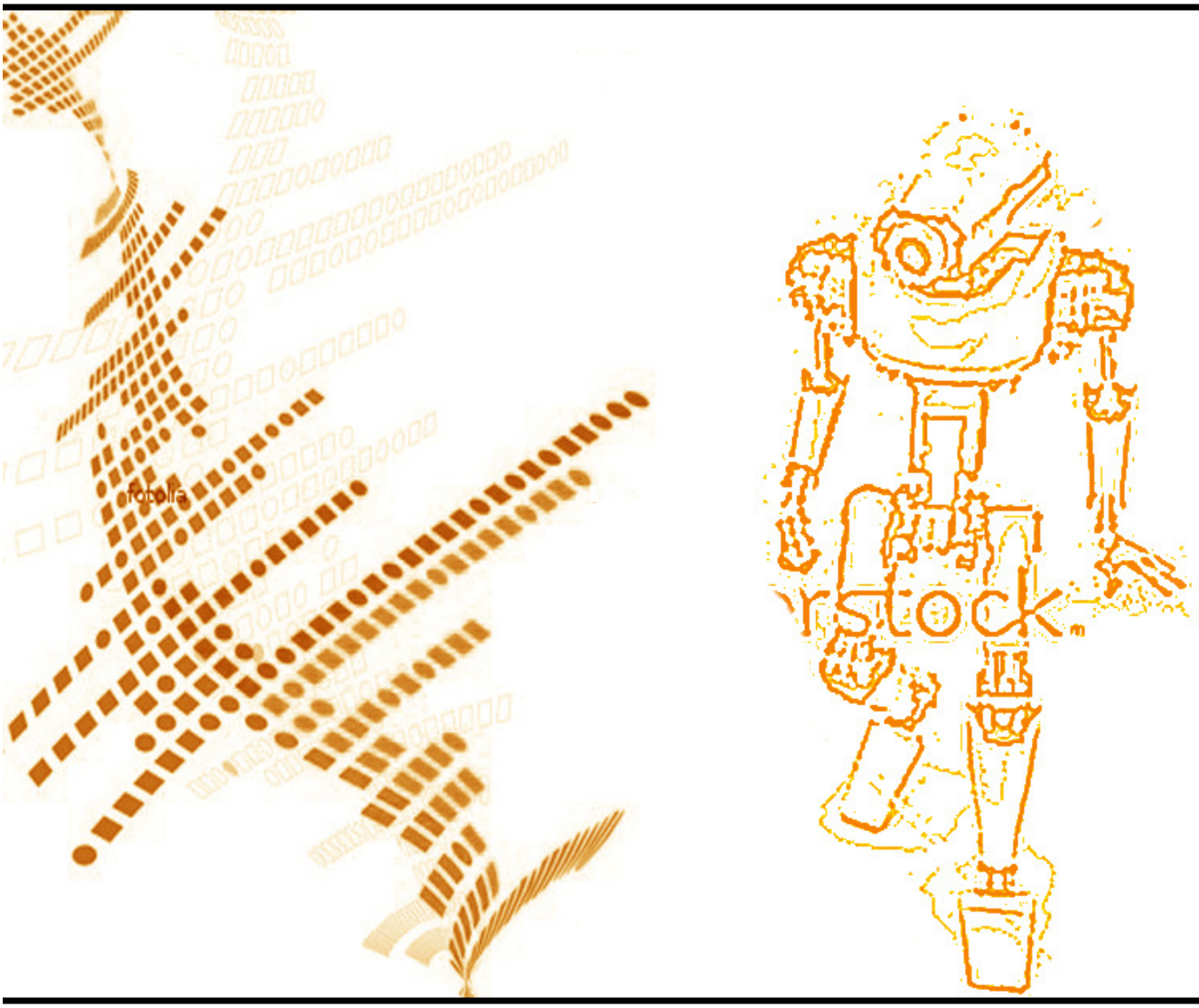
Volume 2 ▪ Issue 1 ▪ March 2011

INTERNATIONAL JOURNAL OF

# ROBOTICS AND AUTOMATION (IJRA)

ISSN : 2180-1312
Publication Frequency: 6 Issues / Year

CSC PUBLISHERS
http://www.cscjournals.org



# INTERNATIONAL JOURNAL OF ROBOTICS AND AUTOMATION (IJRA)

**VOLUME 2, ISSUE 1, 2011**

**EDITED BY**
**DR. NABEEL TAHIR**

ISSN (Online): 2180-1312

I International Journal of Robotics and Automation (IJRA) is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJRA Journal is a part of CSC Publishers
Computer Science Journals
http://www.cscjournals.org

# INTERNATIONAL JOURNAL OF ROBOTICS AND AUTOMATION (IJRA)

Book: Volume 2, Issue 1, March 2011
Publishing Date: 04-04-2011
ISSN (Online): 2180-1312



IJRA Journal is a part of CSC Publishers
http://www.cscjournals.org


Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

**CSC Publishers, 2011**

# EDITORIAL PREFACE

Robots are becoming part of people's everyday social lives - and will increasingly become so. In future years, robots may become caretaking assistants for the elderly or academic tutors for our children, or medical assistants, day care assistants, or psychological counselors. Robots may become our co-workers in factories and offices, or maids in our homes. It is the fourth issue of volume first of International Journal of Robotics and Automation (IJRA). IJRA published six times in a year and it is being peer reviewed to very high International standards.
.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 2, 2011, IJRA appears in more focused issues. Besides normal publications, IJRA intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

IJRA looks to the different aspects like sensors in robot, control systems, manipulators, power supplies and software. IJRA is aiming to push the frontier of robotics into a new dimension, in which motion and intelligence play equally important roles. IJRA scope includes systems, dynamics, control, simulation, automation engineering, robotics programming, software and hardware designing for robots, artificial intelligence in robotics and automation, industrial robots, automation, manufacturing, and social implications etc. IJRA cover the all aspect relating to the robots and automation.

The IJRA is a refereed journal aims in providing a platform to researchers, scientists, engineers and practitioners throughout the world to publish the latest achievement, future challenges and exciting applications of intelligent and autonomous robots. IJRA open access publications has greatly speeded the pace of development in the robotics and automation field. IJRA objective is to publish articles that are not only technically proficient but also contains state of the art ideas and problems for international readership.

In order to position IJRA as one of the top International journal in signal processing, a group of highly valuable and senior International scholars are serving its Editorial Board who ensures that each issue must publish qualitative research articles from International research communities relevant to signal processing fields.

IJRA editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJRA. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJRA provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

**Editorial Board Members**
International Journal of Robotics and Automation (IJRA)

# TABLE OF CONTENTS

Volume 2, Issue 1, March 2011

## Pages

# Design of a Low-cost Autonomous Mobile Robot

**Kasun K. Jinasena** kasun@sjp.ac.lk
*Faculty of Applied Science*
*University of Sri Jayewardenepura*
*Gangodawila, Nugegoda, Sri Lanka*

**Ravinda G.N. Meegama** rgn@dscs.sjp.ac.lk
*Department of Statistics & Computer Science*
*Faculty of Applied Science*
*University of Sri Jayewardenepura*
*Gangodawila, Nugegoda, Sri Lanka*

**Abstract**

Detection of obstacles during navigation of a mobile robot is considered difficult due to varying nature in terms of size, shape and location of the obstacle as well as ambient light that interferes with infra-red (IR) signals of the robot. In this paper, we propose a novel low-cost method that successfully guides a robot along a path using image processing and IR sensor circuits. A high continuous IR signal is converted into a low continuous IR signal by means of a demodulation circuit that enables a peripheral interface controller to receive this low continuous IR signal and take relevant decisions based on the signal. The images taken from a web camera are preprocessed to remove noise and detect edges. Subsequently, an image processing routine effectively calculates the angle to be rotated of the front wheels using a scan line algorithm. A minimum mean distance error of 2.45 was observed in tracking the path at a signal-to-nose ratio of 26.50. The accuracy of speech recognition was 92% for two voice training sessions.

**Keywords:** Mobile Robot, Infra-red Signals, Voice Recognition, Robot Navigation, Infra-red Communication

## 1. INTRODUCTION

Current research challenges on intelligent autonomous robots include navigation, task planning and coordination, learning and adaptation, cooperation between robots and humans [1]. Each model incorporates its own perceptual, modeling, and planning requirements.

Recent studies indicate a growing trend towards tracking of paths using mobile robots. A four wheel robot powered by a single motor was developed in the University of Aveiro [2] that uses complex mechanical techniques for both front and rear wheels. By using high-end firewire cameras and a built-in laptop computer, it managed to navigate along a path interpreting traffic lights. A unique two-body mobility platform with a hybrid electric power system was introduced in [3] resulting in an intelligent robotic vehicle with a sensor suite. In subsequent years, a three-wheeled, differential driven vehicle that uses two brushless smart DC servo motors and a hybrid system that provides sufficient power for the vehicle to run continuously at a length of more than 10 hours was developed [4]. The four sensors, (digital camera, scanning laser range finder, digital compass, and differential global positioning system (GPS)), were integrated into a streamlined system architecture with a laptop serving as the computational core of the system.

All these research finding over the years have contributed immensely to the advancements in robotics applications. More recent studies in mobile navigation techniques include autonomous flying vehicle [5], mine detecting vehicles to be used in military applications [6], wall climbing robots where robotics vehicles have been able to walk along a vertical plane using electromagnetism, electrostatic adhesion or air suction [7], medical surgery where the motion of

the heart is tracked in three-dimension [13] and micro robots used in biomedical applications to remove plaque in arteries [14].

Although we are able to come across many applications in robotics that addresses a specific area of research, one important fact that must be considered when navigating an autonomous robotic vehicle outdoors is interferences caused by sunlight [15]. Especially when the robot is guided by IR sensors, there must be a mechanism to distinguish between the sun's IR rays and the rays reflected back from an obstacle located at a particular distance away from the moving vehicle.

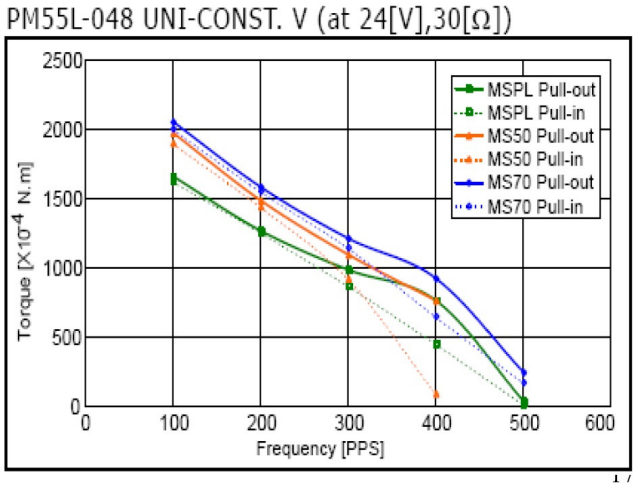



**FIGURE 1:** Torque charactisircs of stepper motors

## 2. METHODOLOGY

This research has its own hypothesis such as the track is on a plane surface with different color to its background, there are no junctions, the environment is static or dynamic, and the track with locations of obstacles are not known at the beginning of its navigation. A mechanical structure was designed for the problem involved with electronic solutions for the speed controllers and motor drivers. Image processing, speech recognition and text to speech software were used for the guidance algorithms.

The following sections describe the methodology we have used in designing and implementing the autonomous mobile robot.

### 2.1 Chassis Design

The chassis provides a solid support for the electronic boards, camera and sensors. Although aluminum pipes are the preferred material to build the robot, iron pipes were selected due to their availability and rigidness. The vehicle moves on three wheels. The two front wheels are each powered by a stepper motor while the rear wheel, hinged freely to the chassis, turns automatically.

The platform is made up of iron and was designed with modularity in mind, i.e., the chassis should provide a solid support for electronic boards, camera and any other items to be mounted in future.

The two main points of concern in selecting a material for the chassis are rigidness and lightness of the material as lightness of the chassis helps save power and enhances transmission. On the other hand, rigidness assists the vehicle to be stronger and carry more weight.

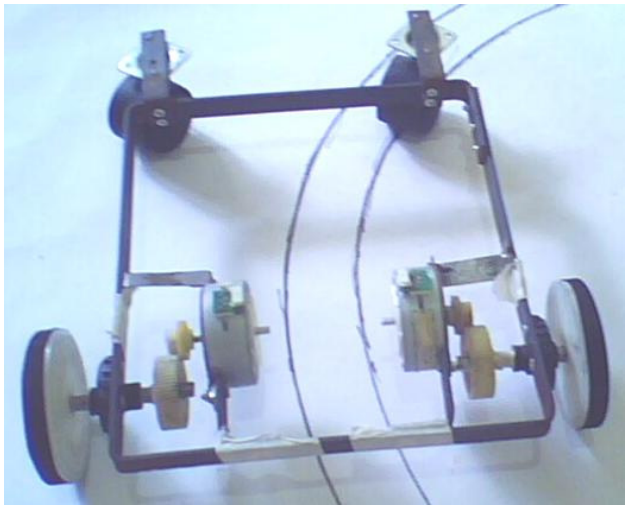


**FIGURE 2**: Wheel setup of the vehicle

### 2.2 Wheel Design

Unipolar 24 V, 800 mA per winding (30 Ω) stepper motors are used to power the front wheels. Each step motor will rotate 7.5 degrees as there are 48 steps for a single rotation. Each motor has maximally 0.2 Nm pull-out torque and 0.16 Nm pull-in torque with 100 steps per second with a maximum of 500 steps per second. A gear system is also used to increase the torque by 3 times. It consist of two wheels having teeth at a ratio of one to three to increased the torque by three times while reducing the speed by three times.

Fig. 1 illustrates the torque characteristics of the stepper motors while Fig. 2 indicates the setting up of wheels of the robot vehicle where the front two are steering wheels each powered by a stepper motor and the rear are free wheels. The vehicle turns by changing speed of its front wheels while rear wheels turn automatically. This will allow the robot to maintain more stability than a three wheel vehicle.

The vehicle transmission consists of two stepper motors responsible for steering the vehicle along the surface. Although rear driven wheels are suitable for heavy vehicles such as buses and lorries, front drive is used mainly for lighter vehicles so that they can be easilly manoeuvreable.

As such, a front driven mechanism is chosen for our robotic vehicle. Two motors are connected to the front wheels of robot to make turning easier.

### 2.3 Obstacle Detection

The vehicle is able to detect certain obstacles along its path and take relevant decisions based on the type of the obstacle. Ultrasound and Infra Red (IR) are the two most widely used obstacle detection methodologies in Robotics [8,9]. By using Ultrasound, the velocity and the direction of a moving obstacle can be identified easily through the Doppler effect. Because ultrasound waves are susceptible to temperature and humidity, IR, whose speed is constant over a particular media, is used in most mobile digital electronics [12].

Our vehicle employs IR to detect obstacles. The main challenge in using IR for this purpose is to distinguish between the robot's and nature's IR at the receiving end. Under strong sunlight, it is possible that a high amount of IR signals may incident on the sensor. As such, we need to overcome the natural IR exerted by the ambient light. One of the solutions to overcome this issue is to investigate the intensity of the two IR signals as natural IR signals has a lower intensity compared to the robot's. Natural IR, however, is not constant over a given period. As such, a

modulated (pulsed) IR signal, obtained with frequency generators, is used to overcome this problem. This modulated pulse signal is shown in Fig. 3.

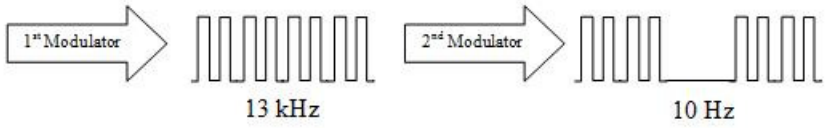



**FIGURE 3:** IR signal modulation

As depicted in Fig. 4, the 1st modulator converts the continuos signal into a 13 kHz square wave and then the 2nd modulator encloses that 13 kHz signal in a 10 Hz signal.

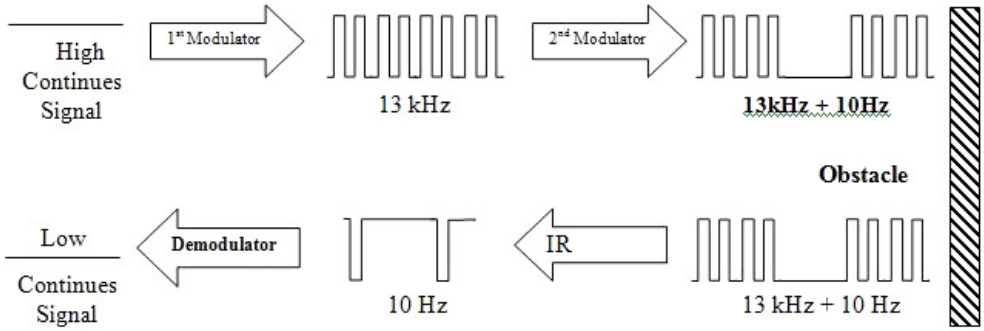



**FIGURE 4:** Demodulating a reflected signal

Finally, the transmiter emits this signal as IR rays. IR is heat radiation which is absorbed readilly by black rough surfaces while reflected by white shiny surfaces. Therefore, in order to make the robot sensitive to all obstacles equally, once the IR module receive signals reflected by an obstacle with an output of 10kHz (square signal), it is converted to a continuos signal by the demodulator. The result is then sent to a PIC microcontroller to make necessory decisions.

### 2.4 IR Signal Demodulation

The task of the Demodulator is to demodulate the signal originating from the IR module and regenerate a continuos signal. This is achieved by using a capacitor and properly biased transistors. This transistor circuit causes the signal to be invert but programmatically corrected. Fig. 5 shows the circuit diagram of the demodulator.

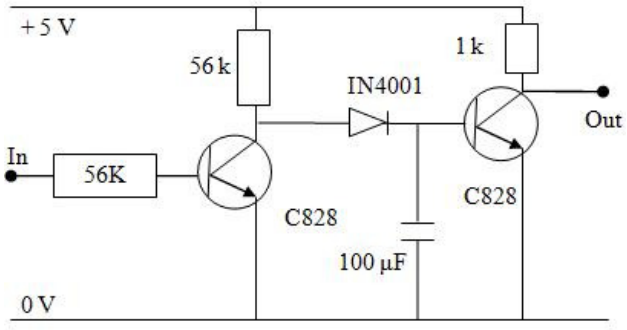



**FIGURE 5:** IR Signal demodulation circuit

### 2.5 Stepper Motor Driving Circuit

Fig. 6 shows only a single channel of the stepper motor driver circuit. The actual circuit carries eight of these circuits for eight windings of two motors. We have integrated a mechanism to protect the IN 4001. A photo coupler makes the isolation between high voltage and the low voltage sections. IN4001 protects the D313 from back electro motive force (EMF) of the motor.

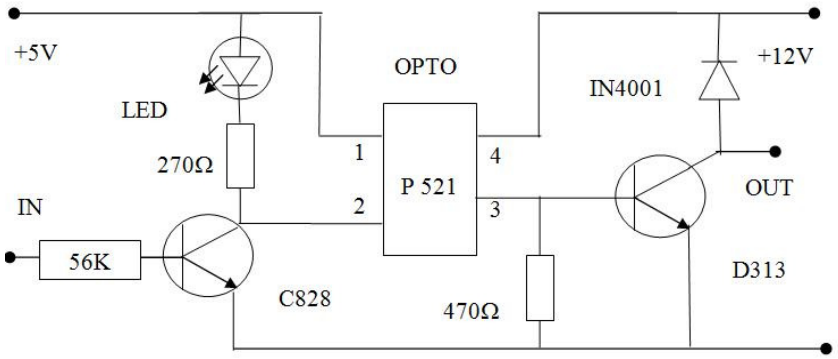



**FIGURE 6:** Stepper motor driving circuit

### 2.6 Image Processing & Vision

The input video stream is captured via web camera of 1.3 mega pixels resolution and the frames are extracted as images. These images are then pre-processed to remove Gaussian noise and detect edges.

As in Fig. 7, two lines scan the acquired image to detect the edges of the path. The angle θ of the inner and outer covertures is separately calculated. At each scan line, the curvature is calculated based on the intersection of lines and the edges. Based on the value of the angle θ, the left and right wheels are stopped to make the turns to left and right, respectively.

