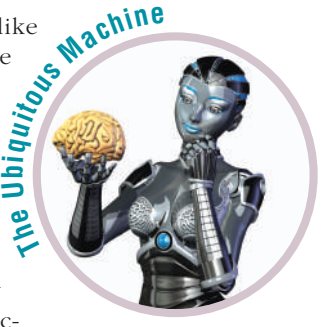
The development of a mobile medical robot using ER1 technology

MOHAMMAD SAUD KHAN



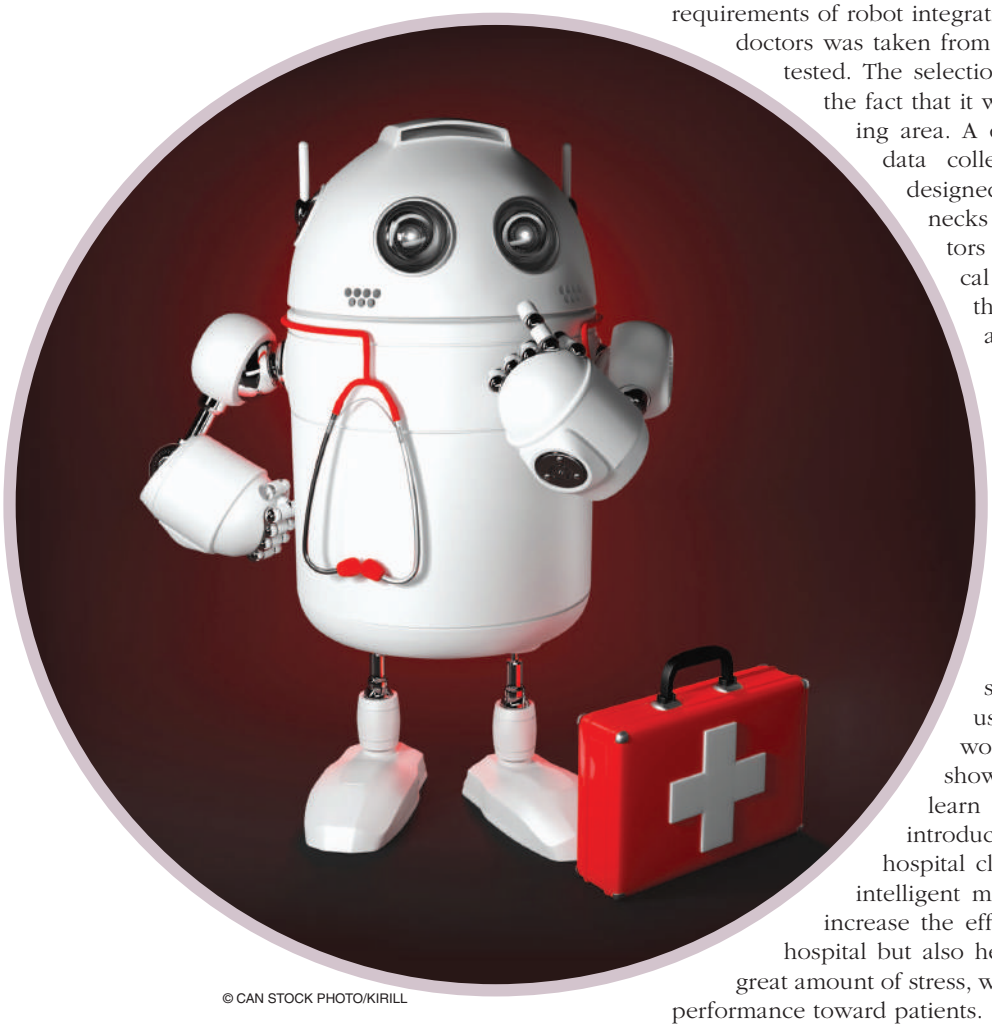
© ISTOCKPHOTO/LINDA BUCKLIN

Modern-day robots do not necessarily look like humans, but they are programmed to handle tasks that are normally carried out by humans, especially in big factories that manufacture products like cars. Employers prefer these mechanical devices because of many reasons: they are faster and more accurate than human workers, they never ask for a pay hike, nor do they take endless coffee breaks. Robots are also capable of working in an environment that is dangerous for humans. They can be found in many places—sorting out bolts in factories, digging a mine shaft, collecting soil samples on the planet Mars, and photographing the ocean floor. Now robots are also being explored within the hospital environment—not as doctors but as couriers. Usually, large hospitals have attendants scurrying around to deliver medicines and bottles to different wards. Mobile robots are ideal for the job of hospital attendants. These robotic couriers could help hospitals in becoming more efficient in the long run.

