvantages of the model-based navigation systems.

The goal of this paper is to design and implement a navigation system for an autonomous mobile robot. Our navigation system is capable of navigating the autonomous robot to the target position correctly. It is also a hybrid of behaviour-based and model-based systems and therefore it has both of their advantages. The research of WSNs (Karl and Willing, 2007; Sheu et al., 2008) is becoming popular recently. A WSN is a network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor/sense environmental conditions. There are some papers that discuss the localisation in WSNs (Sheu et al., 2006) and robots in WSNs (Batalin et al., 2004; Terwilliger et al., 2004). In our scheme, the system stores the map of the environment, which can utilise the model-based algorithm to estimate the position of the robot. In addition, we use the localisation algorithm of WSNs (Sheu et al., 2006) to help our system to localise the robot robustly. The path from mobile robot to the destination is divided into many virtual points to let the behaviour-based algorithm be able to use them. Then, we use the behaviour-based algorithm to approach each virtual point. If there are some obstacles in the way, we can use our algorithm to avoid obstacles and keep the robot moving towards the destination. We use multithread technology to implement our navigation system. Hence, important modules of our navigation system can run concurrently and our system can utilise the multi-core processor efficiently. On the basis of our experimental results, we observe that our navigation system can succeed in navigating the robot to the destination, and the robot with our navigation system can patrol a variety of the indoor passages correctly. Thus, these experimental results make it clear that we present a navigation system for an autonomous robot that would be able to navigate the robot to the correct place.

The remainder of this paper is organised as follows. In Section 2, we review the navigation systems of the robot. In Section 3, we present our navigation system and algorithms. The system implementation and experimental results are shown in Section 4. Finally, the paper is concluded in Section 5.

2 Preliminary

Generally, the navigation of the robots can fall simply into three steps:

1 utilising the robot own sensors to collect the environmental information such as the location of walls, doors, obstacles and people

2 using the information to make a strategic decision

3 controlling motors to move in accordance with the decision.

Therefore, robot depends on its own architecture to achieve its navigation; thus, we have to design a navigation system of the robot according to its own mechanism, hardware architecture and software platform. Various robots have different kinds of shapes, mechanisms and architectures, so their navigation systems must be adjusted in accordance with those differences; besides, the components that are the most related to the navigation system are sensors, motion mechanisms, software platforms and Software Development Kits (SDKs).

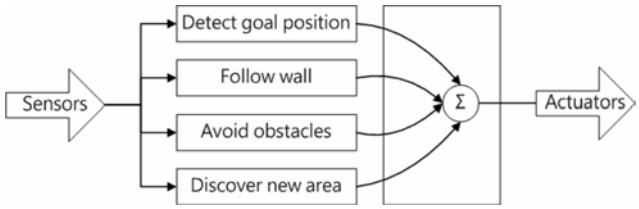
Robots depend on their sensors to obtain the information from the environment, and provide their own navigation systems with this information. There are many common sensors, such as infrared, ultrasound, laser and camera; besides, these sensors have their respective distinguishing characteristics and properties. Software platforms and SDKs of the robots are used to implement the robot navigation system. However, neither software platforms nor SDKs have a product that has a very high market share; therefore, many companies have been investing a lot of resources in developing their products to increase their market share (Kramer and Scheutz, 2007). There are some famous products, such as Microsoft Robotics Studio (MSRS) (Jackson, 2007) and Evolution Robotics Software Platform (ERSP) (Munich et al., 2005).

2.1 Behaviour-based system

In Figure 1, the behaviour-based navigation systems consist of a layered set of task-achieving modules. Each module implements one specific behaviour that achieves or maintains goals, such as wall following (Tarokh and Kuo, 2007), goal detection, obstacle avoidance (Ma and Yang, 2007) and new area discovering. Therefore, each module has to solve only the part of the navigation problem that it requires; in addition, each module can take inputs from the sensors of the robot (e.g., ultrasound, infrared or camera) or from other modules in the system and send outputs to the actuators of the robot (e.g., wheels, arms or legs) or to other behaviours. Thus, a behaviour-based system is a structured network of such interacting modules (i.e., behaviours), and a set of modules can work together in various combinations to display behaviours that mimic more complex actions, such as robot navigation.

Behaviour-based navigation systems are based on a belief that sensors and actuators are noisy and information-limited, so they avoid creating a geometric map. Hence, behaviour-based navigation systems avoid explicit reasoning about localisation and position; instead, it designs sets of behaviours that work together to achieve the desired robot motion. Accordingly, behaviour-based navigation systems have better flexibility because they do not need precise localisations and they would not have the problems of missing their own positions, therefore they can handle more uncertain factors. Furthermore, behaviour-based navigation systems can be implemented quickly in a simple environment, but they may not handle complex situation or environment. The reason for the above is that it is too hard to design sets of behaviours in a complex situation and it may cost the robot too much time to accomplish its navigation in the complex environment.