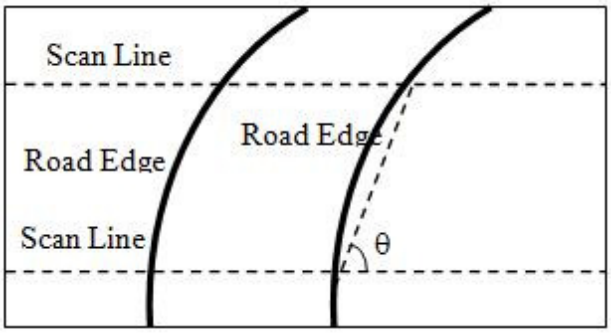



**FIGURE 7:** Detection of path using curvature

At this research, we carried out experiments to determine the vehicle's ability to track a path that consists of two edges. These edges have been drawn in black color on a white background. More complex tracks with multiples edges will obviously require much advanced image processing routines and fast computing power.

Fig. 8 depicts the main information flow of tracking the edges of a path using image processing technques. It can be seen that the vechile makes a decision to turn the wheels based on the location of detected edges.

### 2.7 Speech Recognition

We know that every speech recognition system consists of an engine that translates sound waves into text and a list of speech commands. In this research, we used the SAP15 engine that is integrated into Windows XP operating system. This engine was trained using windows voice training wizard in which a profound usage of grammar increases the probability of misinterpretations.

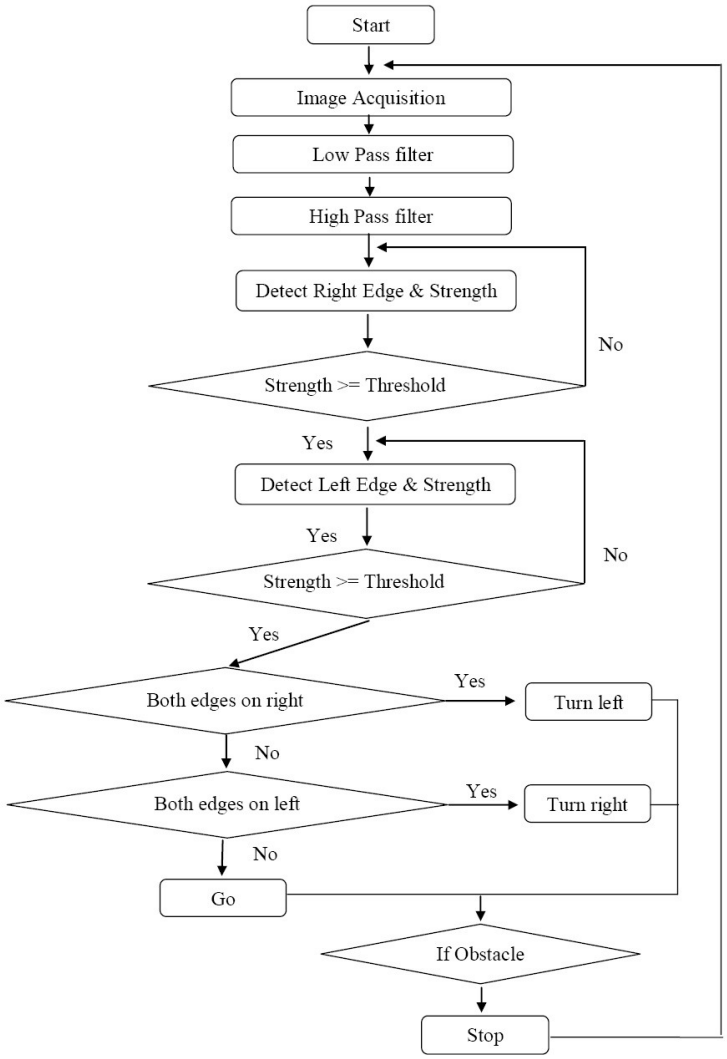



**FIGURE 8:** Flow chart of path tracking procedure

As such, we tried to keep the set of grammar as small as possible without lose of information. Grammar, which contains the commands “Go”, “Left”, “Right” and “Stop”, is then fed into an XML file.

### 2.8 Main Board

A photographic illustration of the main circuit board is given in Fig. 9 showing the IR sensor and associated circuitry. The main circuit board constist of **16F877A** PIC as shown in Fig. 10. The pins 37- 40 are being used for the four voice commands given to the vehicle while pins 27 – 30 are reserved for the right stepper motor driver. The left stepper motor is driven by pins 19 – 22.

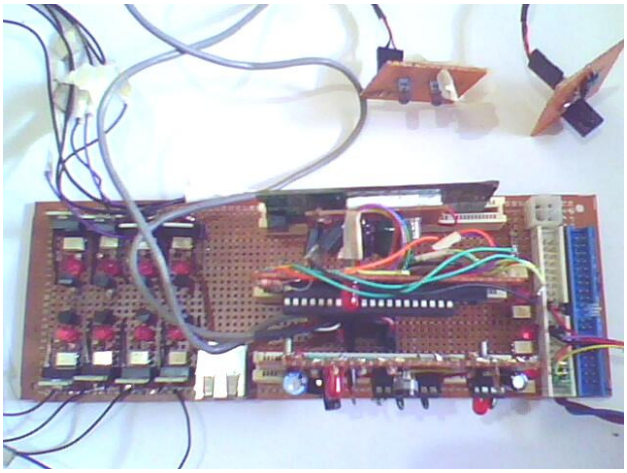


**FIGURE 9**: Circuit board of the vehicle.

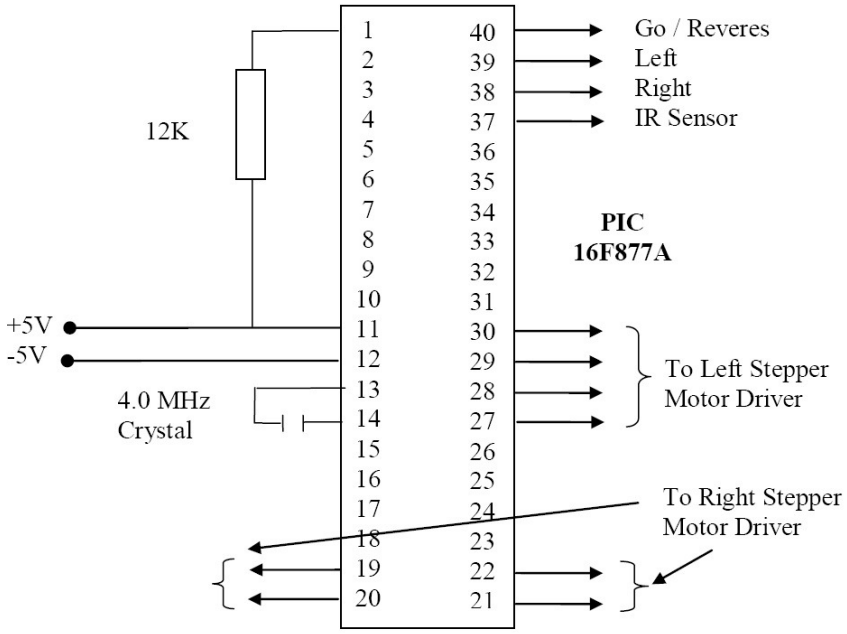



**FIGURE 10:** Main circuit board

**2.9 PCB Design**

Fig. 11 illustrates the printed circuit board (PCB) of the vehicle.

**2.10 Mechanicla Structure**

Fig. 12 shows the dimensions of the robotc vehile. The net weight of the vehicle was found to be 1.2 Kg that includes all the circuitry, wheels, web cam and IR sensor.

The selected dimensions of the robotic vehicle makes it easier to manipulate along a path with sharp edges. Also, the compact size of the structures requires less power to drive forward or backwards.

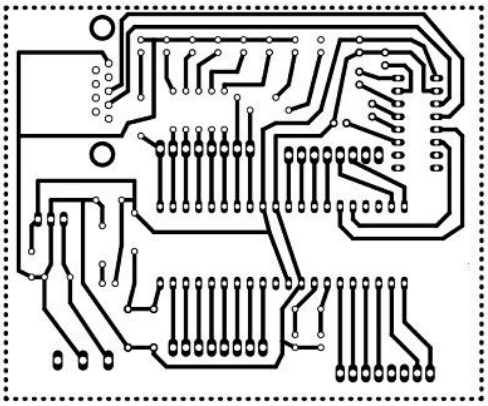


**FIGURE 11:** PCB of the robotic vehicle.

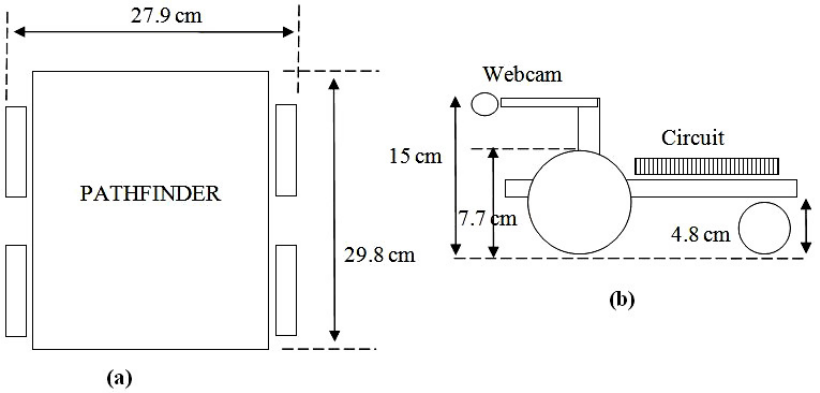



**FIGURE 12:** Dimensions of the robot: as seen from (a) above and (b) right hand side.

## 3. RESULTS AND DISCUSSION

### 3.1 Hardware and Software Requirements

A Personal Computer with a Pentium IV (or above) class Processor, 3.0 GHz, 512 MB or above RAM capacity and 10 GB hard disk space are preferred. All USB ports should be enabled and a microphone, speakers/head phone and web camera are the other accessories that are required. We used Microsoft Windows XP operating system with Microsoft's speech engine 5.1 and DirectX 9 with .NET framework 2.0 and a device driver for the web cam and Useport.sys for port access are required.

### 3.2 Transmission

As shown in Table 1, the vehicle is able to travel forward or backward with 3.95 inches per second on a flat surface while it turns 360 degrees around any of its front wheels, clockwise in 17.35 seconds and anti clock wise in 17.02 seconds. It travels in a straight line with an accuracy of more than 98.64 %. Speed can be increased by decreasing the period of delay while power decreases.

| Characteristics | Result |
|---|---|
| Forward travel | 3.95 inches per second |
| Forward travel deviation | 1.36% |
| Clockwise rotation | 17.36 seconds per 360º |
| Anti-clockwise rotation | 17.02 seconds per 360º |

**TABLE 1:** Transmission characteristics of the robotic vehicle.

### 3.3 IR Sensitivity

At the beginning of the research, an IR sensor typically found in a computer mouse was used to detect obstacles. It managed to detect obstacles at a distance of more than one foot with a single IR light emitting diode (LED). The major problem, however, was sun light. Since this IR sensor does not have a filter, we found a malfunctioning system even under minor sun light at dawn. The IR modules that are used in televisions, resulted filtering a pre define frequency from sun's IR rays. It was seen that IR modules band pass filter was highly sensitive to 13 kHz IR. Therefore, a modulator and demodulator circuits were designed to support this frequency.

It was also observed that high sun rise still can affect obstacle detection system because the frequency of mixed signal is dominated by the frequency of the signal that has the highest amplitude. This is easily solved by using more IR LEDs that causes the system to be over sensitive under minor sun light. In a practical situation, such as a remote control system for toys, the user is allowed to select the sensitivity level manually.

The distance between the IR transmitter and the IR receiver affects the sensitivity as shown in Fig. 14. Increasing the distance between the IR transmitter and the IR receiver decreases the sensitivity whereas decreasing the distance between the IR transmitter and the receiver increases the sensitivity. Results in Table 2 were obtained by changing the distance between the transmitter and the receiver for a white color, landscaped, A4 size obstacle.

| Distance between transmitted & receiver (cm) | Straight detection distance (cm) | Width of detection region (cm) |
|---|---|---|
| 3 | 24 | 22 |
| 4 | 20 | 20 |
| 5 | 17 | 17 |
| 6 | 16 | 14 |

**TABLE 2:** Distance vs width of sensitivity

### 3.4 Color Sensitivity

It was evident that the color of the object was a major factor that determines the correct detection of objects. Fig. 15 shows the sensitivity region of a landscape, A4 size obstacle for the 8 colors. As shown in Table 3, black had a significant deviation while other colors had minor deviations.

### 3.5 Voice Recognition

It was witnessed that all voice commands did not have an equal accuracy of recognition. The accuracy was highly dependent on the speech pattern of the user.

| Color | Distance (cm) |
|---|---|
| White | 24 |
| Yellow | 22 |
| Red | 20 |
| Orange | 19 |
| Green | 21 |
| Blue | 18 |
| Brown | 19 |
| black | 3 |

**TABLE 3:** Color vs distance of sensitivity

The voice of the speaker needs to be trained first and then, the accuracy can be increased by having more voice training sessions with the system. Fast speech causes less accuracy or even unrecognized commands while long phrases resulted in a higher accuracy than shorter phrases.

The accuracy also tends to get low with increasing number of commands. When two or more commands have similar sound characteristics, the accuracy of those commands decreases. Due to disturbances from ambient noise, a good, high quality microphone was used. Table 4 gives the results of accuracy of the system after one voice training session. Each command was tested ten times with different tones. However, we observed an increase in the accuracy when the number of training sessions was increased. Table 5 provides the average accuracy when the number of voice training sessions is two. More training sessions will be required if the vehicle is to recognize voice commands selected from a large vocabulary.

The speech engine was able to recognize voice of different persons for the same set of voice commands.

| Command | Average accuracy (%) |
|---|---|
| Go | 72 |
| Turn left | 67 |
| Turn right | 68 |
| Reverse | 67 |
| Stop | 45 |
| Start CV | 62 |
| Close CV | 66 |
| Who are you | 82 |
| Exit | 76 |

**TABLE 4**: Average accuracy for one voice training session.

| Command | Average accuracy (%) |
|---|---|
| Go | 72 |
| Turn left | 87 |
| Turn right | 88 |
| Reverse | 87 |
| Stop | 65 |
| Start CV | 82 |
| Close CV | 86 |
| Who are you | 92 |
| Exit | 66 |

**TABLE 5:** Average accuracy for two voice training sessions.

### 3.6. Path tracking

Direct application of a high pass filter to the captured image resulted in noise at the output that was highly affected by the path detection module. A significant enhancement was gained by applying a low pass filter to the captured image prior to applying the high pass filter.

The results obtained from image processing were obviously dependent on the resolution of the camera used. In order to keep the cost of the vehicle at a minimum, we used a widely used web camera that can be purchased from any computer store. Also, scanning horizontally on an edge at the middle of the image resulted in several issues especially when there was an object on its side. It was avoided by changing the scan order from the middle to up to the edge. A slight discontinuity or non-uniformity of the road did not make a significant effect to the vision process.

Table 6 provides measured mean distance errors when the vehicle tracks a path under different signal to noise ratios (SNR). The mean distance error *e* was calculated as

$$e(\mathbf{v}, \bar{\mathbf{v}}) = \frac{1}{l}\int_{s} \min_{s' \in \bar{\mathbf{v}}} \left| \mathbf{v}(s) - \bar{\mathbf{v}}(s') \right| ds$$

where **v** is the path of the robot's movement, $\bar{\mathbf{v}}$ is the actual path of the track and *l* is the length of v [10].

| SNR | Mean distance error |
|---|---|
| 26.50 | 2.45 |
| 22.60 | 3.35 |
| 20.15 | 4.42 |
| 18.75 | 5.66 |
| 17.85 | 6.57 |
| 17.30 | 13.11 |
| 16.50 | 15.63 |

**TABLE 6:** Effect of SNR on mean distance error.

### 3.7 Safety of PIC Microcontrollers

Unlike other electronic equipments, PIC microcontrollers are vulnerable to high voltages. It was observed that both hardware and software errors resulted in a damaged PIC. It must be noted that the PIC 16F877A microcontroller is able to provide an output of a maximum current of 200 mA while any of its pins provides an output of a maximum current of 25 mA. Exceeding any of these limits causes a burnt microcontroller. Interferences from other electrical signals also affect the inputs to the microcontroller. Most of such problems, however, can be solved by grounding the pins with a higher resistor, such as 10 K.

### 3.8 Communication

Currently, the robot does all the communication over wired media. Wireless technologies, such as Bluetooth or Wi-Fi, can be introduced so that communication can be done from anywhere even over the Internet. Bluetooth uses the 2.4 GHz radio frequency band to communicate with other Bluetooth devices [11]. However, the issue that needs to be addressed in this scenario is that a majority of popular Bluetooth USB dongles are slave devices without any inbuilt firmware. When it connects to a computer, the computer provides the necessary protocol stack. On the other hand, if it connects to PIC microcontroller, both the USB protocol stack and the Bluetooth protocol stack should be implemented inside the PIC. Since a PIC executes only a single program at a time, a dedicated PIC is needed for this purpose. Although this research uses a PIC16F877A microcontroller, a PIC 18F series also provides the USB interface. IR sensors

IR Sensors tend to get saturated when the distance between the obstacle and the sensor fall below 40 cm. Therefore, it is difficult to use it to measure distances less than 40 cm. When measuring distances that vary between 50 to 70mm, the accuracy of IR sensor ranges from 92 to 95 percent whereas the accuracy of Ultrasound sensor was 90 to 97 percent.

The proposed research show that modulated IR still can be used to detect obstacle even when they are located close to as much as 3 cm. The speed of Ultrasound waves varies with temperature and the humidity] and as such needs correction to obtain correct distances. Since IR is electromagnetic, it neither depends on temperature nor humidity.

## 4. CONCLUSION

This research presents the design, implementation and results of a low cost minatory model of an autonomous land vehicle based on computer vision, speech recognition and IR transmission.

The main feature developed by this project was its ability to track a path avoiding disturbances from sun light using IR, image processing and computer vision. Moreover, the robot can make voice response to its user. Both the path and location of obstacles were not known in advance. Results indicate that the robot is able to recognize speech at higher accuracy when the number of training sessions is increased. We checked the performance of the robot by adding Gaussian noise along the path to be tracked. With the current setup, it successfully tracked the path when the signal to noise ratio of the image is higher than 17.30.

Extending the research for the vehicle to transmit data using wireless or Bluetooth technology is straightforward.

## 5. REFERENCES

[1] P. Alfonso, J. Azevedo, C. Cardeira, B. Cunha, P. Lima and V. Santos, “Challenges and solutions in an autonomous driving mobile robot competition”, In Proceedings of the Controlo Portugese Conference in Automatic Control, Lisbon, 2006.

[2] T. Fong, I. Nourbakhsh, and K. Dautenhahn, “A Survey of Socially Interactive Robots”, Robotics and Autonomous Systems, 42(3-4):143- 166, 2003.

[3] [online] http://www.avt.me.vt.edu/index.htm.

[4] C.F. Reiholtz, "Virginia Tech Autonomous Vehicle: Jhonny-5", Department of Mechanical Engineering, Virginia Tech, USA.

[5] L.O. Mejias, S. Saripallai, P. Cervera and S. Sukhatme, "Visual Surveying of an Autonomous Helicopter in Urban Areas Using Feature Tracking", Journal of Field Robotics, 23(3): 185-199, 2006.

[6] E. Cepolina and M. Zoppi, "Cost-effective Robots for Mine Detection in Thick Vegetation", In Proceedings of the 6th International Conference on Climbing and Walking Robots, Catania, Italy, 2003.

[7] B. Aksak, M.P. Murphy and M. Sitti, "Gecko Inspired Micro-Fibrillar Adhesives for Wall Climbing Robots on Micro/Nanoscale Rough Surfaces", In Proceedings of the IEEE International Conference on Robotics and Automation, 2008.

[8] G. Benet, F. Blanes, J.E. Simo and P. Perez, "Using infrared sensors for distance measurement in mobile robots", Robotics and Autonomous Systems, 40(4): 255-266, 2002.