The goal of the following project was to develop the technology for an intelligent, autonomous mobile robot that can find its way around a hospital by sensing and avoiding obstacles and taking alternative routes if a path is blocked.

Need analysis survey

At the start of the project, it was decided to carry out a need analysis survey from the hospital staff, which included the doctors and nurses, to find out the requirements of robot integration in a hospital. A sample of 20 doctors was taken from a hospital where the robot was tested. The selection of the hospital was based on the fact that it was in close vicinity to the working area. A questionnaire was circulated for data collection. The questionnaire was designed to find out the possible bottlenecks (and problems) if any, the doctors faced with respect to the logistical part of their work. In addition, the use of information technology and a mobile robot as a possible solution to such problems was also inquired. Seventy percent of the respondents felt stressed out at their job, stating the main reason as having to leave their work to carry medical samples or documentation within the hospital. Half of the population had problems in communication with patients and accessing records. More than 80% of the staff (doctors and nurses) already used computers for their daily work and 85% of the respondents showed a great deal of willingness to learn the basics of robot operation if introduced. The above results from one hospital clearly explained the need for an intelligent mobile robot that could not only increase the efficiency and effectiveness of the hospital but also help in relieving the doctors of a great amount of stress, which would in turn improve their performance toward patients.



© CAN STOCK PHOTO/KIRILL

Digital Object Identifier 10.1109/MPOT.2013.2247096
Date of publication: 22 July 2013

Hardware design

ER 1 Robot kit

The hardware used in the project was a commercial robot kit called the ER1 Personal Robot System, supplied by Evolution robotics [1]. The robot kit includes the control software, aluminum beams and plastic connectors to build a chassis, two assembled nonholonomic scooter wheels powered by two stepper motors, one 360° rotating caster wheel, a power module, a battery (12 V, 5.4 A), and a Web camera. The robot also carries additional accessories, three infrared sensors, extra beams, and connectors for reinforcement. A laptop computer, the IBM Think Pad G40 (Pentium 4 processor, 2.0 GHz with 512 MB RAM), was used as a controller device, and Windows XP Professional was loaded as the operating system. A rectangular structure was constructed behind the robot to act as a backpack, using the remaining aluminum beams to carry small equipment like medicine trays and a separate compartment for documents and files (Fig. 1).

Mini camera mount

A camera mount was made for controlling the horizontal motion of the vision camera. The power supply for the camera mount was taken from the digital outputs of the robot control module (RCM). The RCM consists of two Pilot MC3410 motion processors (one for each stepper motor), an FTDI FT232AM chip (for USB to serial data conversion), four A3959SLB Full Bridge motor drivers, and a power regulator with 104FA-5 to 5v and 1A. Motors having greater torque but as low revolutions per minute as possible were utilized for precise maneuverability. The design consisted of a mechanism that utilized a worm gear to achieve a nominal speed for a 180° horizontal motion of the camera. With a 3/8-in worm diameter, the threads per inch (TPI), pitch, depth, and minimum length for the worm were calculated using a series of formulae. To keep a center distance and preferred transmission ratio of 1:52 [2], the corresponding diameter, diametral pitch, pitch, angle, and depth for the worm wheel were calculated. The motor used was a 12 V dc, which was stepped down and operated on 5 V. A small H-bridge circuit was used to drive the dc motor using a ULN 2003 transistor array. The drive circuitry allowed the camera to move clockwise and counterclockwise (Fig. 2).

Collision detection mechanism

Using a simple technique, an effective collision detection mechanism was designed. A small circuit using only switches was used as a bump sensor. A total of 22 switches were shorted in a line on eight equally sized and separated Veroboards (electronics prototyping board characterized by a grid of holes, with wide parallel strips of copper cladding running in one direction all the way across one side of the board). All of the switches were independent from each other. Even if one of the switches were pressed, it would detect a change in the analog voltage and in turn would mean that the robot had detected a collision and should change its current path of motion or retreat. The two distinct properties of the bump sensor mechanism were that it had a low activation force, a wide face design, and independent left and right contact triggers (Fig. 3). Infrared (IR) sensors were used to detect obstacles. This provides dynamic, real-time feedback, allowing viewing sensor reflection values of various objects. The closer a sensor comes to an obstacle, the higher the reflection strength along the threshold values set for the robot. Three IR sensors were placed on the robot in a way so as to cover the maximum area in front of the robot. Because of such a placement, the robot was able to detect larger obstacles that were in front of the robot.