Figure 1 Architecture for behaviour-based navigation system

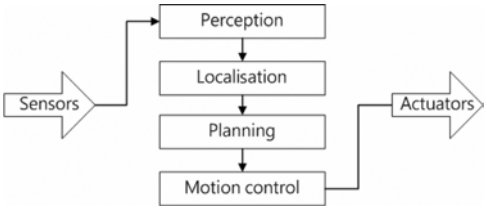


2.2 Model-based system

In contrast to the behaviour-based approach, the model-based (also called map-based) approach includes both localisation and planning modules; besides, it is organised in a hierarchical fashion. In Figure 2, the model-based navigation systems consist of four phases, namely perception, localisation, planning and motion control. First, the navigation system uses the sensors of the robot to take inputs, for example, using ultrasound to get range data and using camera to get an image data, then, system uses algorithms to transform data to information that indicates the environment around the robot, such as the location of walls, doors, obstacles and people (perception). Second, because system owns the current and historical information with the environmental map, it can utilise the localisation algorithms (e.g., Odometry (Go et al., 2006), Markov localisation (Fox et al., 1999) or Kalman filter localisation (Roumeliotis and Bekey, 2000)) to estimate their positions in the map (localisation). Third, for an efficient movement, the navigation systems utilise the path-planning algorithms (Karl and Willing, 2007) to plan the path from the location of the robot to the destination. Finally, it sends outputs to the actuators of the robot to guide the robot along that path. Thus, the robot is going to reach the destination and accomplish the navigation in the end of motion control. In addition, there is an extended technique called Simultaneous Localisation and Mapping (SLAM) (Dissanayake et al., 2001). It is used by an autonomous robot/vehicle to start in an unknown location in an unknown environment, and then to incrementally build up the map of this unknown environment while simultaneously using this map to compute the current position of the robot/vehicle. However, SLAM of the mobile robots generally refers to the process of creating

geometrically accurate maps of the environment. So, the sensors of the robot should be accurate enough to create a precise map such as laser. However, it is costly to build a robot with SLAM.

Figure 2 Architecture for model-based navigation system



Model-based navigation systems use map-based concept of position explicitly. They are suitable for any environment no matter how complex it is, as long as they have the correct and precise maps of the environments. Therefore, when we want to deploy model-based robot to a new environment, we only have to simply give the robot a new map; then, the robot will operate correctly in the new environment as usual. In addition, navigation systems can utilise the maps to do more highly intelligent path planning, and can let robot move to the target position more efficiently. Furthermore, the map can represent a medium for communication between human and robot; it can show a lot of information about the robot and environment with using maps; it lets people be able to understand the state of the robot more clearly and to command the robot more conveniently.

We see that model-based navigation systems are based on the internal stored map and historical information of the environment. However, for a robot, to be able to use an internal representation of the spatial layout of its environment to position is a very complex task. First, sensors are the fundamental robot input for the process of perception, but there are sensor noise problems. Second, the map usually only records the permanent objects (e.g., walls, doorways) and movable static objects (e.g., boxes, chairs or doors), but there are many dynamic objects (e.g., people, dogs, cats or other robots) in the real world. Thus, if there are too many dynamic objects or static objects moved to other positions different from original map, it will cause great difference between the map and the information of the real environment, and robot will get lost because of being incapable of estimating its own position; then, whole navigation system will be invalid, and robot will not be able to accomplish its navigation mission. Therefore, using good sensors and good geometric maps that can indicate the real environment correctly and precisely is important for the model-based navigation. Certainly, it costs higher. In addition, there are still some navigation systems, which are designed to combine behaviour-based and model-based navigation systems (Na and Oh, 2003; Qureshi et al., 2004).

To sum up, both behaviour-based and model-based navigation systems have their distinguishing advantages and disadvantages, and we want our navigation system to have both of their advantages. Besides, we expect that our system can be implemented rapidly and cost less. Thus, we design our navigation system to be a hybrid of behaviour-based and model-based to make it happen. Furthermore, to conquer localisation problem from model-based systems, we use the localisation technology of WSNs to help us. Thus, our system is simple, but effective. It will be not only applied extensively but also robust.

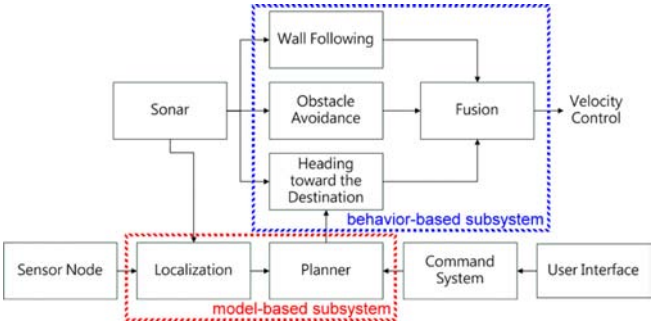
3 System architecture and algorithms

In this section, we present our system including the architecture and algorithms. The purpose of our navigation system is to guide the robot to the desired destination, and it can avoid those obstacles on its navigation way and reach the destination safely and correctly. There are a lot of issues of navigation. The real environment has a lot of kinds of objects such as doors, walls or tables; it is not easy to make a map record of all objects precisely. There are too many uncertain factors in the real world, and too many uncertain factors cause that the navigation would become very hard, and the navigation system is easy to be invalid. The localisation of a robot is a big problem because a robot may have limited sensors (often has its own noise problem) and the stored map may not be precise enough, therefore it is difficult to localise the robot exactly. In addition, the system must be fast enough to react to an emergent accident, and so on. Thus, we need both of the advantages of behaviour-based and model-based navigation systems to overcome those navigation problems.

The architecture of our system is shown in Figure 3. Our system consists of many modules, such as wall following, obstacle avoidance, heading towards the destination, localisation and planner. Among these modules, wall following, obstacle avoidance, heading towards the destination and fusion form the behaviour-based subsystem, and localisation and planner form the model-based subsystem. In our scheme, the behaviour-based subsystem is in charge of low-level reactive actions and the model-based subsystem is in charge of high-level planned actions. In other words, behaviour-based subsystem is responsible for the execution and for reacting to unforeseen situations, and model-based subsystem models the environment and plans actions. Our system use model-based algorithms to localise our robot with the map of the environment and to plan an efficient path to the destination. Then, it divides the path into several virtual points as sections of the path for behaviour-based algorithms, because behaviour-based algorithms are suitable for short distance moving; accordingly, our system can utilise the behaviour-based algorithms to approach each virtual point, and finally reach the destination; in this way, we can conquer the uncertain factors and handle emergent tasks (e.g., obstacle avoidance) more effectively.