[9] V.K. Sehgal, R. Sharma, N. Nitin, D.S. Chauhan, A. Srivastava, A. Kumar, Y. Agerwal and A.M. Khan, "Obstacle Sensing and Anti-falling Sensor Robot Using Embedded Processor", In Proceeding of the 11th International Conference on Computer Modelling and Simulation, 2009.

[10] L.H. Staib and J.S. Duncan," Model-based deformable surface finding for medical images", IEEE Transactions in Medical Imaging, 15(6): 859-870, 1996.

[11] Y.C.F. Amin, S.H.M. Fisal, N.Bakar, "Bluetooth enabled Mobile Robot", In Proceedings of the IEEE International Conference on Industrial Technology, 2002.

[12] W. Fehlman and M.K. Hinders, "Mobile Robot Navigation with Intelligent Infrared Image Interpretation", Springer, 2010.

[13] R. Richa, P. Poignet and C. Liu, "Three dimensional Motion Tracking for Beating Heart Surgery Using a Thin-plate Spline Deformable Model", International Journal of Robotics Research, 29(2-3): 218-230, 2009.

[14] J.J. Abbot, K.E. Peyer, M.C. Lagomarsino, L. Zhang, L. Dong, I.K Kaliakaktos and B.J. Nelson, "How Should Microbots Swim?", International Journal of Robotics Research, 28(11-12): 1434-1447, 2009.

[15] G.Anderson, C. Sheesley , J. Tolson, E. Wilson and E. Tunstel, A Mobile Robot System for Remote Measurements of Ammonia Vapor in the Atmosphere, In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2006.

# Identification and Control of Three-Links Electrically Driven Robot Arm Using Fuzzy Neural Networks

**Salam A. Abdulkereem** salam_eng@yahoo.com
*Department of Computers Engineer*
*University of Basra*
*Basra,Iraq*

**Prof. Dr. Abduladhem A. Ali** abduladem1@yahoo.com
*Department of Computers Engineer*
*University of Basra*
*Basra,Iraq*

**Abstract**

This paper uses a fuzzy neural network (FNN) structure for identifying and controlling nonlinear dynamic systems such three links robot arm. The equation of motion for three links robot arm derived using Lagrange's equation. This equation then combined with the equations of motion for dc. servo motors which actuated the robot. For the control problem, we present the forward and inverse adaptive control approaches using the FNN. Computer simulation is performed to view the results for identification and control.

**Keywords:** Fuzzy Neural Control, Robot Control, Forward Adaptive Control, Inverse Control, Adaptive Systems

## 1. INTRODUCTION

In the past decade, the applications of intelligent control techniques (fuzzy control or neural-network control) to the motion control of robotic manipulators have received considerable attention [1], [5]. In general, robotic manipulators have to face various uncertainties in their dynamics, such as payload parameter, friction, and disturbance. It is difficult to establish an appropriate mathematical model for the design of a model based control system. Thus, the general claim of these intelligent control approaches is that they can attenuate the effects of structured parametric uncertainty and unstructured disturbance by using their powerful learning ability without a detailed knowledge of the controlled plant in the design processes. Feed forward neural networks have been shown to obtain successful results in system identification and control [6]. Such neural networks are static input/output mapping schemes that can approximate a continuous function to an arbitrary degree of accuracy. Results have also been extended to recurrent neural networks [7], [9]. For example, Jin et al. [8] studied the approximation of continuous-time dynamic systems using the dynamic recurrent (DRNN) and a Hopfield-type DRNN was presented by Funahashi and Nakamura [7]. As is widely known, both fuzzy logic systems and neural network systems are aimed at exploiting human-like knowledge processing capability. Moreover, combinations of the two have found extensive applications. This approach involves merging or fusing fuzzy systems and neural networks into an integrated system to reap the benefits of both [10]. For instance, Lin and Lee [11] proposed a general neural network model for a fuzzy logic control and decision system, which is trained to control an unmanned vehicle.
In this paper FNN is used to identify and control a three links robot arm. We present the forward and inverse identification as offline learning to use the parameters of this stage in control stage. For control problem, we present the indirect (forward) and direct (inverse) control. Computer simulation implements to view the results of robot arm application.

This paper is organized as follows. Section II presents the dynamic model of a three-link robot arm including actuator dynamics briefly [12], [14]. Section III shows the FNN structure. Section IV FNN identification .Section V presents the FNN control. Section VI presents the simulation results finally section IX shows the conclusion.

## 2. Dynamic model of three links Robot arm

Dynamic modeling of a robot manipulator consists of finding the mapping between the forces exerted on the structures and the joint positions, velocities and accelerations. Two formulations are mainly used to derive the dynamic model: namely the Lagrange formulation and the Newton-Euler formulation. A large number of authors and researchers [15], [17], used Lagrange's approach to drive the general form of robot equation of motion. The Lagrange equations thus taking on the alternative form [18]:

$$I(\theta)\ddot{\theta} + \dot{I}(\theta,\dot{\theta})\dot{\theta} - \frac{1}{2}\left[\frac{\partial(I\dot{\theta})}{\partial\theta}\right]^{T}\dot{\theta} + \frac{\partial F}{\partial\dot{\theta}} = \tau_n \qquad (1)$$

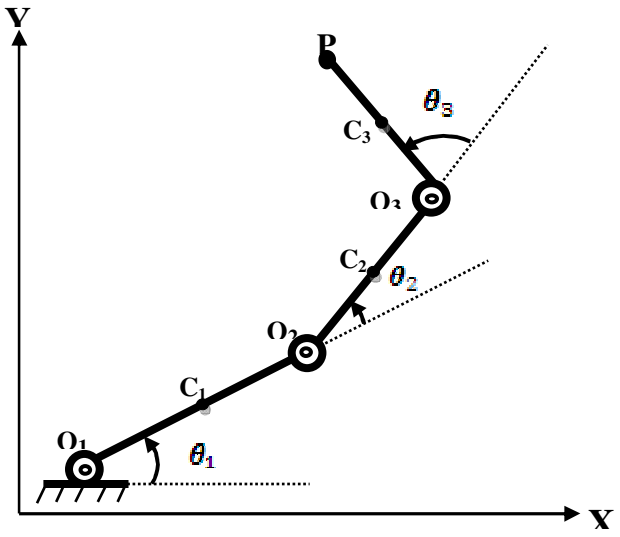



**FIGURE 1:** Three Links Robot Arm

Where $\theta, \dot{\theta} and\ \ddot{\theta} \in R^n$ denote the vectors of joint link positions, velocities and acceleration respectively, I ($\theta$) $\in R^{n\times n}$ denotes the inertia matrix, n denote the number of link, P is the potential energy and $\tau_n$ denoted the torque of n link. Consider the manipulator of Fig. (1), with links designed so that their mass centers, C1, C2, and C3, are located at the midpoints of segments O1O2, O2O3, and O3P, respectively. Moreover, the $ith$ link has a mass ($m_n$ )and a centroidal moment of inertia in a direction normal to the plane of motion ($I_n$ ); while the joints are actuated by motors delivering torques $\tau_1, \tau_2$, and $\tau_3$, the lubricant of the joints producing dissipative torques that we will neglect in this model. Under the assumption that gravity acts in the direction of Y axis. In general, the dynamic model of armature-controlled DC servo motors which shown below, on an $n$- link robot manipulator can be expressed in the following form [17]:

$$\tau_e = K_T i_a \qquad (2)$$

$$\tau_e = J_m \ddot{q}_m + B_m \dot{q}_m + \tau_m \qquad (3)$$

$$v_t = R_a i_a + L_a \frac{di_a}{dt} + K_E \dot{q}_m \qquad (4)$$

Where $\tau_e \in R^n$ is the vector of electromagnetic torque, $K_T \in R^{n\times n}$ is the diagonal matrix of

motor torque constants, $i_a \in R^n$ is the vector of armature currents, $J_m \in R^{n\times n}$ is the diagonal matrix of the moment inertia, $B_m \in R^{n\times n}$ is the diagonal matrix of torsional damping coefficients, $q_m, \dot{q}_m \; and \; \ddot{q}_m \in R^n$ denote the vectors of motor shaft positions, velocities, and accelerations, respectively, $\tau_m \in R^n$ is the vector of load torque, $v_t \in R^n$ is the vector of armature input voltages, $R_a \in R^{n\times n}$ is the diagonal matrix of armature resistance, $L_a \in R^{n\times n}$ is the diagonal matrix of armature inductance, and $K_E \in R^{n\times n}$ is the diagonal matrix of the back electromotive force (EMF) coefficients. In order to apply the dc servo motors for actuating an $n$-link robot manipulator, a relationship between the joint position $\theta$ and the motor-shaft position $q_m$ can be represented as follows [17]:

$$G_r = \frac{q_m}{\theta} = \frac{\tau}{\tau_m} \qquad (5)$$

The governed equation of an n-link robot manipulator including actuator dynamics can be obtained as [1]:

$$I^*(\theta)\ddot{\theta} + D(\theta, \dot{\theta}, \ddot{\theta}) + d = U \qquad (6)$$

Where $U \in R^n$ represents the control effort vector, i.e. armature input voltages,

$$I(\theta) = I_n + I_q(\theta)$$

$$I^* = L_n[I_n + J_n] \qquad (7)$$

$$C(\theta, \dot{\theta}) = \dot{I}(\theta, \dot{\theta}) - \frac{1}{2}\left[\frac{\partial(I\dot{\theta})}{\partial\theta}\right]^T$$

$$D(\theta, \dot{\theta}, \ddot{\theta}) = \{L_n[\dot{I}(\theta, \dot{\theta}) + C(\theta, \dot{\theta}) + B_n] + [I(\theta) + J_n]\}\ddot{\theta} + [L_n\dot{C}(\theta, \dot{\theta}, \ddot{\theta})\dot{\theta} + R_nC(\theta, \dot{\theta})\dot{\theta} + R_nB_n\dot{\theta} + K_{En}\dot{\theta} + L_n\dot{G}(\theta, \dot{\theta}) + R_nG(\theta)] \qquad (8)$$

$$d = L_nI_q(\theta)\ddot{\theta} + L_n\dot{N} + R_nN \qquad (9)$$

Where $G(\theta)$ is gravity vector, N represents the vector of external disturbance $t_l$ and friction term $f(\dot{\theta})$. Then we can re-write Eqn. (6) as:

$$\ddot{\theta} = I^*(\theta)^{-1}[U - (D(\theta, \dot{\theta}, \ddot{\theta}) + d)] \qquad (10)$$

By using method of numerical integration such Euler method for Eqn. (10) we can get position, velocity and acceleration for each link.

## 3. Fuzzy Neural Networks (FNN)

The Architecture of FNN shown in (fig, 2). FNN considered as a special type of neural network [19], this means special connection and node operation. Every layer and every node have its practical meaning because the FNN has the structure which is based on both the fuzzy rules and inference. In the following items each layer shown in (fig, 2) will be described:

1- Input layer
   Input layer transmits the input linguistic variables $x_n$ to the output without changed**.**
2- Hidden layer I
   Membership layer represents the input values with the following Gaussian membership functions [20]:

$$\mu_j^i = \exp\left(-\frac{1}{2}\frac{(x_j - c_{ij})^2}{s^2}\right) \qquad (11)$$

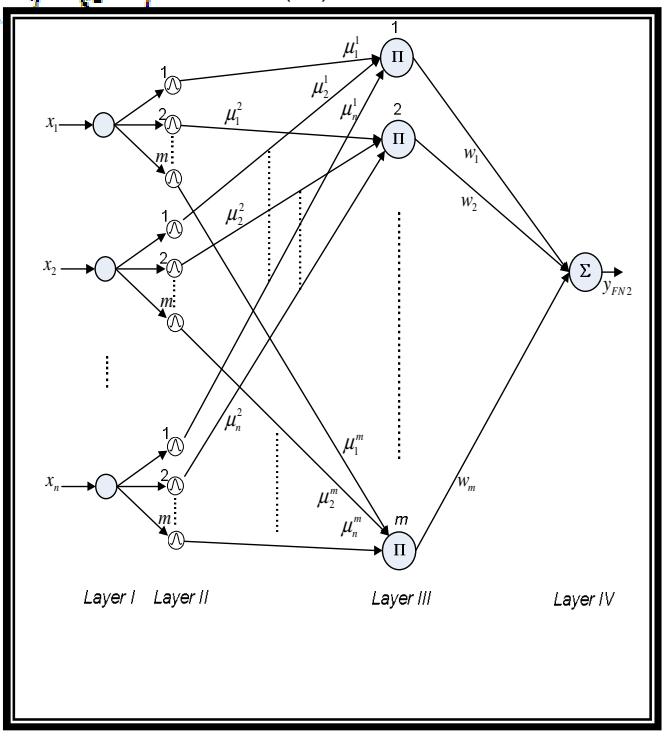



**FIGURE 2:** Architecture of FNN

Where $C_{ij}$ and $S_{ij}$ (i=1, 2, ... , n ; j=1, 2,.., m), respectively, are the mean and standard deviation of the Gaussian function in the j[th] term of the i[th] input linguistic variable $x_n$ to the node of this layer.

3-Hidden layer II

Rule layer implements the fuzzy inference mechanism, and each node in this layer multiplies the input signals and outputs the result of the product. The output of this layer is given as [20]:

$$\emptyset_i = \prod_j^n \mu_j^i \qquad (12)$$

Where $\emptyset_i$ represent the i[th] output of rule layer.

4- Output layer

Layer four is the output layer, and nodes in this layer represent output linguistic variables. Each node $y_o (o = 1, \ldots, N_0)$, which computes the output as [20]:

$$y_o = \sum_i^m w_i^o \emptyset_i \qquad (13)$$

### 3.1 Learning Algorithm FNN Identifier

There are three types of parameters in the fuzzy-neural network can be adapted, in the primes part: the center values $C_{ij}$ and width values $S_{ij}$ of the Gaussian membership functions, whereas, in the consequence part: the consequence weights values $w_i$ . Once the fuzzy-neural network has been initialized, a gradient decent based back-propagation algorithm is employed to adjust

the parameters of the fuzzy-neural network by using the training patterns. The main goal of supervised learning algorithm is to minimize the mean square error function [20]:

$$E = \frac{1}{2}\left(y_{Fnn} - y_p\right)^2 \qquad (14)$$

Where $y_{Fnn}$ is the output of is fuzzy-neural network and $y_p$ is the desired output. The gradient descent algorithm gives the following iterative equations for the parameter values [20]:

$$w_i(k+1) = w_i(k) - \eta_w \frac{\partial E}{\partial w_i} \qquad (15)$$

$$C_{ij}(k+1) = C_{ij}(k) - \eta_c \frac{\partial E}{\partial C_{ij}} \qquad (16)$$

$$S_{ij}(k+1) = S_{ij}(k) - \eta_s \frac{\partial E}{\partial S_{ij}} \qquad (17)$$

Where $\eta$ is the learning rate for each parameter in the system, i=1,2…*n* and j=1,2…*m*. Taking the partial derivative of the error function given by Eqn. (14), we can get the following equations:

$$\frac{\partial E}{\partial w_i} = \left(y_{FNN} \quad y_p\right)\phi_i \qquad (18)$$

$$\frac{\partial E}{\partial C_{ij}} = \left(y_{FNN} - y_p\right)\phi_i w_i \frac{(x_j - C_{ij})}{S_{ij}^2} \qquad (19)$$

$$\frac{\partial E}{\partial S_{ij}} = \left(y_{FNN} - y_p\right)\phi_i w_i \frac{(x_j - C_{ij})^2}{S_{ij}^3} \qquad (20)$$

## 4. Identification

Two representations are available to identify a dynamical system depending on type of the output feedback these are parallel model and Series-parallel model [6]. In this paper the series-parallel identification model is desired. A series-parallel model that is obtained by feeding back the past values of the plant output (rather than the identifier output) as shown in (Fig,3). This implies that in this case the identification model has the form [6]:

$$\hat{y}(k+1) = \begin{bmatrix} y_p(k), y_p(k-1), \ldots, y_p(k-n+1), \\ u(k), u(k-1), \ldots u(k-m+1) \end{bmatrix} \qquad (21)$$

The identifier output is represented by $\hat{y}(k)$ and the plant output is denoted by $y_p(k)$.

## 5. FNN Control

For system control problems, we focus on the adaptive control of dynamic systems using FNN. These algorithms denoted as “Fuzzy Neural Model Reference Controller” (FNMRC) in this type of controllers, back propagation training algorithm is used [16]. There are two distinct approaches for the FNMRC, the first one result in a direct scheme (inverse control) for the controller and the second result in an indirect scheme (forward control), the difference between the two may be shown in figure (4.1) and (4.2).

### 5.1 Learning Algorithm for Indirect FNN Control (Forward)

Indirect control architecture usually requires an identified system model and the controller design is based on the learning algorithm. Our goal is to minimize the following cost function:

$$E_c = \frac{1}{2}\left(e_{c1}^2 + e_{c2}^2 + e_{c3}^2\right) \qquad (22)$$

Where $e_{c1}$, $e_{c2}$ and $e_{c3}$ are errors between reference outputs and robot's link1, link2 and link3

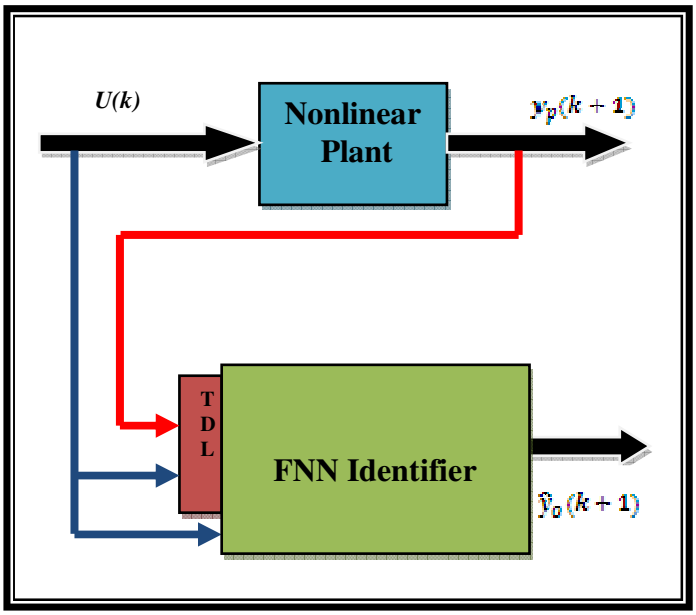



**FIGURE 3:** Series-Parallel identification model

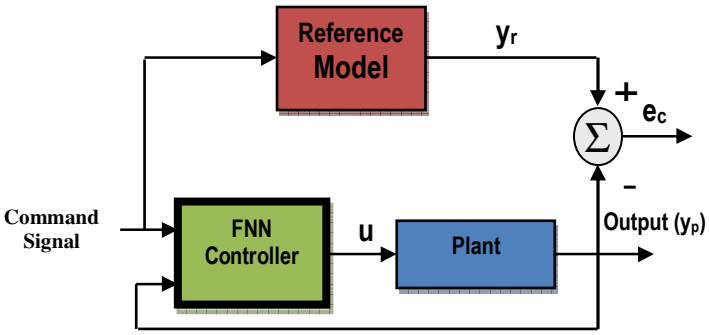



**FIGURE 4.1:** Direct FNN model reference learning controller

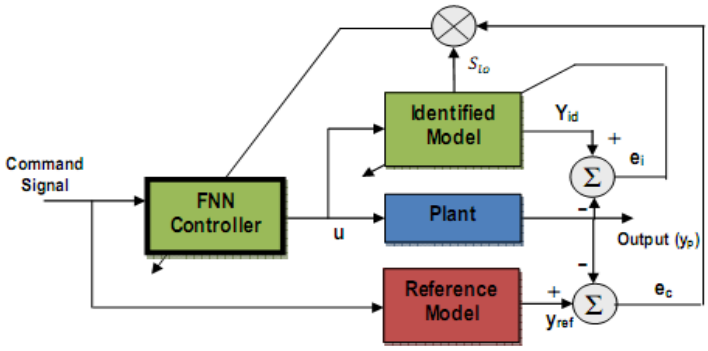



**FIGURE 4.2:** Indirect FNN model reference learning controller

output respectively, then the gradient of error $E_c$ with respect to weights , mean and standard deviation of the Gaussian function are given:

$$\frac{\partial E_c}{\partial w_{cli}} = \emptyset_{ci} \sum_o^3 e_{co} S_{lc} \qquad (23)$$

$$\frac{\partial E_c}{\partial m_{cij}} = A1(e_{c1}S_{11} + e_{c2}S_{12} + e_{c3}S_{13}) + B1(e_{c1}S_{21} + e_{c2}S_{22} + e_{c3}S_{23}) + C1(e_{c1}S_{31} + e_{c2}S_{32} + e_{c3}S_{33}) \qquad (24)$$

$$\frac{\partial E_c}{\partial s_{cij}} = A2(e_{c1}S_{11} + e_{c2}S_{12} + e_{c3}S_{13}) + B2(e_{c1}S_{21} + e_{c2}S_{22} + e_{c3}S_{23}) + C2(e_{c1}S_{31} + e_{c2}S_{32} + e_{c3}S_{33}) \qquad (25)$$

The identifier can provide the system sensitivity $S_{lo}$ and it can be computed by the chain rule:

$$\frac{\partial \hat{y}_{po}}{\partial u_l} = S_{lo} = \frac{\partial \hat{y}_{po}}{\partial \emptyset_i} \frac{\partial \emptyset_i}{\partial mf_{ij}} \frac{\partial mf_{ij}}{\partial u_l}$$

$$S_{lo} = -\sum_{i=1}^{n_i} w_{oi} \emptyset_i \frac{u_l - m_{il}}{s_{il}^2} \qquad (26)$$

Where $u_l$, $y_{po}$ are $l^{th}$ output of FNN controller, $O^{th}$ output of robot plant respectively and where:

$$A1 = w_{c1i} \emptyset_{ci} \frac{x_{cj} - m_{cij}}{s_{cij}^2}$$

$$B1 = w_{c2i} \emptyset_{ci} \frac{x_{cj} - m_{cij}}{s_{cij}^2}$$

$$C1 = w_{c3i} \emptyset_{ci} \frac{x_{cj} - m_{cij}}{s_{cij}^2}$$

$$A2 = w_{c1i} \emptyset_{ci} \frac{(x_{cj} - m_{cij})^2}{s_{cij}^3}$$

$$B2 = w_{c2i} \emptyset_{ci} \frac{(x_{cj} - m_{cij})^2}{s_{cij}^3}$$

$$C2 = w_{c3i} \emptyset_{ci} \frac{(x_{cj} - m_{cij})^2}{s_{cij}^3}$$

## 5.2 Learning Algorithm for Direct FNN Control (Inverse)

The FNN inverse control is shown in the (fig, 5), in which two FNN are present, one for the inverse identification and the other for controller. The basic structure of the inverse controller consists of the controller network only, which is the same to the identifier network in the offline learning. The simple concept of the inverse controller is the controller block that represents the inverse transfer function of the robot plant, so the product result of the two blocks (robot plant and

controller) must equal unity. Hence the output of the robot plant will be equal to desired input of the controller.