A gripper arm allowed the robot to grasp, transport, and release cans, envelopes, light books, and files. The gripper arm's design resulted in easy operation and useful functionality in a single accessory. The gripper employs an IR LED and receiver to detect when an object is between the gripper's two pincers. Unless programmed otherwise, the pincers were ready to close as soon as an object was detected via the IR sensor.

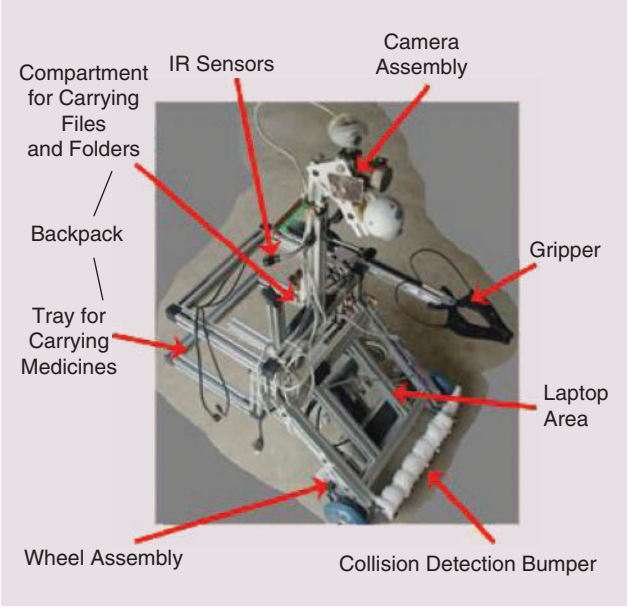


Fig. 1 The hardware design.

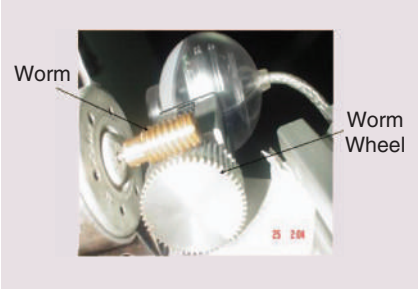


Fig. 2 The camera mount assembly.

Fig. 3 The collision detection mechanism.

Software

Robot control center (RCC)

With the above design it was decided to program the robot in such a way that it should be able to follow predefined paths and reach a specified goal while avoiding obstacles. The RCC (graphical user interface used to program the ER1 robot) allowed the ER1 to recognize objects, colors, sound levels, and words; to send and receive e-mail; to act on schedule; to move around autonomously or by remote control; to play sounds and music; or to take pictures and video. The RCC software can be used to combine these individual behaviors into any number of complex behaviors. It is used to control and receive feedback from the robot. On the left side of the screen is an area to show what the robot is seeing, a list of any recognized objects, any additional

video input from an obstacle avoidance camera, and robot movement controls. The rest of the RCC software consists of a set of "if" conditions and "then" actions that can be used to create a wide variety of robot behaviors or tasks.

It is possible to link together a sequence of up to 288 separate robot behaviors enabling the robot to perform a number of complex maneuvers. The socket application programming interface (API)/command line interface allows creating simple programs that send commands directly to the robot. The APIs can also be used to gather sensory data from the robot. The commands and sensory feedback are sent through TCP sockets, allowing the user to command the ER1 robot using any programming language on any platform that supports TCP/IP sockets. The command line API also supports access to the 15 analog IO ports and 16 digital IO ports of the ER1 robot control module. This allowed us to add custom hardware devices (motor control, collision detection) controlled by the software through the API.

Path planning software

The software to have the robot follow predefined paths was written in Visual Basic 6.0 [3], which had two options in it—the auto mode and the the manual mode.

Auto mode

The auto mode utilized the A* algorithm [4a, 4b] to solve for the shortest path between a start and a goal cell around obstacles that are placed by the user. The software requires the user to make a map clearly indicating the start and end points. Permanent obstacles lying in the path of the robot are marked. The shortest path is solved from the start to the goal point using the A* algorithm. The area covered by the robot using this software is 100 × 100 ft (it can be changed according to the requirements of the hospital). Each square on the grid is called a cell. The A* algorithm uses lists to explore possible solution paths. The two lists are called open and closed. Cells that are to be investigated are placed in the open list while cells that have been investigated are placed in the closed list and are not investigated again. Each cell is given a score F = G + H.

G score = the absolute value from start cell to the cell whose F score is being found. G = (X coordinate of start cell − X coordinate of investigating cell) + (Y coordinate of start cell − Y coordinate of investigating cell).

H score = the absolute value from the goal cell to the cell whose F score is being found. H = (x coordinate of investigating cell − X coordinate of goal cell) + (Y coordinate of investigating cell − Y coordinate of goal cell).