After introducing the whole navigation system, the details of modules of our system are given here. The responsibility of User Interface is simple. User Interface is designed to communicate with users and to receive commands from users. Command System is responsible for transforming user command into a series of system commands that drive the navigation system.

Figure 3 The architecture of our navigation system (see online version for colours)



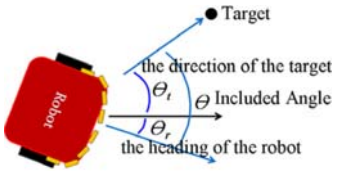
3.1 Behaviour-based algorithms

Our behaviour-based subsystem is composed of wall following, obstacle avoidance, heading towards the destination and fusion. First, we introduce heading towards the destination. The purpose of heading towards the destination is to let the robot rotate and head to the target position. In Figure 4, the first step of it is to use the positions of the robot and target to compute the direction of the target (θ_t) relative to the robot by using equation (1).

$$\theta_t = \tan^{-1}\frac{\Delta y}{\Delta x} \tag{1}$$

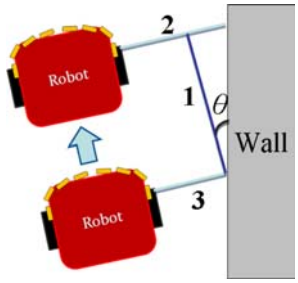
where θ_t is the direction of the target, Δ_x and Δ_y are the difference between the x coordinate and y coordinate of the target and robot, respectively. After we get the direction of the target (θ_t), we can utilise it and the heading of the robot (θ_r) got from the system to compute the included angle (θ) of the direction of the target and the heading of the robot. Accordingly, the robot rotates by the degree of this angle, and then goes straight towards the target.

Figure 4 An included angle between the target and the heading of the robot (see online version for colours)



Second, we describe the module of wall following. Wall following is a typical behaviour of navigation. If the environment is only partially known with lacking position information, the robot can use a wall following strategy to fulfil some navigation missions quickly without having to learn an unknown environment. Certainly, if the environment is known clearly, the robot may not need to sense along a wall and could move along a planned path. In this case, wall following can still help the robot to avoid unforeseen obstacles and to move more smoothly. In our scheme, if a robot has a short distance moving as shown in Figure 5, we utilise the displacement of the robot from a short time before to current moment (as the length of line 1), the current distance sensed by ultrasonic sensor to the wall (as the length of line 2) and the previous distance sensed by ultrasonic sensor to the wall (as the length of line 3) to estimate the included angle (θ) between the heading of the robot and the wall.

Figure 5 Computing the included angle of the heading of the robot and the wall (see online version for colours)



There is the problem of sensor noise that induces a limitation on the consistency of sensor readings in the same environmental state. For reducing the effect of sensor noise, we do not directly use the distance sensed by ultrasonic sensor to the wall (i.e., the present ultrasonic sensor reading), but use equation (2) to estimate the current distance to the wall. In equation (2), we use the last two estimative distances to adjust the current distance sensed by ultrasonic sensor to the wall that can reduce the effect of sensor reading error owing to sensor noise.

$$d_t = d_{\text{now}} \times \alpha + d_{t-1} \times \beta + d_{t-2} \times \gamma \tag{2}$$

where d_t is the estimative distance to the wall on the current time, d_{t-1} is the estimative distance to the wall at last time, d_{t-2} is the estimative distance to the wall at the time before last time, d_{now} is the present distance that sensor is reading, α, β, γ are parameters and $\alpha + \beta + \gamma = 1$ (We set $\alpha = 0.6$, $\beta = 0.3$ and $\gamma = 0.1$ in our experiments). After estimating the included angle between the robot and the wall, the robot knows how to rotate to follow the wall in accordance with this included angle (i.e., try to let the included angle be zero).

Third, we present the algorithm used in obstacle avoidance module. Our obstacle avoidance algorithm is an extended Bug algorithm (Broadhurst et al., 2005). The Bug algorithm is a simple but effective obstacle avoidance algorithm. The behaviours of the Bug algorithm are:

1 following the contour of each obstacle

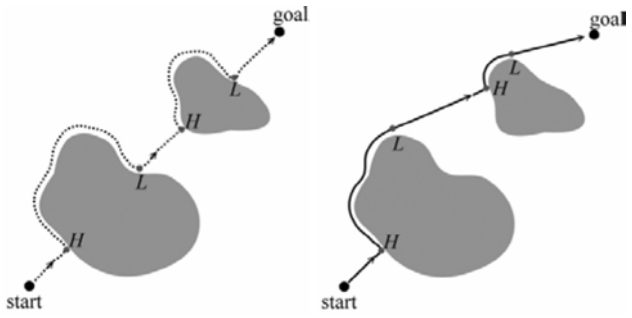
2 moving in a straight line towards goal.

The typical Bug algorithms are Bug 1 and Bug 2; using Bug 1, the robot fully circles the obstacle, and then departs from the point with the shortest distance moving towards the goal, of course, this scheme is inefficient. With Bug 2, the robot heads towards the goal on the line that starts from the location of the robot to the goal. If an obstacle is in the way, the robot follows the contour of the obstacle until the robot encounters the above line again. In general, Bug 2 algorithm will have significantly shorter travel than Bug 1.

We modify the typical Bug algorithms to be our obstacle avoidance algorithm. In our scheme, if an obstacle is in the way, first, we use ultrasonic sensors to find which side of

the robot is nearest obstacles; if the right (left) side of the robot is nearest obstacles, the robot turns left (right) and then using right (left) wall (obstacle) following follows the contour of the obstacle. The robot departs immediately when its heading is equal to the direction of the goal and there is no obstacle in the way, and then it moves directly towards the goal. In general, our algorithm will have shorter total robot travel distance than Bug 2 as shown in Figure 6.