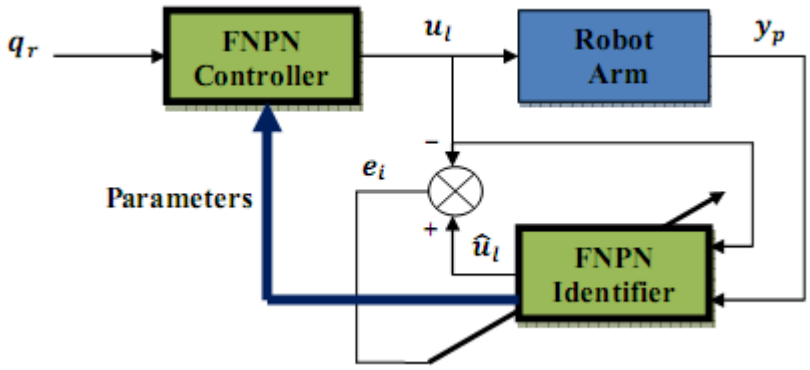



**FIGURE 5:** FNN inverse control

## 6. SIMULATION RESULTS

In the following examples two methods of control presented in above sections are implemented by FNN for three links robot arm. The simulation carried by MATLABT software. The no. of rules and outputs are 50 and 3 respectively, in each method of control. The initial mean and standard of membership function were computed as Eqns. (27) and (28) [21], beside the 0.001values for weights. Following two examples are viewed. The most important parameters that affect the control performance of the robotic system are the external disturbance $t_l(t)$ , the friction term $f(\dot{\theta})$ , and the parameter variation of 3rd link's mass $m_3$ . In all two example simulation, three circumstances including the:

1- Nominal situation ($m_3 = 1$kg and N=0) at beginning.
2- Parameter variation situation occurring at t=15 sec ($m_3 = 2$ kg).
3- Disturbance in addition, friction forces are also considered in this simulation.

Hence,

$$t_l(t) = [5sin(5t) \quad 3sin(5t) \quad sin(5t)]^T$$

$$f(\dot{\theta}) = [20\dot{\theta}_1 + 8sign(\dot{\theta}_1) \quad 10\dot{\theta}_2 + 4sign(\dot{\theta}_2) \quad 5\dot{\theta}_3 + 2sign(\dot{\theta}_3)]^T$$

$$N = t_l(t) + f(\dot{\theta})$$

$$m_{ij} = X_{n\,max} - (i-1)\frac{X_{n\,max} - X_{n\,min}}{N_i - 1} \qquad (27)$$

$$s_{ij} = 2\frac{X_{n\,max} - X_{n\,min}}{N_i - 1} \qquad (28)$$

Where $X_{n\,max}$ , $X_{n\,min}$ are the predetermined maximal and minimal bounds of $n^{th}$ input to FNN.

### 6.1 Example 1

In this example the forward control is implemented by FNN in each one the forward identifier is used to calculate the plant sensitivity, the initial parameters of identifier will take from final values proceed in the offline learning. The eighteen inputs are fed to FNN controller, the learning rate of weights, mean and standard are $\eta_{cw} = .01$, $\eta_{cm} = .0012$ and $\eta_{cs} = .005$ respectively . figures (6.a) to (6.f) are shown the FNN forward control position response and mean square error for link1, link2 and link3 respectively for 100 epochs.

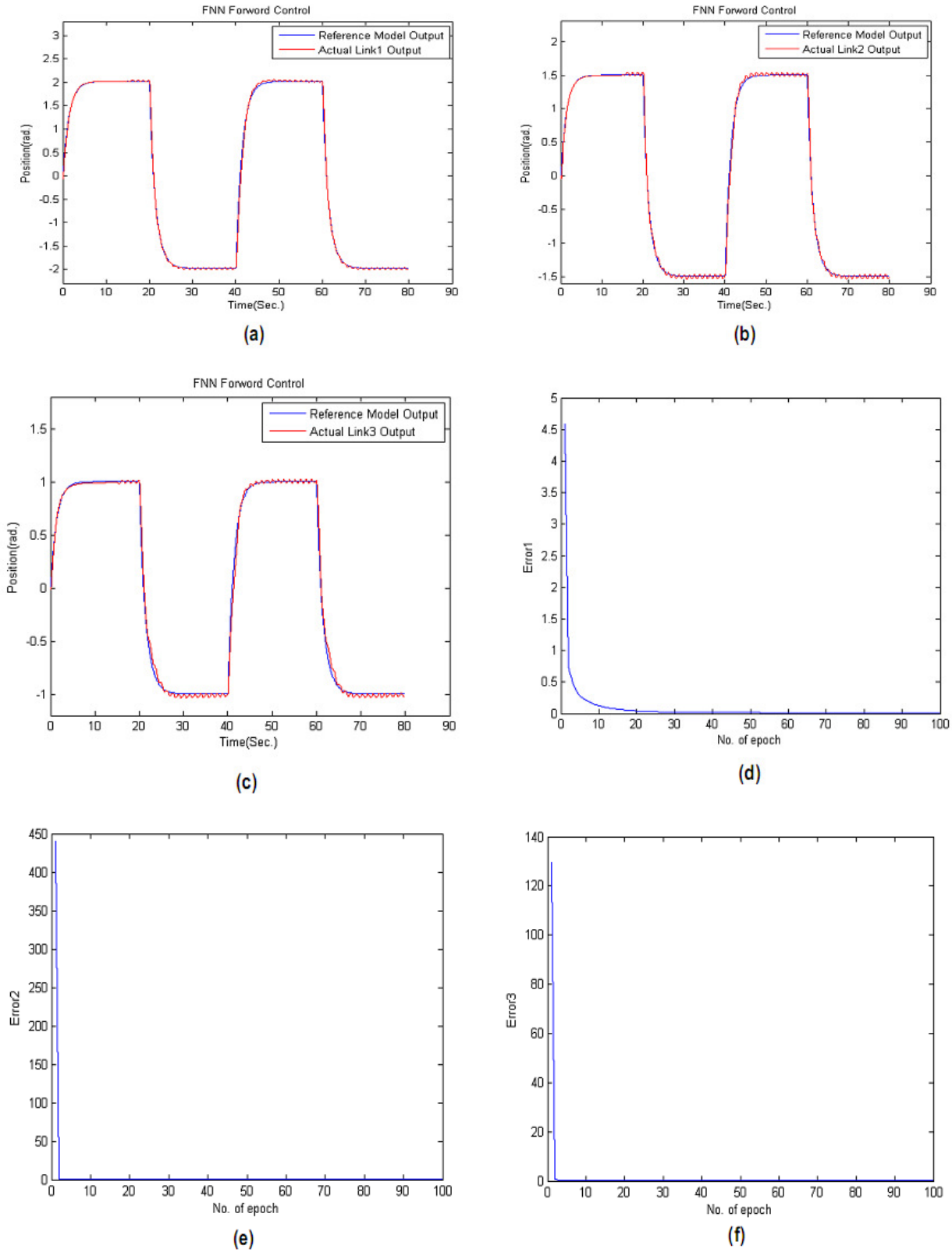



**FIGURE 6:** FNN forward control simulation results of position response and mean square error for Link1, Link2 and Link3 (a)-(f)

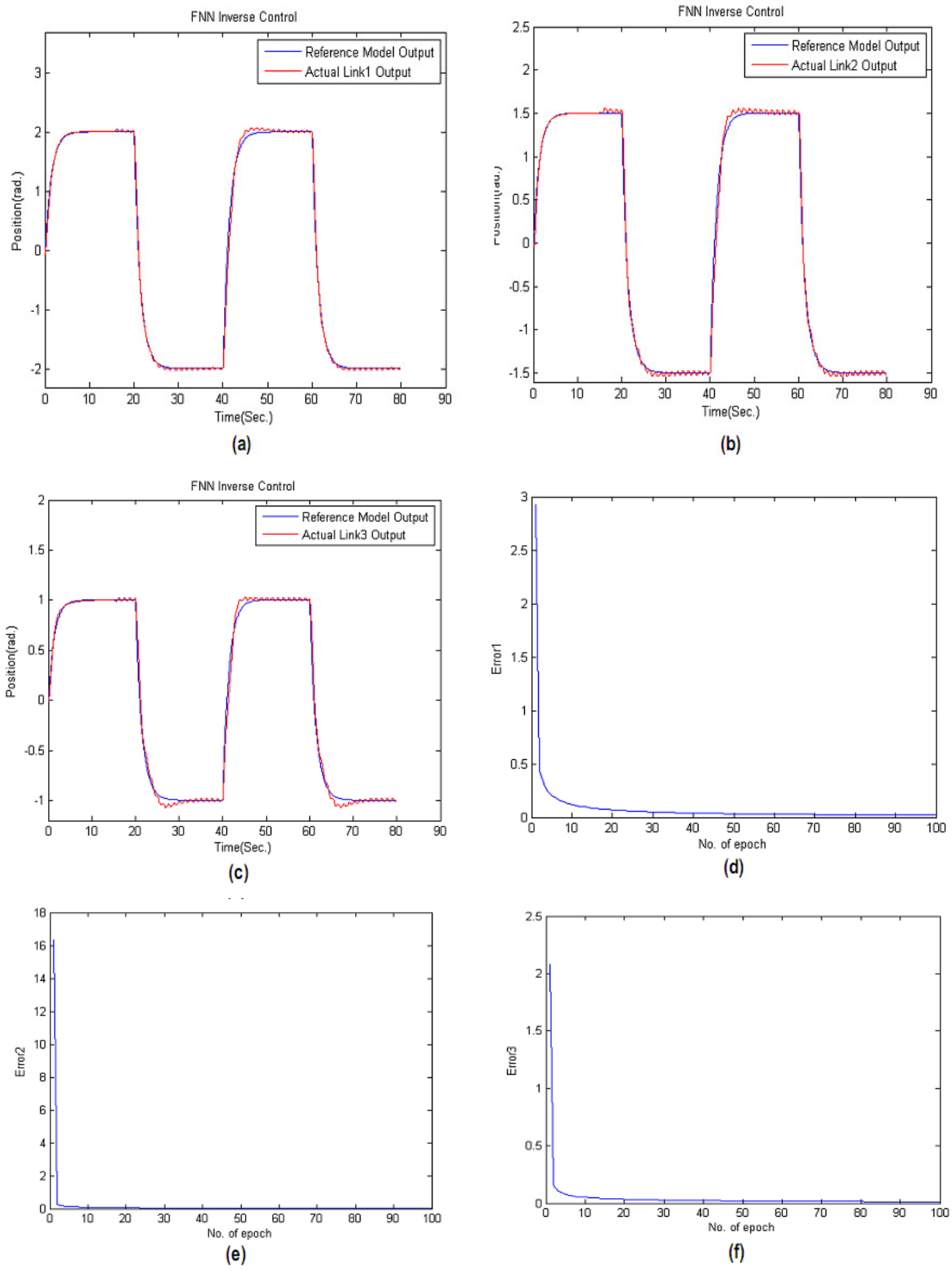



**FIGURE 7:** FNN inverse control simulation results of position response and mean square error for Link1, Link2 and Link3 (a)-(f)

### 6.2 Example 2

The FNN inverse control presented in this example, the structure of controller same the structure of the inverse identifier which only changes the $y_p(k+1)$ input to inverse identifier by reference input $q_r(k+1)$ to inverse controller. Inverse identifier used here so that the parameters generated in offline learning considered initial parameters to online inverse identifier and inverse controller. The eighteen inputs are fed to FNN controller, the learning rate of weights, mean and standard are $\eta_{cw} = 0.02$, $\eta_{cm} = 0.0012$ and $\eta_{cs} = 0.005$ respectively. Figures (7.a) to (7.f) are shown the FNN inverse control position response and mean square error for link1, link2 and link3 respectively for 100 epochs.

## 7. CONCLUSION

In this paper use FNN for identification and control for dynamic nonlinear systems such three links robot arm. From the previous examples we conclude that the FNN is powerful for identify and control nonlinear system, in example1 use indirect control with online forward identification and the gradient in mean square error is done and in example2 use the direct control technique with online inverse identification after use the parameters are get from offline inverse identification to use in online work. Table (1) shows the gradient mean square error for both examples for each link and the mean square error for each link when we applied the traditional PD control (Proportion and Derivative control) on them in order to compare the values of MSE among example1, example2 (they applied by FNN control) and PD control, the main difference between FNN control and traditional PD control is a PD control can't adapt its gains ($k_p$, $k_d$) when some disturbance insert to plant in otherwise the FNN control can adapt its parameters ($w_c$, $m_{cij}$ and $s_{cij}$) by online learning algorithm.
For future work the control technique by FNN without identification will study to reduce load of computation.

| Example | Link1 | Link2 | Link3 |
|---|---|---|---|
| 1 | 0.0093 | 0.0093 | 0.0085 |
| 2 | 0.018 | 0.0181 | 0.0125 |
| PD control | 0.0095 | 0.009 | 0.0098 |

**TABLE 1**: Mean square error

## 8. REFERENCES

[1] Rong-jong Wa and Po-Chen Chen,"Robust Neural-Fuzzy-Network Control for Robot Manipulator Including Actuator Dynamics", IEEE Trans. Indst. Elect. vol. 53, no. 4, Aug. 2006.

[2] S. J. Huang and J. S. Lee, "A stable self-organizing fuzzy controller for robotic motion control," IEEE Trans. Ind. Electron., vol. 47, no. 2, pp. 421–428, Apr. 2000.

[3] B. K. Yoo and W. C. Ham, "Adaptive control of robot manipulator using fuzzy compensator," IEEE Trans. Fuzzy Syst., vol. 8, no. 2, pp. 186–199, Apr. 2000.

[4] Y. C. Chang, “Neural network-based H-infinite tracking control for robotic systems,” Proc. Inst. Electr . Eng.—Control Theory Appl., vol. 147, no. 3, pp. 303–311, May 2000.

[5] Y. H. Kim and F. L. Lewis, “Optimal design of CMAC neural-network controller for robot manipulators,” IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 30, no. 1, pp. 22–31, Feb. 2000.

[6] K. S. Narendra and K. Parthasarathy, “Identification and control of dynamical system using neural networks,” IEEE Trans. Neural Networks, vol. 1, pp. 4–27, Jan. 1990.

[7] K. Funahashi and Y. Nakamura, Approximation of dynamical systems by continuous-time recurrent neural network,” Neural Networks, vol.6, pp. 801–806, 1993.

[8] L. Jin, P. N. Nikiforuk, and M. Gupta, “Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks,” IEEE Trans. Automat. Contr., vol. 40, pp. 1266–1270, July 1995.

[9] C. C. Ku and K. Y. Lee, “Diagonal recurrent neural networks for dynamic systems control,” IEEE Trans. Neural Networks, vol. 6, pp.144–156, Jan. 1995.

[10] Ching-Hung Lee and Ching-Cheng Teng, “Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks”, IEEE Trans. Fuzzy system, vol. 8, no. 4, Aug. 2000

[11] C. T. Lin and C. S. G. Lee, “Neural-network-based fuzzy logic control and decision system,” IEEE Trans. Computer. , vol. 40, pp. 1320–1336, Dec. 1991.

[12] B. S. Chen, H. J. Uang, and C. S. Tseng, “Robust tracking enhancement of robot systems including motor dynamics: A fuzzy-based dynamic game approach,” IEEE Trans. Fuzzy Syst., vol. 6, no. 4, pp. 538–552,Nov. 1998.

[13] C. Ishii, T. Shen, and K. Tamura, “Robust model following control for a robot manipulator,” Proc. Inst. Electr. Eng.—Control Theory Appl., vol. 144, no. 1, pp. 53–60, Jan. 1997.

[14] R. J. Schilling, Fundamentals of Robotics: Analysis and Control. Hoboken, NJ: Prentice-Hall, 1998.

[15] Mark W.Spong, Seth H., M. Vidyasagar, “Robot Modeling and Control”, John Wiley and Sons, INC., 2001

[16] Abdul Baqi, J.N., "Neuro-Fuzzy Control of robot Arm" MSC. Thesis, University of Basrah, College of Engineering, Feb. 2004.

[17] Rong-Jong Wai, P. C. Chen, Chun-Yen Tu, “Robust Neural-fuzzy-network Control for Rigid-link Electrically Driven Robot Manipulator”, IEEE Trans. Ind. Electron., 30th annual conference, pp. 1328–1349, Nov. 2004.

[18] Jorge Angels, ”Fundamentals of robotic mechanical systems: theory, methods and Algorithms”, Springer, 2003.

[19] C. T. Leondes, "Fuzzy logic and expert systems applications", Academic Press, 1998.

[20] Rong-Jong Wai, Chia-Chin Chu," Robust Petri Fuzzy-Neural-Network Control for Linear Induction Motor Drive", IEEE trans. on Ind. Elect. , Vol. 54, No. 1, pp. 177-189, Feb. 2007.

[21] Rong-Jong Wai, Chia-Ming Liu," Design of Dynamic Petri Recurrent Fuzzy Neural Network and Its Application to Path-Tracking Control of Nonholonomic Mobile Robot", IEEE transactions on Ind. Elec., Vol. 56, NO. 7, pp.2667-2683, July 2009.