F = G + H.

The following four steps illustrate the algorithm.

Step 1

Each cell has a status. True status means a cell is a blocked cell while a false status means a cell is unblocked.

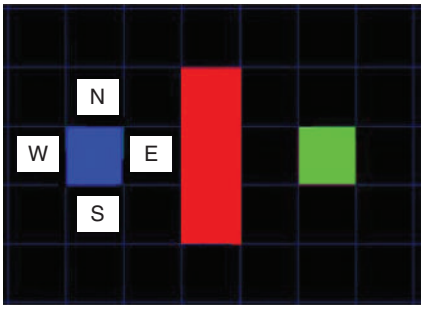


Fig. 4 Step 2.

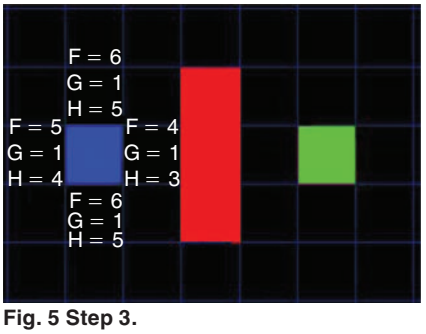


Fig. 5 Step 3.

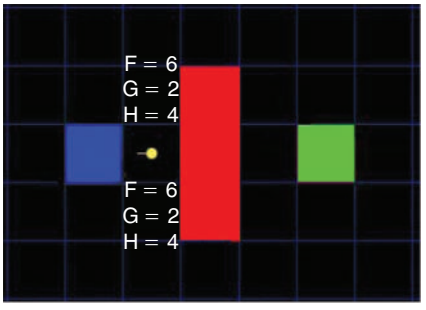


Fig. 6 Step 4.

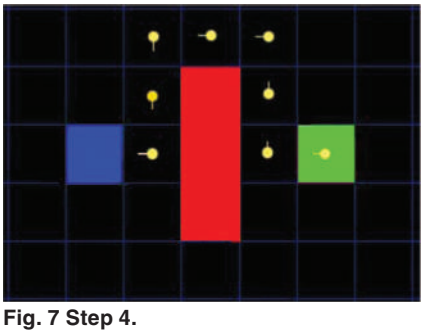


Fig. 7 Step 4.

Blocked cells are not added to the open list. First the start cell is investigated and then added in the closed list.

Step 2

Four adjacent cells of this investigating cell are added to the open list: north, south, east, and west. "F" scores of each of these four cells are calculated (Fig. 4).

Step 3

The cell with the lowest F score is selected. Its parent cell is the start cell. This cell is now removed from the open list and added to the closed list. F scores of its adjacent north and south cells are calculated. The one with the lowest F score is selected (Fig. 5).

Step 4

This process is repeated again and again until we reach the goal cell. We worked backwards from the goal cell, going from each cell to its parent cell until we reached the starting cell. This is the shortest path around the marked obstacles (Fig. 6 and Fig. 7).

At the end of one iteration, the A* algorithm will check to see if it has found the goal cell, and if not, it will loop back and investigate just the chosen cell with the lowest F score. If the goal cell was found, then the solution would be displayed to the user. Simply starting with the goal cell and then proceeding to the parent cell until the start cell is discovered back tracks the solution path. There was still a problem, an unforeseen obstacle that comes in that shortest path that was solved by the A* algorithm. For this purpose, real-time obstacle avoidance was programmed, giving the user the option of either using the camera or the IR sensors to detect obstacles.

Manual mode

The main aim of manual mode was to give an additional option and flexibility in the usage of the robot in the hospital environment. The salient features of manual mode include manual control of drawing an exact path (map) that the robot should follow (for routine tasks). This would allow the doctor to draw and save maps for different routes throughout the hospital that could be simply utilized later by loading the appropriate map and sending the robot immediately.

The manual mode has the following functionalities: A 50 × 50 grid map was representative of 100 ft × 100 ft area. A path can be drawn exactly the way the user wants the robot to move on this map. No

blocked cells are drawn here as in the auto mode, instead nodes are drawn. Nodes are points where the robot turns or stops for any given period of time. An option is also available to control the robot and camera through the keyboard of the laptop.

At its start, the robot aligns itself in the direction as that on the grid and displays the movement as the robot changes direction throughout its path. The first node is marked in green whereas the rest are red in color. A blue line appears between the two nodes representing the distance between two nodes. Node numbers of the marked nodes on the map are displayed. Dynamic direction display of the robot provides continuous mapping by knowing the exact whereabouts of the robot. The coordinates and distance between the current and previous node are also displayed. By providing a time in seconds, a break at the most recent node can be programmed for the specified period of time. The robot can move through the nodes continuously unless a "stop at node" option is used. Stop at node stops the robot motion at that point.