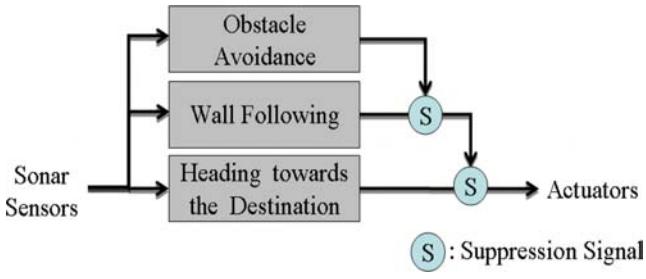
Figure 6 Examples of obstacle avoidance with *H*, hit point, and *L*, leave point (a) Bug 2 algorithm and (b) our obstacle avoidance algorithm



(a) (b)

Finally, we present the design of the fusion module. The purpose of the fusion is to combine behaviour modules (wall following, obstacle avoidance, heading towards the destination) to display the short-distance navigation. In Figure 7, the structure of the fusion is layered, and each layer is a behaviour module. Besides, layers of the fusion have different levels; a higher layer has a higher level. When one layer (module) has to be active, the layer can send a 'suppressive signal' to suppress other lower-level layers. Therefore, if there are several behaviours operating at the same time, only a behaviour module can drive the actuators.

Figure 7 The layer structure of the fusion (see online version for colours)



In our design, we assign the highest level to the layer (module) of obstacle avoidance, because avoiding obstacles has top priority to prevent the robot from being damaged. The layer (module) of wall following is assigned the normal level; when the robot is near the wall and wall following benefits approaching the target, the robot will follow the wall to approach the target. Finally, we assign the lowest level to the layer (module) of heading towards the destination; the fundamental purpose of the navigation is to let the robot reach the destination, so the robot should be heading towards the destination all the time in an ordinary situation.

The behaviour-based subsystem that consists of wall following, obstacle avoidance, heading towards the destination and fusion can achieve the short-distance navigation. The operating process of the behaviour-based subsystem is as follows. At the beginning of the process, the behaviour-based subsystem receives the information of the target including its position. Then, it uses the module of heading towards the destination to approach the target constantly in an ordinary situation. If there is a wall near the robot, the subsystem let the robot follow the wall by using the module of wall following as long as the robot following this wall profits approaching the target. When an obstacle is in the way, the subsystem utilises the module of obstacle avoidance to avoid it. After repeating these steps, the subsystem will guide the robot to the target, and will have accomplished its short-distance navigation task in the end.

3.2 Model-based algorithms

Model-based subsystem is composed of localisation and planner. First, we introduce localisation. The purpose of localisation is to estimate the robot position. Our module of the localisation consists of three techniques. The first technique is odometry that is the most widely used method for estimating the position of a mobile robot. Besides, relative positioning is usually based on odometry that is monitoring the wheel revolutions to compute the offset from a known starting position. Odometry is simple, inexpensive, and easy to accomplish in real time. But, the drawback of odometry is unbounded accumulation of errors.

Furthermore, when the robot is operating, the robotic coordinates system may be different with the global coordinates system as shown in Figure 8. So, we have to transform the position in the robotic coordinates system into the position in the global coordinates system. By using equation (3).

$$p_G = \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \times p_R \tag{3}$$

where p_R is the position of the robot in its own robotic coordinates system, and p_G is the position of the robot in the global coordinates system. Both p_R and p_G are vectors with x coordinate and y coordinate, and θ is the angular difference between the global and the robotic coordinates system. In Figure 8, by applying $\theta = 90°$ to equation (3), we can get that the x coordinate of the global coordinates system (X_G) is equal to the negative of the y coordinate of the robotic coordinates system ($-Y_R$), and that the y coordinate of the global coordinates system (Y_G) is equal to the x coordinate of the robotic coordinates system (X_R).

Accordingly, we design a simple and effective localisation algorithm for our system to reduce accumulative errors of odometry. The main idea of this localisation is shown in Figure 9. First, the robot uses wall following algorithm to follow the wall, the state of the robot (i.e., the position and the heading of the robot) in reality

would be Figure 9(a). Besides, because of accumulative errors of odometry, the internal state of the robot estimated by odometry with the stored map would be inaccuracy like Figure 9(b). Therefore, we have to calibrate the internal state of the robot close to the state of the robot in reality (i.e., reduce errors of odometry). We use the ultrasonic sensor readings to calculate the heading of the robot in reality, and then use it to calibrate the internal heading of the robot as shown in Figure 9(c). Afterward, using the distance sensed by ultrasonic sensor to the wall calibrates the internal position of the robot as shown in Figure 9(d). Thus, we have accomplished the calibration of the internal position and heading of robot estimated by odometry with the stored map (localisation), and have reduced the errors of odometry to let the robot have a more accuracy estimated state. Furthermore, when the robot follows a wall every time, the robot will check the difference between the position and heading of robot calculated by using ultrasonic sensor readings and the internal position and heading of the robot estimated by the odometry with stored map. If the difference is bigger than a threshold (the threshold of the positional difference is 66 cm and the threshold of the heading difference is 5° in our experiments), we calibrate the internal position and heading of the robot with this algorithm.