# Model Based Hierarchical and Distributed Control of Discrete Event Robotic Systems Using Extended Petri Nets

**Gen'ichi Yasuda** yasuda.genichi@gmail.com
*Department of Human and Computer Intelligence*
*Nagasaki Institute of Applied Science*
*Nagasaki, 851-0193, Japan*

**Abstract**

This paper presents a method of model based hierarchical and distributed control for discrete event robotic systems. Based on the hierarchical approach, the Petri net is translated into the detailed Petri net from the highest conceptual level to the lowest local control level. A coordination algorithm through communication between the coordinator and the local controllers, is specified and the overall distributed control system is implemented using multithread programming. By the proposed method, modeling, simulation and control of large and complex robotic systems can be performed consistently using extended Petri nets.

**Keywords:** Robotic Systems, Model Based Design, Distributed Control, Petri Nets, Implementation.

## 1. INTRODUCTION

Large and complex robotic systems such as flexible manufacturing systems are commonly decomposed into a hierarchy of abstraction levels: planning, scheduling, coordination and local control. Each level operates on a certain time horizon. The planning level determines at which time each product will be introduced in the system. The scheduling level produces a sequence of times for the execution of each operation on each machine or a total ordering of all the operations. The coordination level updates the state representation of the system in real time, supervises it and makes real-time decisions. The local control level implements the real-time control of machines and devices etc., interacting directly with the sensors and actuators. All the emergency procedures are implemented at this level, so real-time constraints may be very hard. At each level, any modeling has to be based on the concepts of discrete events and states, where an event corresponds to a state change [1].

A manufacturing cell is an elementary manufacturing system consisting of some flexible machines (machine tools, assembly devices, or any complex devices dedicated to complex manufacturing operations), some local storage facilities for tools and parts and some handling devices such as robots in order to transfer parts and tools. Elementary manufacturing cells are called workstations. At the local control level of manufacturing cells many different kinds of machines can be controlled, and specific languages for different application domains are provided; for example, block diagrams for continuous process control and special purpose languages for CNC or robot programming. For common sequential control of such machines, special purpose real-time computers named Programmable Logic Controllers (PLCs) are used. PLCs are replacements for relays, but they incorporate many additional and complex functions, such as supervisory and alarm functions and start-up and shut-down operations, approaching the functionalities of general purpose process computers. The most frequent programming languages are based on ladder or logic diagrams and boolean algebra. However, when the local control is of greater complexity, the above kinds of languages may not be well adapted. The development of industrial techniques makes sequential control for manufacturing cells more large and complicated one, in which some subsystems operate concurrently and cooperatively. Conventional representation methods based on flowcharts, time diagrams, state machine diagrams, etc. can not be used for such systems [2].

To realize control systems for complex robotic systems such as flexible manufacturing cells, it is necessary to provide effective tools for describing process specifications and developing control algorithms in a clear and consistent manner. In the area of real-time control of discrete event systems the main problems that the system designer has to deal with are concurrency, synchronization, and resource sharing problems. For this class of problems, Petri nets have intrinsic favorable qualities and it is very easy to model sequences, choices between alternatives, rendez-vous and concurrent activities by means of Petri nets [2]. When using Petri nets, events are associated with transitions. Activities are associated to the firing of transitions and to the marking of places which represents the state of the system. The network model can describe the execution order of sequential and parallel tasks directly without ambiguity. Moreover, the formalism allowing a validation of the main properties of the Petri net control structure (liveness, boundedness, etc.) guarantees that the control system will not fall immediately in a deadlocked situation. In the field of flexible manufacturing cells, the last aspect is essential because the sequences of control are complex and change very often [3,4]. In addition to its graphic representation differentiating events and states, Petri nets allows the modeling of true parallelism and the possibility of progressive modeling by using stepwise refinements or modular composition. Libraries of well-tested subnets allow components reusability leading to significant reductions in the modeling effort. The possibility of progressive modeling is absolutely necessary for large and complex systems, because the refinement mechanism allows the building of hierarchically structured net models. Furthermore, a real-time implementation of the Petri net specification by software called a token player can avoid implementation errors, because the specification is directly executed by the token player and the implementation of these control sequences preserves the properties of the model. In this approach, the Petri net model is stored in a database and the token player updates the state of the database according to the operation rules of the model. For control purposes, this solution is very well suited to the need of flexibility, because, when the control sequences change, only the database needs to be changed. Some techniques derived from Petri nets have been successfully introduced as an effective tool for describing control specifications and realizing the control in a uniform manner. However, in the field of flexible manufacturing cells, the network model becomes complicated and it lacks of readability and comprehensibility [5]. Therefore, the flexibility and expandability are not satisfactory in order to deal with the specification change of the control system. Despite the advantages offered by Petri nets, the synthesis, correction, updating, etc. of the system model and programming of the controllers are not simple tasks.

The author proposes a Petri net based specification and real-time control method for large and complex robotic systems. Based on the hierarchical and distributed structure of the system, the specification procedure is a top-down approach from the conceptual level to the detailed level such that the macro representation of the system is broken down to generate the detailed Petri nets at the local machine control level. Then the Petri nets are decomposed and assigned to the machine controllers to perform distributed control using Petri net based multitask processing. An algorithm is proposed for coordination of machine controllers such that the behavior of the distributed control system is the same as that of the original system. By the proposed method, modeling, simulation and control of large and complex robotic systems can be performed consistently using Petri nets.

## 2. DISCRETE EVENT SYSTEM MODELING USING EXTENDED PETRI NETS

From the viewpoint of discrete event process control, an overall robotic system can be viewed as a set of distinct events and conditions mutually interrelated in a complex form. An event is the start or end of an activity of a process executed by a subsystem. A condition is a state in the process such as operation mode. Considering the nature of discrete event systems which are characterized by the occurrence of events and changing conditions, the condition-event net based specification method has been investigated. The Petri net is one of the effective means to represent condition-event systems. The specification method is a graphical model used as a tool to identify types of events, conditions, and their mutual interrelation.

In the condition-event systems, bumping occurs when despite the holding of a condition, the preceding event occurs. This can result in the multiple holding of that condition and the Petri net is said to be unsafe. When we consider not only the modeling but also the actual well-designed control of robotic systems, the guarantee of safeness and the capability to represent input and output signals from and to the machines are required. Thus, the fundamental Petri net which is called the Place/Transition net should be modified and extended in order to represent the activity contents and control strategies for the system control in detail.

The extended Petri net consists of the following six elements: (1) Place, (2) Transition, (3) Directed arc, (4) Token, (5) Gate arc, (6) Output signal arc [6]. A place represents a condition of a system element or action. A transition represents an event of the system. A directed arc connects a place to a transition and vice versa, and its direction shows the input and output relation between them, thus places and transitions are alternately connected using directed arcs. A token is placed in a place to indicate that the condition corresponding to the place is holding. A gate arc connects a transition with a signal source, and it either permits or inhibits the occurrence of the event which corresponds to the connected transition. Gate arcs are classified as permissive or inhibitive, and internal or external. An output signal arc sends the signal from a place to an external machine. The axioms for the extended Petri net execution are as follows. A transition is firable or enabled if and only if it satisfies all the following firability conditions:

(1) It does not have any output place filled with a token.
(2) It does not have any empty input place.
(3) It does not have any internal permissive arc signaling 0.
(4) It does not have any internal inhibitive arc signaling 1.

IAn enabled transition fires when it does not have any external permissive arc signaling 0 nor any external inhibitive arc signaling 1. It is considered that the time required for firing is infinitely small. The firing of a transition removes a token from each input place and put a token in each output place connected to the transition. In any initial marking, there must not exist more than one token in a place. According to these rules, the number of tokens in a place never exceeds one, thus the Petri net is essentially a safe graph; the system is free from the bumping phenomenon. The assignment of tokens into the places of a Petri net is called marking and it represents the system state. The behavior of discrete event systems can be described in terms of system states and their changes. In order to simulate the dynamic behavior of a system, a state or marking in a Petri net is changed according to the transition firability conditions. Figure 1 shows the place and gate variables used to decide the firing of a transition.

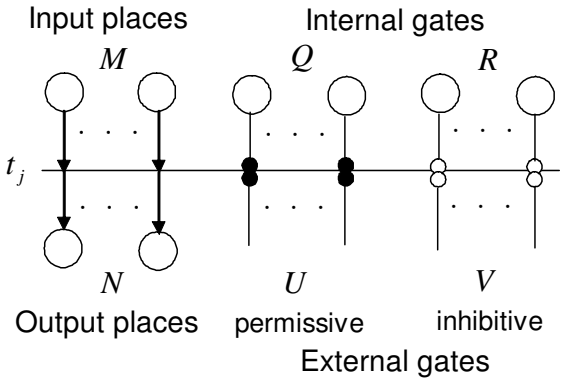



**FIGURE 1:** Place and gate variables associated with transition firing.

Formally, the firability condition and external gate condition of a transition $j$ are defined using the logical variables in Figure 1 as follows:

$$t_j(k) = \bigcap_{m=1}^{M} p_{j,m}^{I}(k) \wedge \bigcap_{n=1}^{N} \overline{p_{j,n}^{O}(k)} \wedge \bigcap_{q=1}^{Q} g_{j,q}^{IP}(k) \wedge \bigcap_{r=1}^{R} \overline{g_{j,r}^{II}(k)} \quad (1)$$

$$g_j^E(k) = \bigcap_{u=1}^{U} g_{j,u}^{EP}(k) \wedge \bigcap_{v=1}^{V} \overline{g_{j,v}^{EI}(k)} \quad (2)$$

where,

| | |
|---|---|
| $M$ | : set of input places of transition $j$ |
| $p_{j,m}^{I}(k)$ | : state of input place $m$ of transition $j$ at time sequence $k$ |
| $N$ | : set of output places of transition $j$ |
| $p_{j,n}^{O}(k)$ | : state of output place $n$ of transition $j$ at time sequence $k$ |
| $Q$ | : set of internal permissive gate signals of transition $j$ |
| $g_{j,q}^{IP}(k)$ | : internal permissive gate signal variable $q$ of transition $j$ at time sequence $k$ |
| $R$ | :set of internal inhibitive gate signals of transition $j$ |
| $g_{j,r}^{II}(k)$ | : internal inhibitive gate signal variable $r$ of transition $j$ at time sequence $k$ |
| $U$ | : set of external permissive gate signals of transition $j$ |
| $g_{j,u}^{EP}(k)$ | : external permissive gate signal variable $u$ of transition $j$ at time sequence $k$ |
| $V$ | : set of external inhibitive gate signals of transition $j$ |
| $g_{j,v}^{EI}(k)$ | : external inhibitive gate signal variable $v$ of transition $j$ at time sequence $k$ |

The state (marking) change, that is, the addition or removal of a token of an input place and an output place, is defined as follows:

$$p_{j,m}^{I}(k+1) = p_{j,m}^{I}(k) \wedge \overline{(t_j(k) \wedge g_j^E(k))} \quad (3)$$

$$p_{j,n}^{O}(k+1) = p_{j,n}^{O}(k) \vee (t_j(k) \wedge g_j^E(k)) \quad (4)$$

or,

$$p_{j,m}^{I}(k+1) = RST(t_j(k) \wedge g_j^E(k)) \quad (3')$$

$$p_{j,n}^{O}(k+1) = SET(t_j(k) \wedge g_j^E(k)) \quad (4')$$

where, $y = RST(x)$ , or $SET(x)$ denotes that if $x = 1$ then $y = 0$ (reset), or $y = 1$ (set), respectively.

Figure 2 shows Petri net representations of an elementary machine control process with machine activity. For the extended Petri net, it is proved that, with the reversion of the existence of token in a place as well as the directions of input and output arcs of the place, the firability condition of the transitions of the net is not changed. So, two Petri net representations in Figure 2(a) and (b) are equivalent. The token in the place "Machine" indicates that the machine is free (Figure 2(a)) or the machine is operating (Figure 2(b)). Furthermore Figure 2(c) shows a hierarchical representation of a task; the place "Machine task A" is the macro representation of the detailed representation composed of the subtasks 1 and 2. A task executed by a robot or an intelligent machine tool can be seen as some connection of more detailed subtasks. For example, transferring an object from a start position to a goal position is a sequence of the following subtasks; moving the hand to the start position, grasping the object, moving to the goal position, and putting it on the specified place.

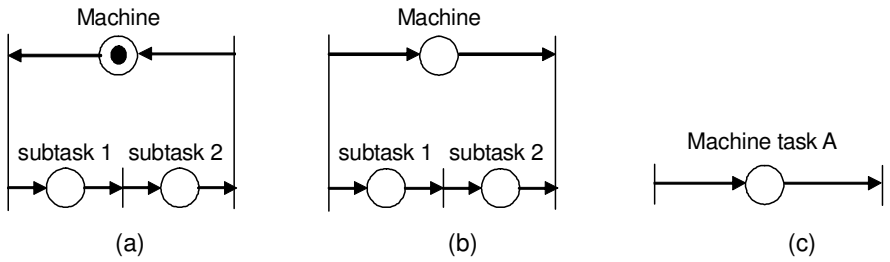



**FIGURE 2:** Petri net representation of elementary process with machine activity;
(a) with loop net, (b) with source and sink transitions, (c) macro representation of robotic task.

For the actual machine control, output signal arcs are connected from associated places to the machine, and external gate arcs are connected from the machine to the transitions to coordinate the succeeding subtasks as shown in Figure 3. When a token enters a place that represents a subtask, the machine defined by the machine code is informed to execute a subtask with some control data that are defined as the place parameters.

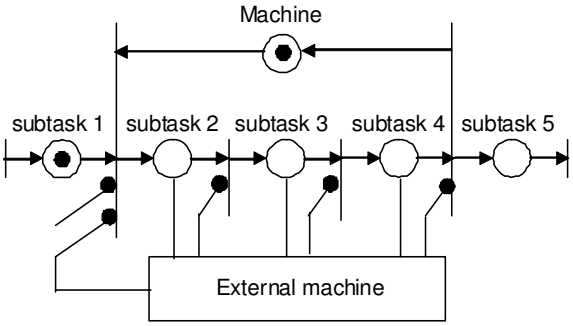



**FIGURE 3:** Petri net representation of manufacturing process control with output arcs and external gates.

Two important system controls in discrete event robotic systems are (a) parallel control, and (b) selective control. Figure 4 shows the Petri net representation of parallel control. In Figure 4 the parallel activities begin at the firing of transition t1 and end with transition t4. Transitions t2 and t3 are said to be concurrent if they are causally independent. The subclass of Petri nets where each place has exactly one incoming arc and exactly one outgoing arc is known as marked graphs.

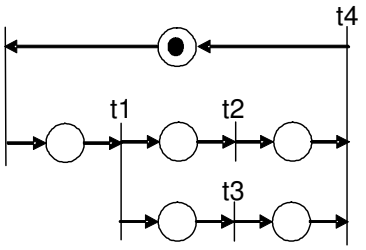



**FIGURE 4:** Petri net representation of parallel control of robotic task.

The subclass of Petri nets where each transition has exactly one incoming arc and exactly one outgoing arc is known as state machines. Any finite state machine and its state diagram can be modeled with a state machine. The structure of the place having two or more output transitions is referred to as a conflict. State machines allow the representation of conflicts, decisions, or choices, but not the synchronization of parallel activities. When two or more transitions are firable only one transition should fire using gate arcs or some arbitration rule, such as logical priority or specified order. Figure 5 shows an example Petri net with corresponding arbiters.

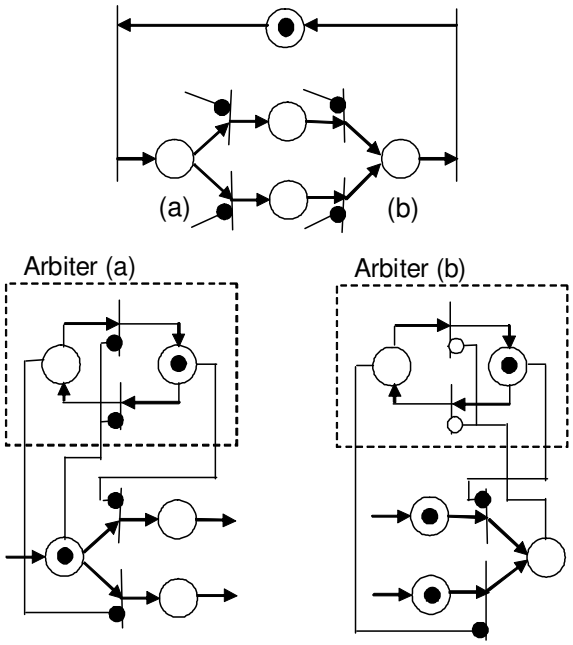



**FIGURE 5:** Petri net representation of selective control of robotic task with arbiters.

The Petri net described in detail with the above procedures can be used as a program for the system control, while features of discrete event systems such as ordering, parallelism, asynchronism, concurrency and conflict can be concretely described through the extended Petri net. The extended Petri net is a tool for the study of condition-event systems and used to model condition-event systems through its graphical representation. Analysis of the net reveals important information about the structure and the dynamic behavior of the modeled condition-event system. This information can then be used to evaluate the system and suggest improvements or changes. A Petri net model includes control algorithms, and is used to control the real robotic process by coincidence of the behavior of the real system with the Petri net model.

## 3. MODEL BASED HIERARCHICAL AND DISTRIBUTED CONTROL

A specification procedure for discrete event robotic systems based on Petri nets is as follows. First, the conceptual level activities of the system are defined through a Petri net model considering the task specification corresponding to the aggregate discrete event process. The places which represent the subtasks indicated as the task specification are connected by arcs via transitions in the specified order corresponding to the flow of subtasks and a workpiece. The places representing the existence of machines used for the subtasks are also added to connect transitions which correspond to the beginning and ending of their subtasks. Then, the detailed Petri nets describing the subtasks are deduced based on activity specification and required control strategies for resources such as robots and other intelligent machines. At each step of detailed specification, places of the Petri net are substituted by a subnet in a manner which maintains the structural properties [7].

Since in the large and complex systems, the controllers are geographically distributed according to their physical (hardware) structure, it is natural to implement a hierarchical and distributed control system, where one controller is allocated to each control layer or block. For robotic systems composed of robots, machine tools, and conveyors, an example structure of hierarchical and distributed control is composed of one station or system controller and three machine or local controllers as shown in Figure 6, although each robot may be controlled by one robot controller. The detailed Petri net is decomposed into subnets, which are assigned to local controllers.

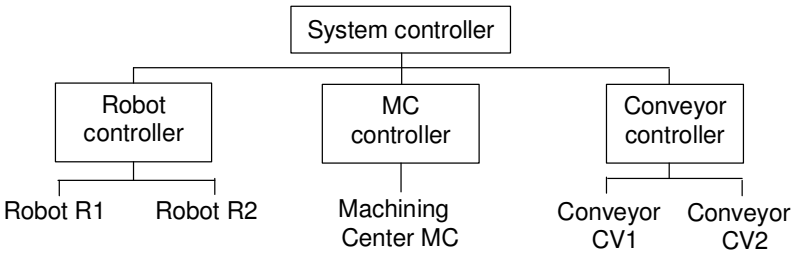



**FIGURE 6:** Example of distributed control system.

In the decomposition procedure, a transition may be divided and distributed into different machine controllers as shown in Figure 7. The machine controllers should be coordinated so that these transitions fire simultaneously, that is, the aggregate behavior of decomposed subnets should be the same as that of the original Petri net. Decomposed transitions are called global transitions, and other transitions are called local transitions.