A single node can be erased from the map at any time during the robot's path. The refresh button clears the grid and refreshes the node number counter. Linear and angular velocity settings of the robot and camera through keyboard control are also provided. The images of doctors can be scrolled to find a specific doctor's saved map, which could then be traversed. A motion detection scheme was deployed in which a reference frame was compared with each incoming frame to calculate the change in color and intensity and triggering the robot in motion.

Future work

The work done in this project can be taken further in three specific directions.

1) Department delivery system: In such a system the robot can have a built-in Web server that the doctors can log in to and use to post requests for deliveries of medicines and documents via wireless LAN.

2) An integration of the two modes of operation (auto and manual) would help in implementing a synergy between manual predefined maps and a real-time automatic routing (auto mode). Such a scenario would provide the flexibility in robot maneuvers in known and unknown environments.

3) The implementation of behavior-based robotics: All the information is gleaned from the input of the robot's sensors regarding obstacles in the auto and manual mode. The robot utilizes this information to react to the changes in its environment.

Experiments and overcoming problems

The experiments with the ER1 were initially conducted in a straight corridor of the hospital basement with dim lighting conditions. As a result, the robot showed both desired and problematic behaviors in terms of image processing. The motion trigger (of the robot) on color did not give accurate results, as the robot would take extreme color values like black or white even though the color specified would be different (like blue or green). The visual pattern recognition algorithm of the RCC worked quite effectively for object recognition purposes but only when eight to ten pictures of a particular scene were stored and trained in the software library.

The laptop battery proved to be a big hurdle in running the robot on long stretches and required continuous recharging. While making a pan mechanism for the vision camera, it became very difficult to find a 5-V motor that would have a large torque and small revolutions per minute. This was overcome by using a 12-V dc motor (run on 5 V) and a calculated worm gear ratio of 1:52. This large gear reduction brought the revolutions per minute to a considerable low value to precisely control the camera.

During experiments on the obstacle avoidance capability, at small distances (under 1 ft) the robot would not return to its original path after avoiding the obstacle. This meant that very small distances like 1 ft or under gave inaccurate results for obstacle avoidance. The positioning of the three sensors was challenging, as they should not conflict with each other when an obstacle would come in front, making the robot totally confused on how to react. This problem was overcome by placing one sensor facing forward while the other two were placed facing backwards, to achieve different stimuli for the robot.

The lesson learned with respect to teaching objectives is that entry level robotics students can add or change various functionalities in their algorithms without worrying about a graphical user interface (GUI). This allows them to focus on robotics specific programming rather than the GUI itself. Another learning aspect is to get a sense of the practical challenges students can experience while performing experiments on such a platform. As ER1 is a robotic peripheral rather than a stand-alone system, students learn not only computation but also interfacing basic self-developed hardware components (e.g., sensors). The modular nature of the platform reinforces students' views of what constitutes a robotic system and the notion that a computer is a robot that only requires sensors and actuators to realize the inherent potential.

Conclusion

The efforts with Evolution's ER1 suggest that it is a promising platform for the development of specific robotics functionalities. The path planning software was designed and tested successfully in the hospital environment, which demonstrates that the ER1 can open up algorithms for student experimentation that would otherwise be difficult or impossible to attempt. The obstacle avoidance using the second camera proved to be a better tool for avoiding smaller objects and in areas where lighting conditions remained the same throughout. Because the focus was programming for robotic autonomy, two out of the three IR sensors pointed away from the laptop so that the robot interacts with the world and also avoids larger obstacles. Effective yet simple techniques for collision and motion detection for the robot were implemented with positive results. The ER1 thus represents one of the most accessible and reconfigurable platforms that is believed to help redefine the content and practice of robotics and artificial intelligence.

Read more about it

- Evolution. (2005). [Online]. Available: http://www.evolution.com
- V. B. Bhandari, *Design of Machine Element*. New Delhi, India: Tata McGraw-Hill, 2010.
- E. Petroutsos, *Mastering Visual Basic 6*, 1st ed. New Delhi, India: BPB Publications, 2000.
- GameProgramming. (2005). [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming
- StarTutorial. (2005). [Online]. Available: http://www.policyalmanac.org/games/aStarTutorial.htm

About the author

Mohammad Saud Khan (mohammadsaud19@gmail.com) is currently a doctoral candidate and researcher working in the area of innovation management and entrepreneurship. He earned a master's degree in management, technology, and economics from ETH Zurich.