However, even we use the above-mentioned localisation techniques; the robot may be still lost sometimes, because there is the accumulation of odometry errors, and not every environment can use our localisation algorithm. Therefore, we use the localisation technology of the WSNs to help us localise our robot more robustly. The location algorithm of WSN we used is based on Received Signal Strength Indicator (RSSI) values that will decrease when the distance increases. Reference nodes are static nodes placed at known positions. A Blind node is a node that will collect signals from all reference nodes responding to a request. Then, it reads out the respective RSSI values and feeds the collected values into the hardware engine. Afterward, it reads out the calculated position and sends the position information to our navigation system. Thus, we deploy the reference nodes in some complex environment such as around intersection, corners, or some environments without walls, and put the blind node on our robot to let it operate with our navigation system. In this way, we could utilise the localisation of the WSNs to help us localise our robot and check the estimated position in our system.

However, RSSI that can be affected by multi-path and noise is unstable. Therefore, the localisation of the WSNs based on RSSI sometimes would be unstable, and we would get a big positional difference between the current and previous calculated positions. If we detect that the difference between the current and previous calculated positions is bigger than a threshold, we use the last calculated position plus the last displacement of the robot to be the new calculated position instead. Since the average speed of our robot is 0.5 m/s, and we receive an estimated location of the robot from sensor node per second, we set the threshold as 2 m in our experiments.

Figure 8 The robotic coordinates system and the global coordinates system (see online version for colours)

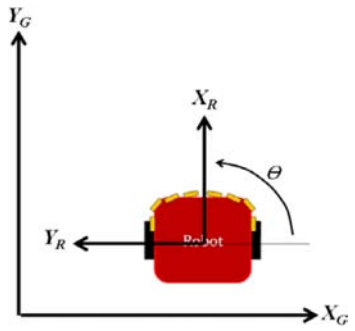
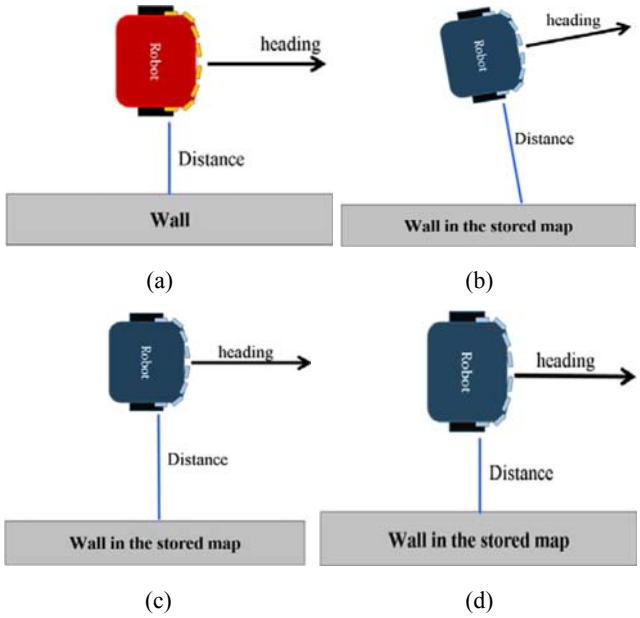


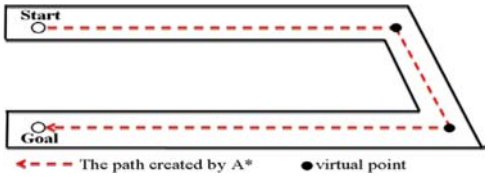
Figure 9 The calibration of the position and heading of the robot: (a) the state of the robot in reality; (b) the internal state of the robot; (c) calibrate the heading of the robot and (d) calibrate the position of the robot (see online version for colours)



Finally, we present the module of the planner. The main purpose of planner is that it plans an efficient path from the location of the robot to the destination, and divides that path into many segments for behaviour-based subsystem. In Figure 10, we use the A* algorithm (Dalmau, 2003) to be our path-planning algorithm. A* is a well-known and effective algorithm, and is a best-first, graph search algorithm that finds the least-cost path from a given initial node/position to the goal node/position. After A* have planned the path, the path would be divided into many segments, because the behaviour-based subsystem is capable of accomplishing a short-distance navigation task very well, but it is not good at long-distance navigation task or complex situation (e.g., the robot should turn many times in the navigation task). Therefore, we assign many virtual points to the path, and these virtual points divide the path into segments. The method of assigning virtual points is that every short distance (we used 10 m in our experiments), we assign a virtual point to the straight-line region of the path, and we assign a virtual point to each turning point

of the path. Afterward, the behaviour-based subsystem can approach each virtual point and reach the destination finally.

Figure 10 A path from start to goal with virtual points (see online version for colours)



To sum up, the process of our navigation system is as follows. First, the system gets the destination required from the user; it uses the model-based algorithm to plan the path to the destination with the stored map and gets many virtual points that divide this path into segments. Then, it uses the behaviour-based algorithm to approach each virtual point. During the period of the approaching, each virtual point, if there is an obstacle in the way, can use the behaviour-based algorithm to avoid it; besides, it uses the model-based algorithm with WSN to localise the robot continuously. Finally, the robot reaches the destination without fail, and our navigation system has accomplished a successful navigation mission.

4 System implementation and experiments

In this section, we present how to implement our navigation system and show the results of our experiments. We chose Pioneer 3-DX as our robot platform. The P3-DX is a two-wheel-drive robot. It is $44 \times 38 \times 22$ cm^3 aluminium body and its weight is 9 kg. The P3-DX can move at speeds of 1.6 m/s on the flat floor. In addition, the P3-DX base includes eight ultrasonic sensors that read ranges from 10 cm to 5 m basically. The P3-DX can contain up to three 12 volts direct-current batteries for long missions. Besides, it includes a 32-bit RISC-based controller; the robotic software running on a computer can be connected with its microcontroller via the host serial link to provide the high-level intelligent robot controls.