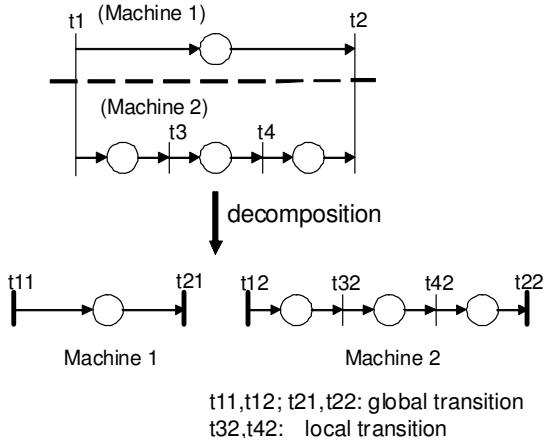



**FIGURE 7:** Decomposition of transitions.

By the Petri net model, the state of the discrete event system is represented as the marking of tokens, and firing of any transition brings about change to the next state. So the firability condition and state (marking) change before decomposition should be the same as those after decomposition. If transition $j$ is divided into $s$ transitions $j1$, $j2$, , , $js$, as shown in Figure 8, the firability condition of a transition after decomposition is described as follows:

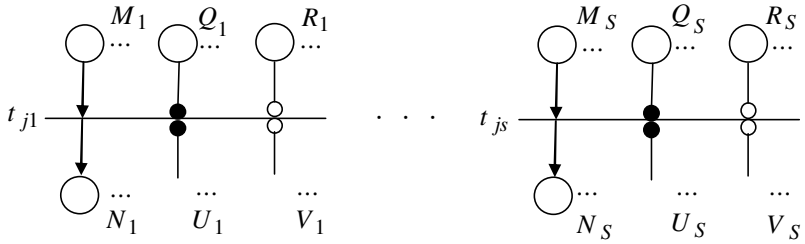



**FIGURE 8:** Place and gate variables after decomposition of transition.

$$t_{jsub}(k) = \bigcap_{m=1}^{Msub} p^{I}_{jsub,m}(k) \wedge \bigcap_{n=1}^{Nsub} \overline{p^{O}_{jsub,n}(k)} \wedge \bigcap_{q=1}^{Qsub} g^{IP}_{jsub,q}(k) \wedge \bigcap_{r=1}^{Rsub} \overline{g^{II}_{jsub,r}(k)} \quad (5)$$

$$g^{E}_{jsub}(k) = \bigcap_{u=1}^{Usub} g^{EP}_{jsub,u}(k) \wedge \bigcap_{v=1}^{Vsub} \overline{g^{EI}_{jsub,v}(k)} \tag{6}$$

From Eq. (1) and Eq.(5),

$$t_j(k) = \bigcap_{sub=1}^{S} t_{jsub}\ (k) \tag{7}$$

From Eq. (2) and Eq. (6),

$$g^{E}_{j}(k) = \bigcap_{sub=1}^{S} g^{E}_{jsub}(k) \tag{8}$$

where,

$S$ : total number of subnets
$Msub$ : set of input places of transition $jsub$ of subnet $sub$
$p^{I}_{jsub,m}(k)$ : state of input place $m$ of transition $jsub$ of subnet $sub$ at time sequence $k$
$Nsub$ : set of output places of transition $jsub$ of subnet $sub$
$p^{O}_{jsub,n}(k)$ : state of output place $n$ of transition $jsub$ of subnet $sub$ at time sequence $k$
$Qsub$ : set of internal permissive gate signals of transition $jsub$ of subnet $sub$
$Rsub$ : set of internal inhibitive gate signals of transition $jsub$ of subnet $sub$
$Usub$ : set of external permissive gate signals of transition $jsub$ of subnet $sub$
$Vsub$ : set of external inhibitive gate signals of transition $jsub$ of subnet $sub$

The addition or removal of a token of a place connected to a decomposed transition is described as follows:

$$p^{I}_{jsub,m}(k+1) = p^{I}_{jsub,m}(k) \wedge \overline{(t_j(k) \wedge g^{E}_{j}(k))} \tag{9}$$

$$p^{O}_{jsub,n}(k+1) = p^{O}_{jsub,n}(k) \vee (t_j(k) \wedge g^{E}_{j}(k)) \tag{10}$$

Consequently it is proved that the firability condition of the original transition is equal to AND operation of firability conditions of decomposed transitions. If and only if all of the decomposed transitions are firable, then the global transitions are firable. To utilize the above results, the coordinator program has been introduced to coordinate the decomposed subnets so that the aggregate behavior of decomposed subnets is the same as that of the original Petri net. In case that a transition in conflict with other transitions is decomposed, these transitions should be coordinated by the system controller. If arbitrations of the transitions are performed independently in separate subnets, the results may be inconsistent with the original rule of arbitration. Therefore the transitions should be arbitrated together as a group. On the other hand, arbitration of local transitions in conflict is performed by local machine controllers. In the hierarchical and distributed control system composed of one system controller and several machine controllers, the conceptual Petri net model is allocated to the Petri net based controller for management of the overall system, while the detailed Petri net models are allocated to the Petri net based controllers in the machine controllers. The control of the overall system is achieved by coordinating these Petri net based controllers. System coordination is performed through communication between the coordinator in the system controller and the Petri net based controllers in the machine controllers [8].

## 4. IMPLEMENTATION OF CONTROL SYSTEM

The example robotic system has one or two robots, one machining center, and two conveyors, where one is for carrying in and the other one is for carrying out, as shown in Figure 9. The main execution of the system is indicated as the following task specification:

(1) A workpiece is carried in by the conveyor CV1.
(2) The workpiece is loaded to the machining center MC by the Robot.
(3) The workpiece is processed by the machining center MC.
(4) The workpiece is unloaded to the conveyor CV2 by the Robot.
(5) The workpiece is carried out by the conveyor CV2.

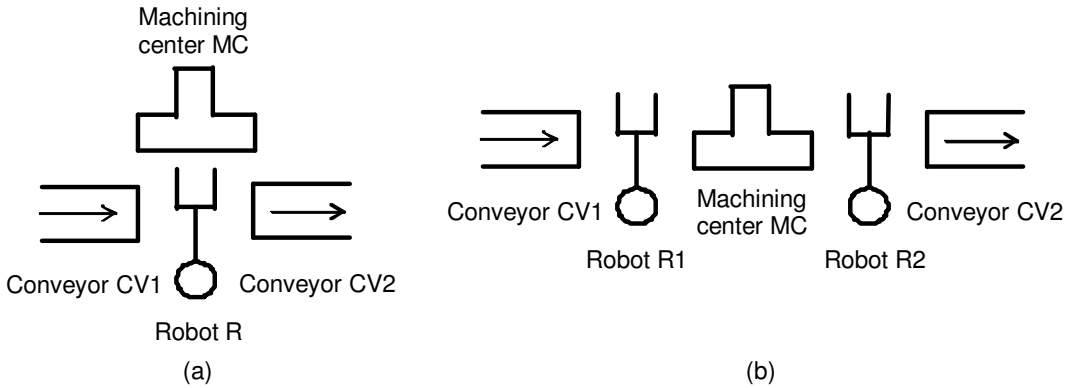



**FIGURE 9:** Example of robotic system; (a) with one robot, (b) with two robots.

At the conceptual level the discrete event processes of the robotic system are represented as shown in Figure 10 for the systems with one robot and with two robots, respectively. In this step, if necessary, control conditions such as the capacity of the system between the respective subtasks must be connected to regulate the execution of the manufacturing process. For the system with one robot, the place "Robot" has two input transitions and two output transitions, but these transitions are not firable at the same time because a token is in the place "MC", so they are not in conflict for firing. The firing of only one of these transitions is permitted using external gate arcs.

Next, each place representing a subtask at the conceptual level is translated into a detailed subnet. Figure 11 shows the detailed Petri net representations of subtasks: loading, processing and unloading in Figure 10. While a Petri net at the conceptual level represents a flow of task-level commands, they represent sequences of motion-level operations. For registered subtasks, the translation procedure is automatically executed by software.

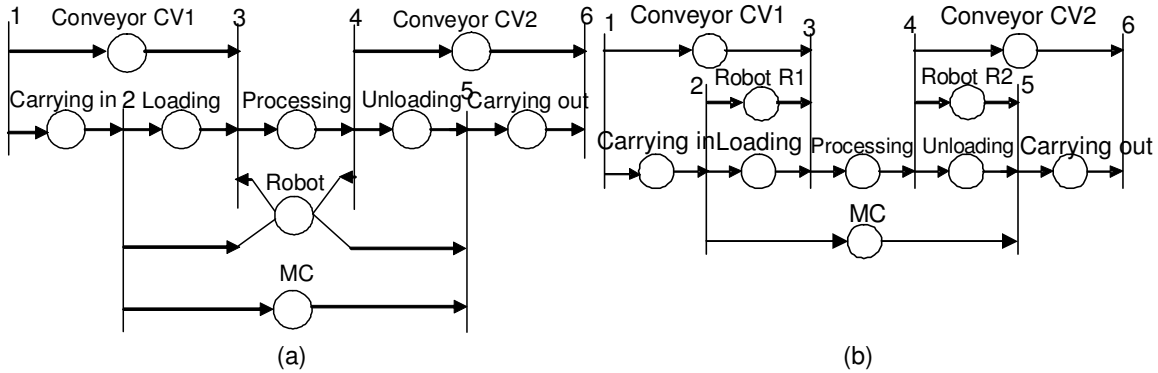



**FIGURE 10:** Petri net representation of the example cell at the conceptual level; (a) with one robot, (b) with two robots.

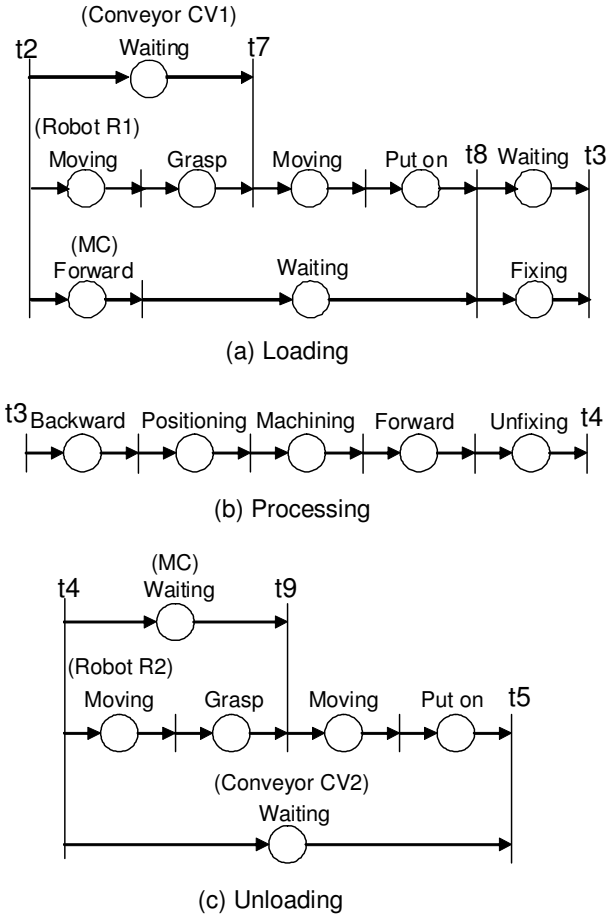



**FIGURE 11:** Detailed Petri net representation of subtasks.

The detailed Petri net representation of the example system is shown in Figure 12, where the detailed Petri net is not decomposed and the control system is centralized. For the distributed control system shown in Figure 6, the Petri net representations assigned to the machine controllers are shown in Figure 13. External permissive gate arcs from sensors for detecting the completion of work handling are employed. For example, for grasping action of a robot, the set of touch sensors on the fingers of the robot indicates whether the robot has or has not grasped a workpiece, and for moving action of a robot, the set of limit switch sensors for robot arm positioning indicates whether the robot has or has not reached a target position.

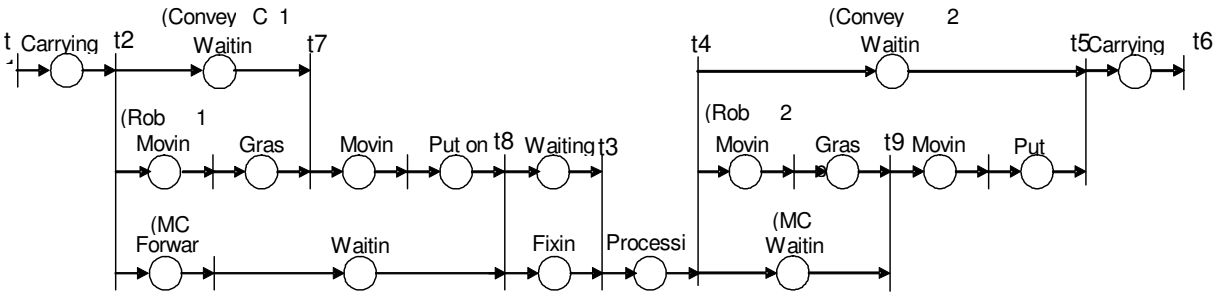



**FIGURE 12:** Detailed Petri net representation of whole system task.

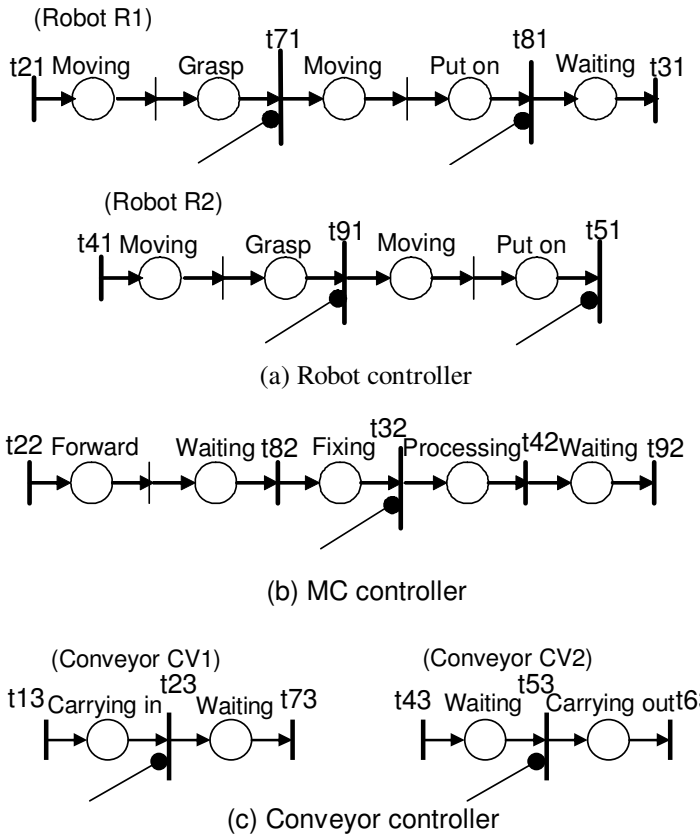



**FIGURE 13:** Detailed Petri net representation of machine controllers with external permissive gate arcs from sensors for work handling.

( : global transition, : local transition, : permissive gate arc)

For the example robotic system, a hierarchical and distributed control system has been realized using a set of PCs. Each machine controller is implemented on a dedicated PC. The system controller is implemented on another PC. Communications among the controllers are performed using serial communication interfaces (RS-485). The names of global transitions and their conflict relations are loaded into the coordinator in the system controller. The connection structure of a decomposed Petri net model and conflict relations among local transitions are loaded into the Petri net based controller in a machine controller. Using the names of transitions in the subsystems, global transitions are defined; for example, G2: t0-2, t1-21, t2-22, t3-23 indicates that the global transition G2 is composed of the transition no.2 of System controller (subsystem no.0), the transition no.21 of Robot controller, the transition no.22 of MC controller (subsystem no.2), and the transition no.23 of Conveyor controller (subsystem no.3). Then, the coordinator information for the example distributed control system is as follows.

| | |
|---|---|
| G1: t0-1, t3-13 | ; start of carring in |
| G2: t0-2, t1-21, t2-22, t3-23 | ; start of loading from CV1 |
| G3: t1-71, t3-73 | ; end of grasp on CV1 |
| G4: t1-81, t2-82 | ; end of putting on MC |
| G5: t0-3, t1-31, t2-32 | ; end of loading into MC |
| G6: t0-4, t1-41, t2-42, t3-43 | ; start of unloading from MC |
| G7: t1-91, t2-92 | ; end of grasp on MC |
| G8: t0-5, t1-51, t3-53 | ; end of putting on CV2 |
| G9: t0-6, t3-63 | ; end of carring out |

By executing the coordinator and Petri net based controllers algorithms based on loaded information, simulation experiments have been performed. The robot controller executes robot motion control through the transmission of command. The MC controller and the conveyor controller communicate with a dedicated PLC. The Petri net simulator initiates the execution of the subtasks attached to the fired transitions through the serial interface to the robot or other external machines. When a task ends its activity, it informs the simulator to proceed with the next activations by the external permissive gate arc. The detailed Petri net representation for real-time external machine control is shown in Figure 14.

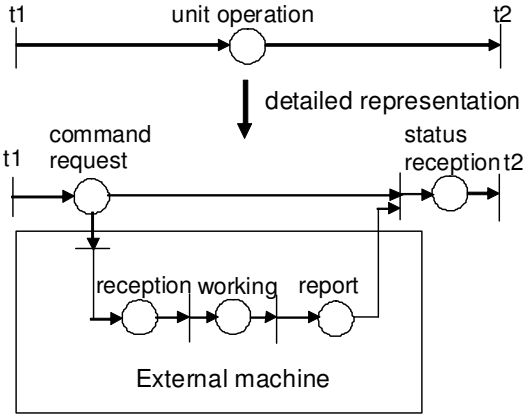



**FIGURE 14:** Detailed Petri net representation of external machine control.

Experimental set up of the example of robotic system presented in this paper is shown in Figure 15. Control software on each PC is written using multithreaded programming [9]. Petri net simulation and task execution program through serial interface are implemented as threads in each machine controller. Real-time task execution using multithreaded programming is shown in Figure 16. In a multithreaded operating system, a thread is an independent flow of execution in a process and all threads share the code, data, heap, and system memory areas of that process. So all threads in a process can access all global data. Further, because each thread has a separate CPU state and its own stack memory, all local variables and function arguments are private to a specific thread. It's easy for threads to interact with each other by the way of synchronization objects and intertask communications because they have some shared memory for communication as the process's global data. Context switching between threads involves simply saving the CPU state for the current thread and loading the CPU state for the new thread [10].

In the control system both the Petri net simulation thread and the task execution thread access to the external gate variables as shared variables; the task execution thread writes and the Petri net simulation thread reads. Mutual exclusive access control was implemented as shown in Figure 16, so that, while one thread accesses to the shared variables, the other thread can not access to them. The Petri net simulator thread waits for the external gate signal using an event object, and after the task execution threads write, the Petri net simulator thread reads and then the associated transition fires directly. Control software using multithreaded programming was written in Visual C# under OS Windows XP SP3 on general PCs.

The Petri net modeling thread is executed as the main thread to perform the Petri net modeling, drawing and modification for task specification. For the example system, detailed Petri net models can be automatically generated using the database of robotic operations. When the transformation of graphic data of the Petri net model into internal structural data is finished, the Petri net simulation thread starts. Experiments using a real robot show that the Petri net simulation thread and the task execution threads proceed concurrently with even priority, and the values of external gate variables are changed successfully.

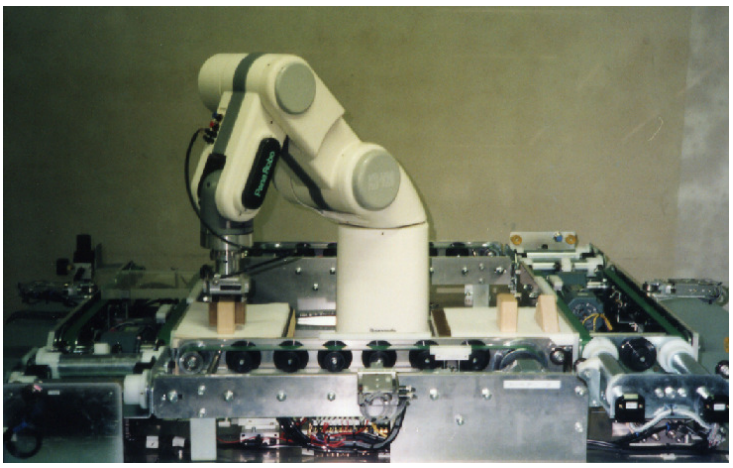


**FIGURE 15:** View of experimental set up of robotic system.

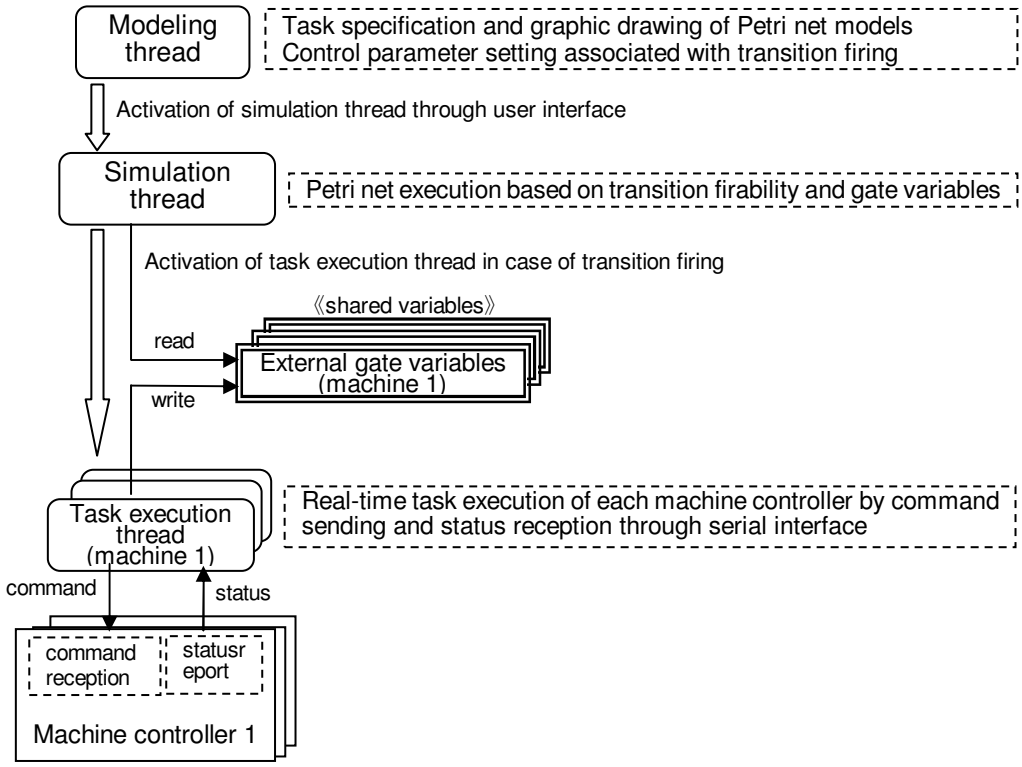



**FIGURE 16:** Real-time execution of subtasks using multithreaded programming.

Experimental results show that the decomposed transitions fire simultaneously as the original transition of the detailed Petri net of the whole system task [11]. The robots cooperated with the conveyors and the machining center, and the example robotic system performed the task specification successfully. Firing transitions and marking of tokens can be directly observed on the display at each time sequence using the simulator on each PC [12], as shown in Figure 17. During simulation and task execution, it is judged by the simulator that whether it is in a deadlocked situation or not, and the user can stop the task execution through the simulator at any time.

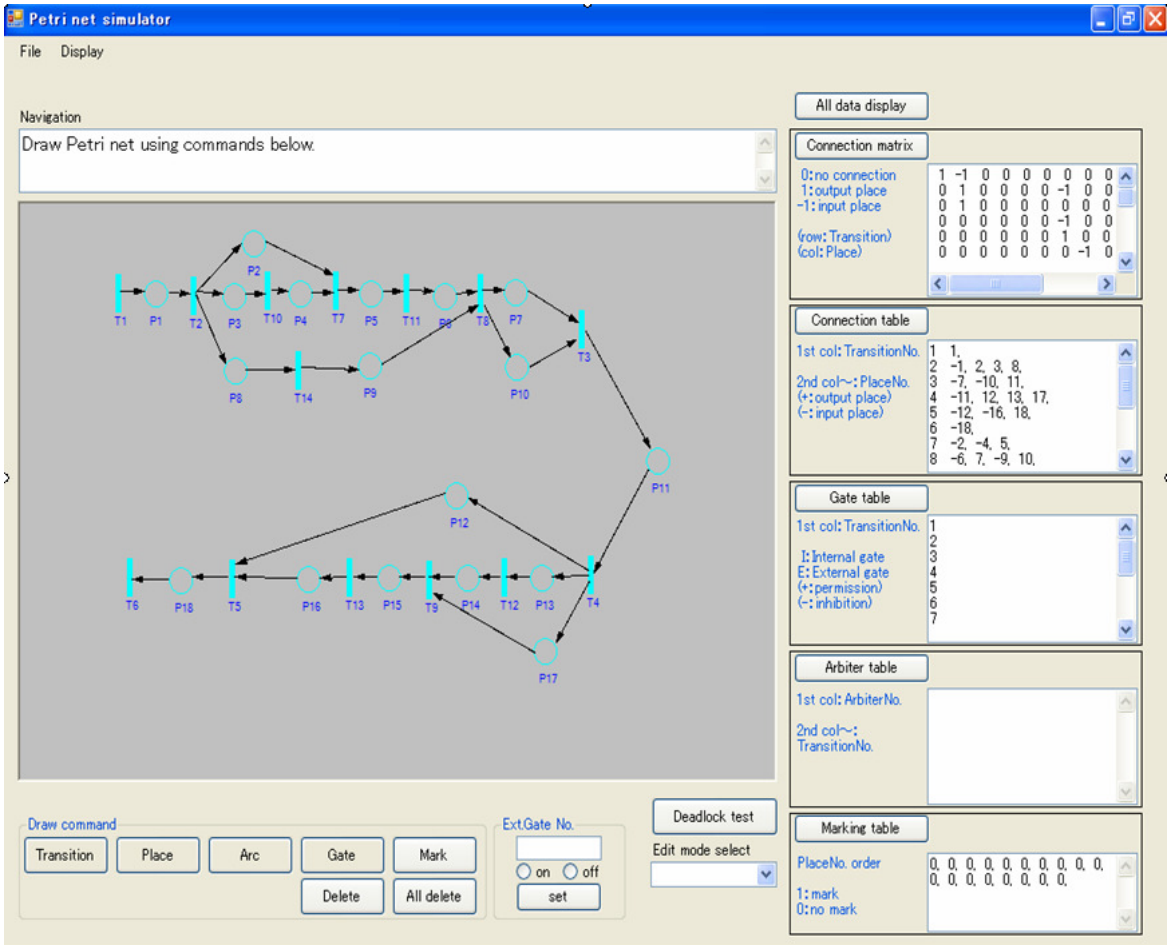



**FIGURE 17:** View of Petri net simulator.

## 5. CONCLUSIONS

A methodology of model based design and implementation using Petri nets to construct hierarchical and distributed control systems, which correspond to the hardware structure of robotic systems, has been proposed. By introduction of the coordinator, the Petri net based controllers are arranged according to the hierarchical and distributed nature of the overall system; the coordination mechanism is implemented in each layer repeatedly. The Petri net model in each Petri net based machine controller is not so large and easily manageable. The overall control structure of the example robotic system was implemented on general PCs using multithreaded programming. In accordance with the implementation using multithreaded programming, hierarchical and distributed implementation under a real-time operating system on a network of microcomputers connected via a serial bus is also possible, where each microcomputer is dedicated to the local Petri net model of a subsystem in the overall robotic system. Local Petri net models are so simple that they can be implemented on general PLCs. Thus, modeling, simulation and control of large and complex manufacturing systems can be performed consistently using Petri nets.

## 6. REFERENCES

1. M. Silva. "Petri nets and flexible manufacturing," in *Advances in Petri Nets 1989, Lecture Notes in Computer Science*. vol. 424, G. Rozenberg, Ed. Berlin: Springer-Verlag, 1990, pp. 374-417.

2. M. Silva and S. Velilla. "Programmable logic controllers and Petri nets: A comparative study," in *IFAC Software for Computer Control 1982*. G. Ferrate and E.A. Puente, Eds. Oxford, UK: Pergamon, 1982, pp. 83-88.

3. W. Reisig. *Petri Nets*. Berlin: Springer-Verlag, 1985.

4. T. Murata, N. Komoda, K. Matsumoto and K.Haruna. "A Petri net based controller for flexible and maintainable sequence control and its application in factory automation". *IEEE Trans. Industrial Electronics*, vol. 33(1), pp. 1-8, 1986.

5. R. David and H. Alla. *Petri Nets and Grafcet: Tools for Modelling Discrete Events Systems*. UK: Prentice-Hall International, 1992.

6. K. Hasegawa, K. Takahashi, R. Masuda and H. Ohno. "Proposal of Mark Flow Graph for discrete system control". *Trans. of SICE*, vol. 20(2), pp. 122-129, 1984.

7. P.E. Miyagi, K. Hasegawa and K. Takahashi. "A programming language for discrete event production systems based on Production Flow Schema and Mark Flow Graph". *Trans. of SICE*, vol. 24(2), pp. 183-190, 1988.

8. G. Yasuda. "Implementation of distributed control architecture for robotic manufacturing systems using Petri nets". in Proceedings of 2006 IMACS Multiconference on Computational Engineering in Systems Applications, 2006, pp. 1155-1160.

9. G. Yasuda. "A distributed control and communication structure for multiple cooperating robot agents," in *IFAC Artificial Intelligence in Real Time Control 2000*. Oxford, UK: Pergamon, 2000, pp. 283-288.

10. R. Grehan, R. Moote and I. Cyliax. *Real-time Programming A Guide to 32-bit Embedded Development*. Reading, Massachusetts: Addison-Wesley, 1998.

11. G. Yasuda. "Implementation of hierarchical and distributed control for discrete event robotic manufacturing systems". in Proceedings of the 38th International Conference on Computers and Industrial Engineering, 2008, pp. 2580-2589.

12. G. Yasuda. "Design and implementation of distributed control architecture for flexible manufacturing systems based on Petri nets". in Proceedings of the 10th Asia-Pacific Industrial Engineering and Management Systems Conference, 2009, pp. 2525-2534.

# Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching

**Santiago Garrido** sgarrido@ing.uc3m.es
*Departamento de sistemas y automática*
*Universidad Carlos III de Madrid*
*Leganés, 28914,Spain*

**Luis Moreno** moreno@ing.uc3m.es
*Departamento de sistemas y automática*
*Universidad Carlos III de Madrid*
*Leganés, 28914,Spain*

**Dolores Blanco** dblanco@ing.uc3m.es
*Departamento de sistemas y automática*
*Universidad Carlos III de Madrid*
*Leganés, 28914,Spain*

**Piotr Jurewicz** pjurewic@ing.uc3m.es
*Departamento de sistemas y automática*
*Universidad Carlos III de Madrid*
*Leganés, 28914,Spain*

**Abstract**

For navigation in complex environments, a robot needs to reach a compromise between the need for having efficient and optimized trajectories and the need for reacting to unexpected events. This paper presents a new sensor-based Path Planner which results in a fast local or global motion planning able to incorporate the new obstacle information. In the first step the safest areas in the environment are extracted by means of a Voronoi Diagram. In the second step the Fast Marching Method is applied to the Voronoi extracted areas in order to obtain the path. The method combines map-based and sensor-based planning operations to provide a reliable motion plan, while it operates at the sensor frequency. The main characteristics are speed and reliability, since the map dimensions are reduced to an almost unidimensional map and this map represents the safest areas in the environment for moving the robot. In addition, the Voronoi Diagram can be calculated in open areas, and with all kind of shaped obstacles, which allows to apply the proposed planning method in complex environments where other methods of planning based on Voronoi do not work.

**Keywords:** Voronoi Diagram, ath Planning, ast Marching.

## 1. INTRODUCTION

Since the 1980s mobile robot motion planning problems have become an important research topic that has attracted the attention of many researchers who have worked extensively to obtain efficient methods to solve these problems. This issue has been dealt with in two general ways: one approach has concentrated on solving motion planning problems using a previously known global environment or obstacle information and the robot characteristics, whilst the second approach has concentrated on planning motion using local sensor information and the robot characteristics.

In recent years, sensor-based path planning has emerged as a compromise between the need for reacting to unexpected environmental events and the need for having efficient optimized trajectories. Different variations of classical methods have been proposed to achieve an operative sensor-based path planning. One of these techniques is the Voronoi-based path planning. In order to use a sensor-based method an efficient and fast algorithm is required to extract and follow Voronoi Diagrams.

The Voronoi Diagram has been studied by many researchers. The advantages of Voronoi-based path planning are that it diminishes the dimension of the problem to one and that the trajectories follow the maximum clearance map. This means that the trajectories are the safest ones. Moreover, the Voronoi-based navigation reminds of topological navigation and, because of that, it is similar to the human one in some aspects. However, most Voronoi-based methods have the difficulty of calculating the Voronoi Diagram by studying lines and polygons, finding the vertices and nodes, and creating a tree to find the path. In addition, the method is not very efficient in three dimensions and it does not work at all in higher dimensions. As an alternative to solve these problems, we propose the use of image processing methods.

In order to calculate the trajectory, the new Motion Planning method is based on the combination of a Voronoi Diagram and the expansion of a wave from the start point to the goal following the Voronoi Diagram.

The proposed sensor-based planning method uses a combination of sensor information and the known map of the environment. The Voronoi Diagram is built using image methods (skeletonization) based on the Breu method in 2D [7] and Gagvani in 3D [16]. This Voronoi Diagram (skeleton) is dilated and inverted to obtain a viscosity map. The next step is to calculate the trajectory in the image generated by the thick Voronoi Diagram using a wave expansion. The wave expansion is calculated solving the Eikonal equation for which the Fast Marching Method has been used, though there are other similar ones, such as the Tsiksitlis [54] method, MCC [32], and others.

Then, the obtained path verifies the smoothness and safety considerations required for mobile robot path planning.

The approach implemented in this work is based on the potential field method of the wave expansion, which repels a robot away from obstacles and towards the goal using a carefully designed artificial potential function. The main advantages of this method are:

- It is easy to implement.
- The computation time is low, so this algorithm works in Real-Time.

## 2. RELATED WORK

### 2.1 Voronoi Diagram

The Voronoi concept has been used for four centuries. In his Traité de la Lumiére published in 1644, Descartes uses diagrams similar to Voronoi to show the disposition of matter in the solar system and its environment. Algorithms of Voronoi Diagrams have been appearing since the 1970s. See the surveys by Aurenhammer [2,3], de Berg [5], and Okabe [39] on various algorithms, applications, and generalizations of Voronoi Diagrams.

The Generalized Voronoi Diagram is the set of points where the distance to the two closest objects is the same [11]. Many algorithms have been proposed for constructing Generalized Voronoi Diagrams using the distance information provided by different external sensors like sonar sensors, laser scanners and stereo cameras. The method for construction of a Generalized Local Voronoi Diagram (GLVG) applied by Mahkovic in [31] uses measurements from a laser scanner. First, the points that belong to the same object are clustered, then the Voronoi Diagram is

generated, and finally, the edges outside the visible region and those which both side generator points belonging to the same object are deleted. In [53], Sudha constructs a Voronoi Diagram from a binary image of the workspace obtained using a digital camera.

One class of algorithms involves incremental construction procedures based on the exploration of an unknown environment. It is the case of the approach proposed by Nagatani, Choset, and Thrun [35,9,11]. This algorithm does not consider the encoder errors and it cannot be used in a large environment.

There are some path planning algorithms inspired by the Voronoi Diagram concept, such as the MAPRM method [57], which retracts sampled configurations onto the medial axis (or Generalized Voronoi Diagram). Also, the EquiDistance Diagram (EDD) [22] is based on the Voronoi roadmap idea, that is a roadmap formed by connecting the local maxima of a clearance function defined using distance functions. The main shortcoming of these methods is that their roadmap is constructed offline and the environment information must be provided in advance.

Breu et al. [7] present a linear time (and hence asymptotically optimal) algorithm for computing the Euclidean distance transform of a two-dimensional binary image. The algorithm is based on the construction and regular sampling of the Voronoi Diagram whose sites consist of the unit (feature) pixels in the image. Their algorithm constructs the Voronoi Diagram where it intersects the horizontal lines passing through the image pixel centers. They also discuss the extensions to higher dimensional images and to other distance functions. The method proposed in this paper uses, in its first part, this algorithm.

### 2.2 Potential Field-based Motion Planning

From a theoretical point of view, the motion planning problem is well understood and formulated, and there is a set of classical solutions capable of computing a geometrical trajectory that avoids all known obstacles. Most of these general methods are not applicable if the environment is dynamic or there are no modelled obstacles. Hence, to avoid the problem of executing an unrealistic geometrical trajectory, in which the robot can collide with objects, obstacle avoidance algorithms have been developed to provide a robust way of coping with the problem.

One of the classical methods for dynamically solving the collision avoidance problem is the potential field approach developed by O. Khatib [23]. This approach is based on the creation of an artificial potential field in which the target is an attractive pole and the obstacles are repulsive surfaces. The robot follows the gradient of this potential toward its minimum. The derived force induces a collinear and proportional acceleration enabling easy dynamic and kinematic control actions. This technique can be used at a global or local level depending on the information used to generate the potentials. The major advantage of this method is its simplicity and its capability of being used dynamically because of the easy treatment of fixed and mobile obstacles. Its major disadvantage is the possible existence of local minima and the oscillations for certain configurations of obstacles. In spite of this problem, this technique has been used extensively in reactive architectures because of its natural ability to encapsulate specific robot behaviors.

In [33], Melchior at al. find an optimal trajectory using the Fast-Marching technique and compare it with the A* algorithm.According to Melchior, the solutions obtained with the Fast-Marching technique are the true approximations of the continuous and optimal trajectories. Thus the level line curvature and the gradient orientation are continuous; the robot can be oriented quasi-continuously. However, the minimum length trajectory obtained by the A* algorithm is not the discrete approximation of the optimal length continuous trajectory.

The Fast Marching Method has been used previously in Path Planning by Sethian [49,48], but using only an attractive potential. This method has some problems. The most important one that typically arises in mobile robotics is that optimal motion plans may bring robots too close to obstacles (including people), which is not safe. This problem has been dealt with by Latombe [26], and the resulting navigation function is called NF2. Melchior, Poty and Oustaloup [33,42]

present a fractional potential to diminish the obstacle danger level and to improve the smoothness of the trajectories; Philippsen [41] introduces an interpolated Navigation Function, but with trajectories too close to obstacles and without smooth properties; and Petres et al. [40] introduce efficient path-planning algorithms for Underwater Vehicles that take advantage of underwater currents. They developed an algorithm called $FM^*$, which is a continuous version of the $A^*$ algorithm based on Fast Marching. The Voronoi-based Method presented in [18] tries to follow a maximum clearance map.

### 2.3 Grid-based Planning Methods

In the navigation of mobile robots, many methods consist on a grid-based representation of the environment. This representation can have occupied or free cells (binary representation), or each cell can have an associated weight that represents the difficulty of traversing that area.

This grid is usually approximated by a graph, with the nodes situated in the center of each cell. Many algorithms have been developed to find a path in the graph. Dijkstra's algorithm computes the optimal path between a single source point to any other point in the graph. This algorithm is indeed efficient, but suffers from metrication errors [13], as shown in Fig. 1. $A^*$ uses a heuristic to focus the search from a particular start location towards the goal, and thus produces a path from a single location to the goal very efficiently [37]. $D^*$, Incremental $A^*$, and $D^*$ Lite are extensions of $A^*$ that incrementally repair solution paths when changes occur in the underlying graph [52]. These incremental algorithms have been used extensively in robotics for mobile robot navigation in unknown or dynamic environments.

Grid or graph based methods, such as Dijkstra's or $A^*$, calculate a navigation function constrained to movement choices along graph edges that are neither optimal for execution nor an exact solution. The graph based algorithms consider the image as an oriented graph in which a pixel is a node, and the 4 (or 8) connections to the neighboring pixels are the vertices of the graph.

These methods can not converge to the solution of a continuous problem. Even for very fine grids, the paths restrict the agent's heading to increments of $\pi/2$ or $\pi/4$ . This results in paths that are suboptimal in length and difficult to traverse in practice. The different metrics used lead to very different results.
Frequently, a smoothing step is performed after planning to alleviate this problem, which yields to unsatisfactory results because it solves only locally the discrete nature of the problem.

Various efforts have been made to overcome these shortcomings, including increasing the available headings or approximating continuous paths (see [36] and [14]), using suboptimal variants of the search algorithms to reduce computation time [28], and efficiently updating existing solutions when new information is received [52], [25].

The fast marching method uses the L2 metric, instead of the L1 (Manhattan) of Dijkstra's similar methods, and it makes an intrinsic interpolation that allows it to solve the continuous eikonal efficiently equation and to give the shortest path.