Furthermore, we use a notebook computer for running our robotic software. We put this notebook computer on our robot (P3-DX), and connect it with the robot for providing intelligent controls. In addition, we deployed a WSN using CC2431 (Texas Instruments Inc., 2007) nodes 0 in the portion of the experimental environment, and put a CC2431 node communicating with the notebook computer on the robot. The CC2431 is a true system-on-chip for wireless sensor networking ZigBee/IEEE 802.15.4 solutions. The chip includes a location engine that can be used in nodes with unknown location (i.e., blind node) to receive signals from nodes knowing their locations (i.e., reference nodes). On the basis of the location engine, it can calculate an estimate of position of the blind node.

The operating system we used on our notebook computer is Microsoft Windows XP Service Pack 2. We chose the programming language C++/CLI and Microsoft Visual Studio 2008 to develop our navigation system in the .NET 2.0 platform. For controlling our robot, we utilised Advanced Robotics Interface for Applications (ARIA) to achieve it. The ARIA written in C++ language is an object-oriented robot control application, programming interface for P3-DX, and is a programming library for programmers to access/control their robots.

Moreover, we used multithread technology to establish our navigation system. A multithreaded application allows us to run several threads, and each thread is running in its own process. Therefore, we are able to run important subprogrammes (modules) of our navigation concurrently. Obviously, it is the advantage of our navigation system. When our robot is heading towards the destination, our robot can detect constantly if there are obstacles in its way concurrently. Hence, if there are some obstacles in the way, our robot can react to those obstacles more rapidly and would prevent itself from the collision more effectively. In addition, there are more and more multi-core computers in the future and we see that the benefit of having multi-core processor is that the system can handle more than one thread. Our navigation system designed with multithread can utilise the multi-core processor more efficiently than any single-threaded robotic programme.

The following is the results of our experiments. In our experiments, the moving speed of the robot is set to 0.5 m/s. In the first experiment, we wanted to test if our navigation system is able to guide the robot to the destination correctly in an ordinary environment. We chose an indoor passage of the building as our navigation environment. The rough environmental map of the first experiment is shown in Figure 11. The robot with our navigation system was going to patrol this section of the passage. The result of the first experiment is shown in Figure 12.

Figure 11 The rough environmental map of the first experiment (see online version for colours)

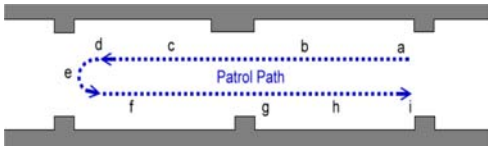


Figure 12 The robot with our navigation system is patrolling the passage (see online version for colours)

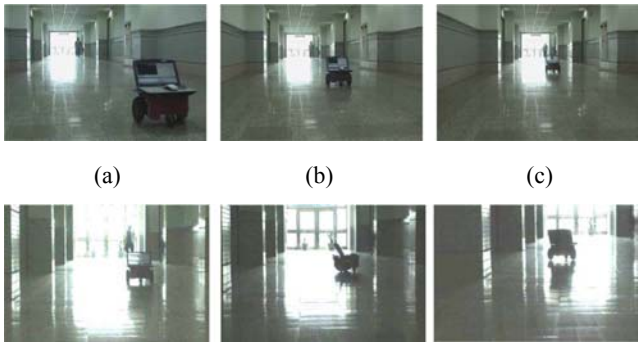


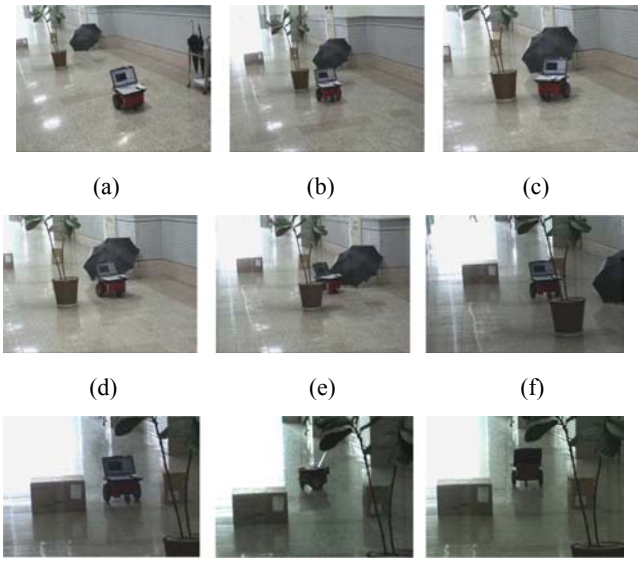
Figure 12 The robot with our navigation system is patrolling the passage (see online version for colours) (continued)

(g) (h) (i)

The position of the robot in Figure 12(a)–(i) is equal to the position (a)–(i) in Figure 11. In Figure 12(a)–(d), the robot is moving forward in a straight line with wall following. Figure 12(d)–(f) shows the turnaround of the robot. We can see that the robot is following wall to move in a straight line again in Figure 12(f)–(i). On the basis of the result, we can see that our robot can move in a straight line by using wall following and the included angle of the heading of the robot and the wall is smaller than 3°; while our robot is following the wall, it keeps a distance to the wall that is bigger than 90 cm for safe; our navigation system can operate correctly and control the robot to achieve a successful patrol. Furthermore, the length of this patrol is 26 m. The total time of the patrol is 54 s, and the average speed is 0.4815 m/s. It is clear that the average speed is close to the speed we set and the movement of our robot is very fluent.

In the second experiment, we wanted to challenge the ability of our navigation system. For this reason, we added an opened umbrella and a big box in the environment of the above-mentioned experiment to build a much more complex environment. Our robot should go through these obstacles and accomplish its patrol mission to prove that it can operate very well in a much more complex environment. The experimental result of our robot patrolling this environment is shown in Figure 13.

Figure 13 Our robot is patrolling the passage with a bonsai, an open umbrella and two boxes in its way (see online version for colours)

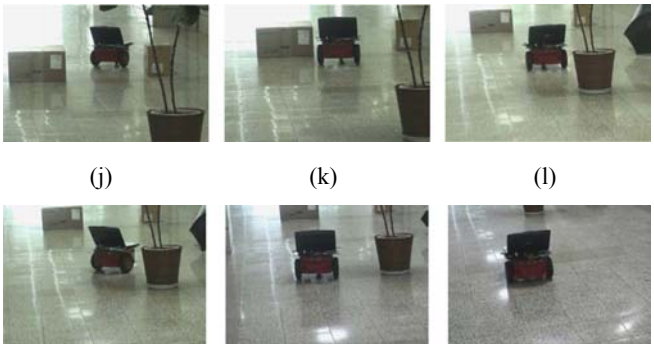


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 13 Our robot is patrolling the passage with a bonsai, an open umbrella and two boxes in its way (see online version for colours) (continued)



(j) (k) (l)

(m) (n) (o)

In Figure 13(a)–(e), our robot finds the obstacles in its way; then, the robot follows continuously the contours of the bonsai and the opened umbrella to avoid them in an impressive manner. Figure 13(f)–(k) shows that our robot goes through between the two boxes efficiently twice. Finally, our robot avoids the bonsai again and returns to the start as shown in Figure 13(l)–(o). According to this experimental result, we see that our navigation system is also suitable for a much more complex environment, and that our navigation system can achieve its patrol task effectively as usual in the complex environment. Furthermore, the length of this patrol is 26 m. The total time of the patrol is 78 s, and the average speed is 0.3333 m/s. The average speed of the robot in this experiment decreases more than in the second experiment because of more obstacles in the way.

In the last experiment, we chose a different form of environment for our test as shown in Figure 14. We would test if our system can patrol various forms of the surroundings. The result of our robot patrolling this area is shown in Figure 15.

Figure 14 The rough environmental map of the fourth experiment (see online version for colours)

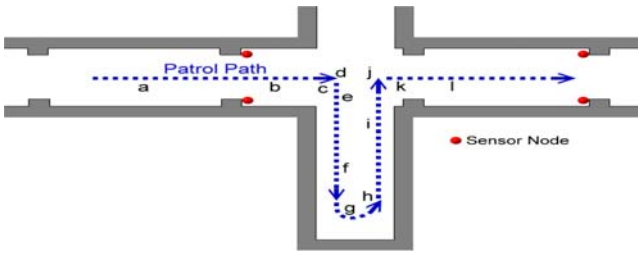
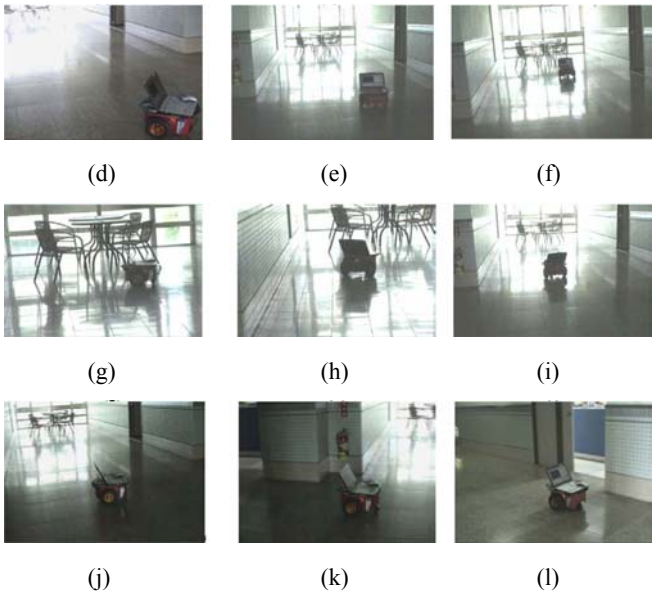


Figure 15 The robot with our navigation system is patrolling the passage (see online version for colours)

(a) (b) (c)

Figure 15 The robot with our navigation system is patrolling the passage (see online version for colours) (continued)



(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

The position of the robot in Figure 15(a)–(l) is equal to the position (a)–(l) in Figure 14. In Figure 16(a)–(c), the robot is moving in a straight line with wall following as earlier. The robot turns right into another section of the passage for patrolling in Figure 15(d)–(e). To mention in passing, it would increase accumulative errors of odometry after the robot turned many times. Hence, the estimated position of the robot would become inaccurate. However, we could utilise our localisation algorithm and WSN to help us to calibrate the estimated position of the robot. So, our robot could have a reliable estimated position of itself. Therefore, after a sequence of actions in Figure 15(a)–(h), the estimated position of our robot has become a little inaccurate. Then, our system detects this inaccuracy and localises the robot for decreasing the inaccuracy of the estimated position of our robot, as shown in Figure 15(i). Afterward, robot can continue patrolling the area to accomplish its patrol mission, referring to Figure 15(i)–(l). On the basis of this experiment result, it is clear that our system can handle various forms of the indoor environments. Furthermore, the length of this patrol is 33 m. The total time of the patrol is 74 s, and the average speed is 0.4459 m/s. The patrol path in this experiment is more complex than in the first experiment so the average speed in this experiment is slower than in the first experiment. However, there are no obstacles in the way; hence, the average speed in this experiment is faster than in the second experiments.

5 Conclusion

In this paper, a navigation system for autonomous mobile robots has been developed, which achieves safe and robust navigation in an arbitrary indoor environment. To be specific, we combine behaviour-based and model-based navigation systems to form our navigation system. We utilise the behaviour-based subsystem to achieve low-level reactive actions such as wall following or obstacle avoidance; the model-based subsystem is in charge of high-level planned actions such as planning or localisation. Furthermore, our system can communicate with WSN and use the localisation technology of WSN to calibrate the estimated position of our robot. Hence, our robot can localise itself more robustly. On the basis of our experimental results, the average speed of our robot closes to the speed we set, and the movement of our robot is fluent. Besides, the robot with our navigation system can use wall following to move in a straight line and can avoid obstacles in its way successfully. In conclusion, the robot with our navigation system can still patrol the passages correctly and efficiently even if there are many unforeseen obstacles in the way.

References

Batalin, M.A., Sukhatme, G.S. and Hattig, M. (2004) 'Mobile robot navigation using a sensor network', *Proceedings of IEEE International Conference on Robotics and Automation*, Los Angeles, USA, Vol. 1, May, pp.636–641.

Broadhurst, A., Baker, S. and Kanade, T. (2005) 'Monte Carlo road safety reasoning', *Proceedings of IEEE Intelligent Vehicles Symposium*, Pittsburgh, USA, June, pp.319–324.

Dalmau, D.S-C. (2003) *Core Techniques and Algorithms in Game Programming*, Published by New Riders, USA.

Dissanayake, M.W.M.G., Newman, P., Clark, S., Durrant-Whyte, H.F. and Csorba, M. (2001) 'A solution to the Simultaneous Localization And Map (SLAM) building problem', *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 3, June, pp.229–241.

Fox, D., Burgard, W. and Thrun, S. (1999) 'Markov localization for mobile robots in dynamic environments', *Journal of Artificial Intelligence Research*, pp.391–427.

Go, Y., Yin, X. and Bowling, A. (2006) 'Navigability of multi-legged robots', *IEEE/ASME Transactions on Mechatronics*, Vol. 11, No. 1, February, pp.1–8.

Grush, R. (2004) 'The emulation theory of representation: motor control, imagery, and perception', *Journal of Behavioral and Brain Sciences*, Vol. 27, June, pp.377–396.

Jackson, J. (2007) 'Microsoft robotics studio: a technical introduction', *IEEE Robotics and Automation Magazine*, Vol. 14, No. 4, December, pp.82–87.

Karl, H. and Willing, A. (2007) *Protocols and Architecture for Wireless Sensor Networks*, Published by John Wiley & Sons, England.

Kramer, J. and Scheutz, M. (2007) 'Development environments for autonomous mobile robots: a survey', *Journal of Autonomous Robots*, Vol. 22, No. 2, February, pp.101–132.

Luke, R.H., Keller, J.M., Skubic, M. and Senger, S. (2005) 'Acquiring and maintaining abstract landmark chunks for cognitive robot navigation', *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Columbia, USA, August, pp.2566–2571.

Ma, M. and Yang, Y. (2007) 'Adaptive triangular deployment algorithm for unattended mobile sensor networks', *IEEE Transactions on Computers*, Vol. 56, No. 7, July, pp.946–958.

Munich, M.E. Ostrowski, J. and Pirjanian, P. (2005) 'ERSP: a software platform and architecture for the service robotics industry', *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pasadena, USA, August, pp.460–467.

Na, Y-K. and Oh, S-Y. (2003) 'Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification', *Autonomous Robots*, Vol. 15, No. 2, September, pp.193–206.

Qureshi, F., Terzopoulos, D. and Gillett, R. (2004) 'The cognitive controller: a hybrid, deliberative/reactive control architecture for autonomous robots', *Proceedings of the 17th International Conference on Innovations in Applied Artificial Intelligence*, May, Springer-Verlag, pp.1102–1111.

Roumeliotis, S.I. and Bekey, G.A. (2000) 'Bayesian estimation and Kalman filtering: a unified framework for mobile robot localization', *Proceedings of ICRA'00 IEEE International Conference on Robotics and Automation*, April, Los Angeles, USA, Vol. 3, pp.2985–2992.

Sheu, J-P., Chen, P-C. and Hsu, C-S. (2008) 'A distributed localization scheme for wireless sensor networks with improved grid-scan and vector-based refinement', *IEEE Transaction on Mobile Computing*, Vol. 7, No. 9, September, pp.1110–1123.

Sheu, J-P., Hsieh, K-Y. and Cheng, P-W. (2008) 'Design and implementation of mobile robot for nodes replacement in wireless sensor networks', *Journal of Information Science and Engineering*, Vol. 24, February, pp.393–410.

Sheu, J-P., Li, J-M. and Hsu, C-S. (2006) 'A distributed location estimating algorithm for wireless sensor networks', *Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, June, Taichung, Taiwan, Vol. 1, pp.218–225.

Simpson, J., Jacobsen, C.L. and Jadud, M.C. (2006) 'Mobile robot control: the subsumption architecture and Occam-Pi', *Proceedings of the 29th WoTUG Technical Meeting of Communicating Process Architectures*, September, Amsterdam, The Netherlands, pp.225–236.

Tarokh, M. and Kuo, J. (2007) 'Vision based person tracking and following in unstructured environments', in Billingsley, J. and Bradbeer, R. (Eds.): *Mechatronics and Machine Vision in Practice*, Springer Berlin Heidelberg, December, pp.99–109.

Terwilliger, M., Gupta, A., Bhuse, V., Kamal, Z.H. and Salahuddin, M.A. (2004) 'A localization system using wireless network sensors: a comparison of two techniques', *Proceedings of the First Workshop on Positioning, Navigation and Communication*, March, Hannover, Germany, pp.95–100.

Texas Instruments Inc. (2007) *CC2431DK Development Kit User Manual Rev. 1.5*, June, pp.1–33.