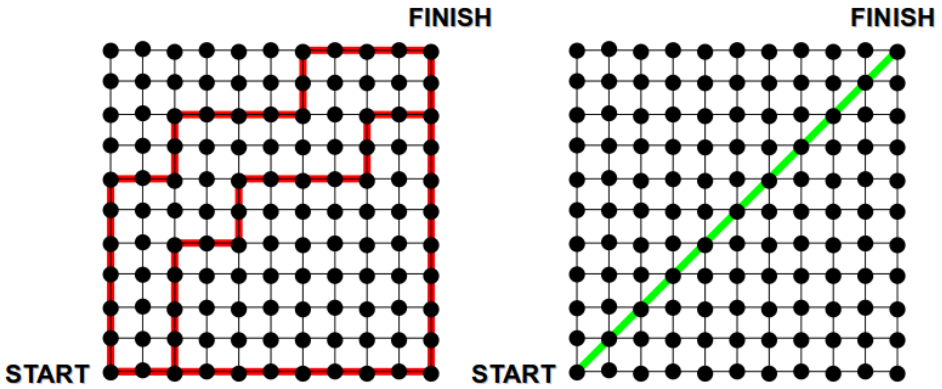



**FIGURE 1:** The Dijkstra Method gives multiple short paths but always following the conection between the graph points. Fast Marching Method gives the optimal diagonal path, because it interpolates, so the path obtained is the shortest.

## 3. INTRODUCTION TO THE VORONOI DIAGRAM AND SKELETON

The Voronoi Diagram concept is a simple but intuitively appealing one. Given a finite set of objects in a space, all locations in that space are associated with the closest member of the object set. The result is a partition of the space into a set of regions, Voronoi regions. The Generalized Voronoi Diagram is the frontier between these regions. Given its widespread use, it is not surprising that this concept has been discovered many times in many different places.

The close relationship between the Voronoi Diagram and the medial axis has already been pointed out in literature [38]. In the field of computer vision, the skeleton of an object or image is the Medial Axis Transform, which coincides with the definition of the GVD used in robotics that uses lines. This is not to be confused with the Voronoi Diagram of a discrete set of points: for each point $x$ in the set $P$, there exists a boundary polygon enclosing all the intermediate points lying closer to $x$ than to other points in the set $P$, and the set of these polygons for a given point set is called the Voronoi Diagram.

A very illustrative definition of the skeleton is given by the prairie-fire analogy: the boundary of an object is set on fire and the skeleton is made up of the loci where the fire fronts meet and quench each other.

Based on Blum's definition of a skeleton [6], the skeleton $S$ of a set $D$ is the locus of the centers of maximal disks. A maximal disk of $D$ is a closed disk contained in $D$ that is interiorly tangent to the boundary $\delta D$ and that is not contained in any other disk in $D$. Each maximal disc must be tangent to the boundary in at least two different points. With every skeleton point $s \in S$ we also store the radius $r(s)$ of its maximal disk. The skeleton $\sigma(D)$ is the set of centers of maximal disks in $D$. The skeleton is desired to be a 'graph-like' retraction of the original set. For a good overview of skeletonization techniques, see [51].

A class of skeletonization approaches is based on distance transform. In this paper algorithms based on distance transform (DT) are used for computing the skeleton (Voronoi Diagram) of the binary image that represents the environment. This associates every pixel in image space with its distance to the nearest point to it. The distance transform is important for natural neighbor interpolation of a function and for morphological operations on images, such as medial axis and skeleton computation, and polygon morphing. Many efficient algorithms exist for the computation of the distance transform, or a close approximation of it (meaning that the distances are not exactly the true distances) for images. The medial surface/axis can be defined as the locus of

maximal circles (2D) or ball (3D). A particularly efficient algorithm that has been used for the calculation of the Voronoi Diagram (2D) in this work is due to Breu et al [7]. For its way of working it is not necessary to calculate vertices and nodes of the Voronoi Diagram and it can be used for all kinds of walls and obstacles, even curved ones. It runs in $O(m)$ time, where $m$ is the number of image pixels. This algorithm is based on the construction and regular sampling of the Voronoi Diagram whose sites consist of the unit (feature) pixels in the image. The algorithm constructs the Voronoi Diagram where it intersects the horizontal lines passing through the image pixel centers. This algorithm has been used to compute the Voronoi Diagram in 2D environments.

An important drawback of the skeletonization methods is the existence of spurious *branches* on the generated skeletons. This undesired fact is caused by the irregularities of the boundary. Boundary noise would cause the Voronoi Diagram to be very dense which would lead to the need for appropriately pruned it to generate the medial axis. Ogniewicz [38] reduced skeletons formed from raster boundary points to a simple form. This was achieved by pruning the hair nodes of the skeleton until a given minimum circumcircle was reached. On the other hand, it has been demonstrated that hair nodes correspond to a location of minimum curvature on the boundary [4,6]. This means that for a sampled boundary curve, three adjacent points are co-circular, with their center at the skeleton hair. For the simplification of the skeleton, the hairs should be retracted to their parent node location. The central point of the three is moved towards the parent node until it coincides with the parent node circumcircle, resulting in smoothing outward-pointing salients of the boundary.

The process is repeated from the other side of the boundary, also retracting those salients. This may displace some of the points involved in the first smoothing step, but the process is convergent and a small number of iterations suffices to produce a smoothed curve having the same number of points as the original one, but with a simplified skeleton.

In order to extract the skeleton in 3D, the algorithm described in [16] has been implemented. That skeletonization method for 3D objects is based on a quasi-Euclidean distance transform. Once the distance transform value at every object point is known, a linded list of all the object points is created. One pass is done over the list to test if a local maximum of it satisfies the condition:

$$MNT_p \ll DT_p - TP \tag{1}$$

Where $TP$ is a thinness parameter, $DT_p$ is the distance transform of the voxel $p$, and $MNT_p$ is the mean of the neighbors' distance transform. The effect of boundary noise is reduced by choosing a thinness parameter that removes most of the *'hairs'* in the skeleton.

## 4. INTUITIVE INTRODUCTION OF THE EIKONAL EQUATION AND THE FAST MARCHING PLANNING METHOD

The ideal planner always drives the robot in a smooth and safe way to the goal point. In nature there are phenomena with the same way of working: electromagnetic waves. If there is an antenna at the goal point that emits an electromagnetic wave, then the robot could drive himself to the destination following the waves to the source. The concept of the electromagnetic wave is especially interesting because the potential has all the good properties desired for the trajectory, such as smoothness (it is $C^{\infty}$) and the absence of local minima.

Maxwell's laws govern the propagation of the electromagnetic waves and can be modelled by the wave equation:

$$\frac{\partial^2 \phi}{\partial t^2} = c^2 \nabla^2 \phi \tag{2}$$

A solution of the wave equation is called a wave. The moving boundary of a disturbance is called a wave front. The wave front can be described by the Eikonal equation. The reasoning that demonstrates it can be followed in [12].

In Geometrical Optics, Fermat's *least time principle* for light propagation in a medium with space varying refractive index $\eta(\boldsymbol{x})$ is equivalent to the Eikonal equation and can be written as $||\nabla\Phi(\boldsymbol{x})|| = \eta(\boldsymbol{x})$, where the Eikonal $\Phi(\boldsymbol{x})$ is a scalar function whose isolevel contours are normal to the light rays. This equation is also known as Fundamental Equation of the Geometrical Optics.

The Eikonal (from the Greek 'eikon', which means 'image') is the phase function in a situation for which the phase and amplitude are slowly varying functions of position. Constant values of the Eikonal represent surfaces of constant phase, or wave fronts. The normals to these surfaces are rays (the paths of energy flux). Thus the Eikonal equation provides a method for 'ray tracing' in a medium of slowly varying the refractive index (or the equivalent for other kinds of waves).
In the Sethian notation [47] the eikonal equation is written as

$$|\nabla T(x)|F(x) = 1 \tag{3}$$

where $\boldsymbol{T}(\boldsymbol{x})$ represents the wave front (time), $\boldsymbol{F}(\boldsymbol{x})$ is the slowness index of the medium, and $\boldsymbol{x} = (x, y)$ in 2D or $\boldsymbol{x} = (x, y, z)$ in 3D.

The theory and the numerical techniques known as Fast Marching Methods are derived from an exposition to describe the movement of interfaces, based on a resolution of the equations on partial differential equations as a boundary condition problem. They have the merit of unifying the ideas relative to the evolution of fronts in a general framework.

For the formulation of a front in evolution, having considered a generic limit, we introduce, for example, a unidimensional expansion, one curve in two dimensions (see Fig. 1), or one surface in three dimensions, which separates one region from another.

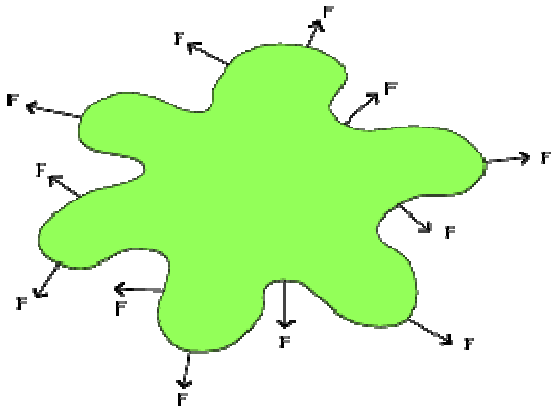



**FIGURE 2:** Wave front propagating with velocity F

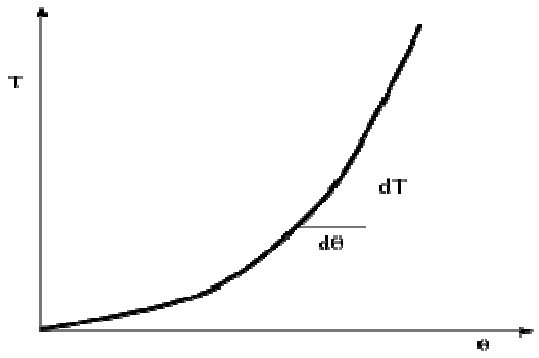



**FIGURE 3:** Formulation of the arrival function $T(\theta)$, for a unidimensional wavefront

Let us imagine that the curve or surface moves in its normal direction with a known speed F. The objective would be to follow the movement of the interface while this one evolves. A large part of the challenge in the problems modelled as fronts in evolution consists on defining a suitable speed that faithfully represents the physical system.

One way to characterize the position of a front in expansion is to compute the time of arrival T, in which the front reaches each point of the underlying mathematical space of the interface. It is evident that for one dimension (see fig. 2) we can obtain the equation for the arrival function T in an easy way, simply considering the fact that the distance $\theta$ is the product of the speed $F$ and the time $T$.

$$\theta = F \cdot T$$

The spatial derivative of the solution function becomes the gradient

$$1 = F\frac{dT}{d\theta}$$

and therefore the magnitude of the gradient of the arrival function $T(\theta)$ is inversely proportional to the speed.

$$\frac{1}{F} = |\nabla T|$$

For multiple dimensions, the same concept is valid because the gradient is orthogonal to the level sets of the arrival function $T(\theta)$.

This way, we can characterize the movement of the front as the solution of a boundary condition problem. If speed F depends only on the position, then the equation $\frac{1}{F} = |\nabla T|$ can be reformulated as the eikonal equation

$$|\nabla T|F = 1. \tag{4}$$

As a simple example we defined a circular front $\gamma_t = \{(x,y)/T(x,y) = t\}$ for two dimensions that advance with uniform speed. The evolution of the value of the arrival function $T(\theta)$ can be seen as time increases (i.e. $T = 0$, $T = 1$, $T = 2$, ...), and the arrival function comes to points of the plane in more external regions of the surface (see fig. 3). The boundary condition is that the value of the wave front is null in the initial curve.

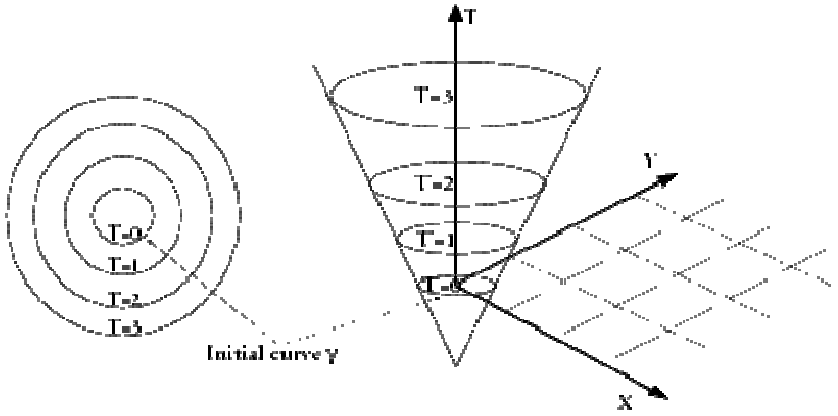



**FIGURE 4:** Movement of a circular wave front, as a problem of boundary conditions

The direct use of the Fast Marching Method does not guarantee a smooth and safe trajectory. From the way the front wave is propagated the shortest geometrical path is determined. This makes the trajectory unsafe because it touches the corners, walls, and obstacles, as shown in Fig. 4. This problem can be easily solved by enlarging the obstacles, but even in that case the trajectory tends to go close to the walls and it is not sufficiently smooth or safe enough.

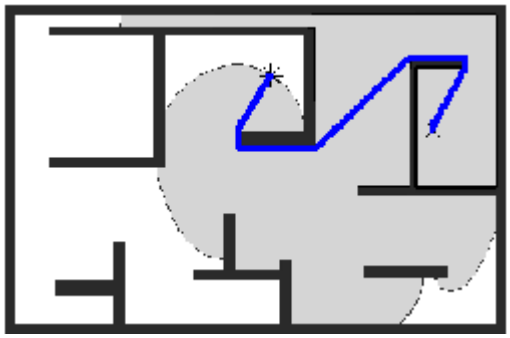


**FIGURE 5:** Trajectory calculated by Fast Marching Method directly. The trajectory is not safe because it touches the corners and walls.

The image 4 shows that the Fast Marching Method can not be used directly *Fast Marching Methods* are designed for problems in which the speed function never changes sign, so that the front is always moving forward or backward. This allows to convert the problem into a stationary formulation, because the front crosses each grid point only once. This conversion into a stationary formulation, in addition to a whole set of numerical tricks, gives it a tremendous speed to the method.

The distance given by the wave propagation of the Fast Marching Method is the Geodesic distance, which is different from the Euclidean Distance, as shown in Fig. 5. In that figure it is possible to observe how the points on the left foot are farther from the start when Geodesic distance is used. The Geodesic distance is the length of the shortest path between the two points.

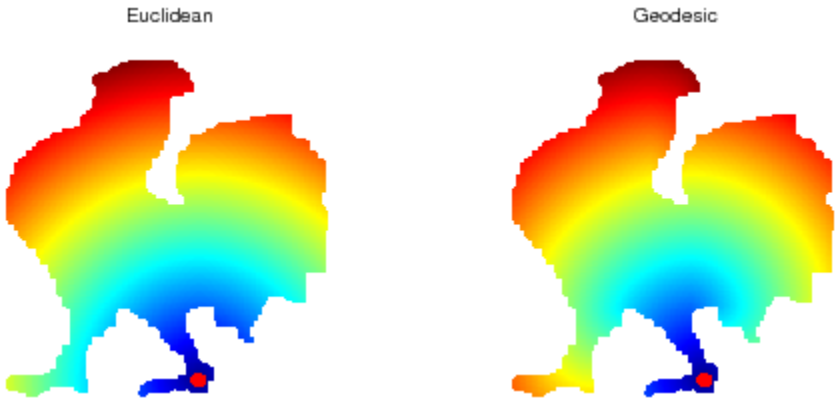



**FIGURE 6**: Difference between the Euclidean distance and the Geodesic distance calculated with the Fast Marching Method. The Geodesic distance is the distance of the minimum length inside the figure path and the Euclidean distance is the straight line distance

Since its introduction, the Fast Marching Method approach has been successfully applied to a wide set of problems that arise in geometry, mechanics, computer vision, and manufacturing processes, see Sethian [48] for details. Numerous advances have been made to the original technique, including the adaptive narrow band methodology [1] and the Fast Marching Method for solving the static eikonal equation ([49],[48]). For further details and summaries of level set and fast marching techniques for numerical purposes, see [48].

## 5. IMPLEMENTATION OF THE METHOD

This method operates in two main steps. The first step consists on the calculation of the Voronoi Diagram of the 2D or 3D map of the environment (that is, the cells located equidistant to the obstacles). This process is done by means of morphological operations on the image of the environment map. In the second step, the Fast Marching Method is used to create a potential T(x) that represents the propagation of an electromagnetic wave. This potential is used to calculate the trajectories based on the potential surface defined by the slowness map. To be more precise, the planner follows the following steps:

1. **Modelling.** A grid-based map of the environment is the model to be used. The a priori known map (if it is known) and the sensor information are integrated in an updated image of the environment with black and white cells. The only criteria to select the grid size is the computation time. The grid does not need to be uniform, being possible to use a triangular mesh.

2. **Object enlarging.** In order to avoid unrealistic trajectories, the objects and walls are enlarged by the radius of the mobile robot before the calculation of the Voronoi Diagram to ensure it neither collides with obstacles or walls nor accepts passages narrower than the robot size. This allows to consider the robot as a point from now on.

3. **Filter.** In order to remove the small *'hairs'* of skeletons, an averaging filter is applied. This filter is applied over the binary image to eliminate the corners and avoid the appearance of hairs.

4. **Voronoi Diagram.** Morphological image processing techniques are applied to the map to build the Voronoi Diagram. To be exact, a skeletonization of the image based in the Breu method (2D) [7] and Gagvani's (3D) [16] is applied in order to obtain the Voronoi Diagram as shown in Fig. 5. This skeleton is an image in which the points of the Voronoi diagram are black and the others white, but there are no information of vertices and nodes. It is possible to calculate the nodes, but it is more practical to search the path over the image with the Fast Marching method, because the calculation time is smaller and it works even with curved obstacles and walls.

5. **Dilatation** After that, a dilatation is done in order to have a thick Voronoi Diagram (see Fig. 5) wherein to calculate the propagation of the wave front of the Fast Marching Method. This is done in order to obtain two advantages. First, the sharp angles between segments of the diagram are smoothed, which improves the continuity of the generated trajectories and frees them from sharp angles. Second, the Voronoi Diagram is thickened in order to eliminate excessive narrowing, and thereby to allow a better propagation of the wave front when applying the Fast Marching Method. This way, the trajectory can be calculated.This thickening of all the lines of the Voronoi Diagram is done in the same amount, but the grade of dilatation is the only design parameter of the method. It can be small enough to have no isolated points or big enough to have a tube in which non-holonomic vehicles maneuver (see non-holonomic section) .

6. **Viscosity Map.** In order to apply the FMM, it is necessary to invert the image (see Fig. 1) to obtain a viscosity (or slowness) map since the wave travels faster in the clearer zones and slower in the darker ones.

7. **Fast Marching Method.** The next step is to calculate the trajectory in the image generated by the Voronoi Diagram using a wave expansion. The wave equation is approximated using the paraxial approximation used in optics. This way, the Eikonal equation is obtained. To solve the Eikonal equation the Fast Marching Method has been used. The Fast Marching Method is used to create a potential T(x) that represents the propagation of an electromagnetic wave in a viscosity map, to which the time is added as the third axis in 2D or the fourth axis in 3D. The origin of the wave is the goal point, which continues propagating until it reaches the current position of the robot. The Fast Marching Method produces a potential with a funnel-shaped surface, in which the time is the last axis, i.e. ,$T(x)$.

8. **Path Calculation.** This potential is used to calculate the trajectories based on the potential surface defined by the slowness map using the gradient method with the current position of the robot as the starting point and the goal position as the final point (see Fig. 1). The gradient determines the direction of travel after $FMM$ has calculated $T(x)$.

The algorithm steps are summarized in the flowchart in Fig.5. The trajectory is obtained inside the safest areas provided by the thickened Voronoi Diagram and is properly smooth, especially at the angles, since the Fast Marching Method selects a continuous path in gradient terms. Moreover, the path extraction is very fast because the Fast Marching Method propagates in an almost unidimensional curve (although this is an approximation since the Voronoi Diagram is thickened in the perpendicular direction to the diagram curves).

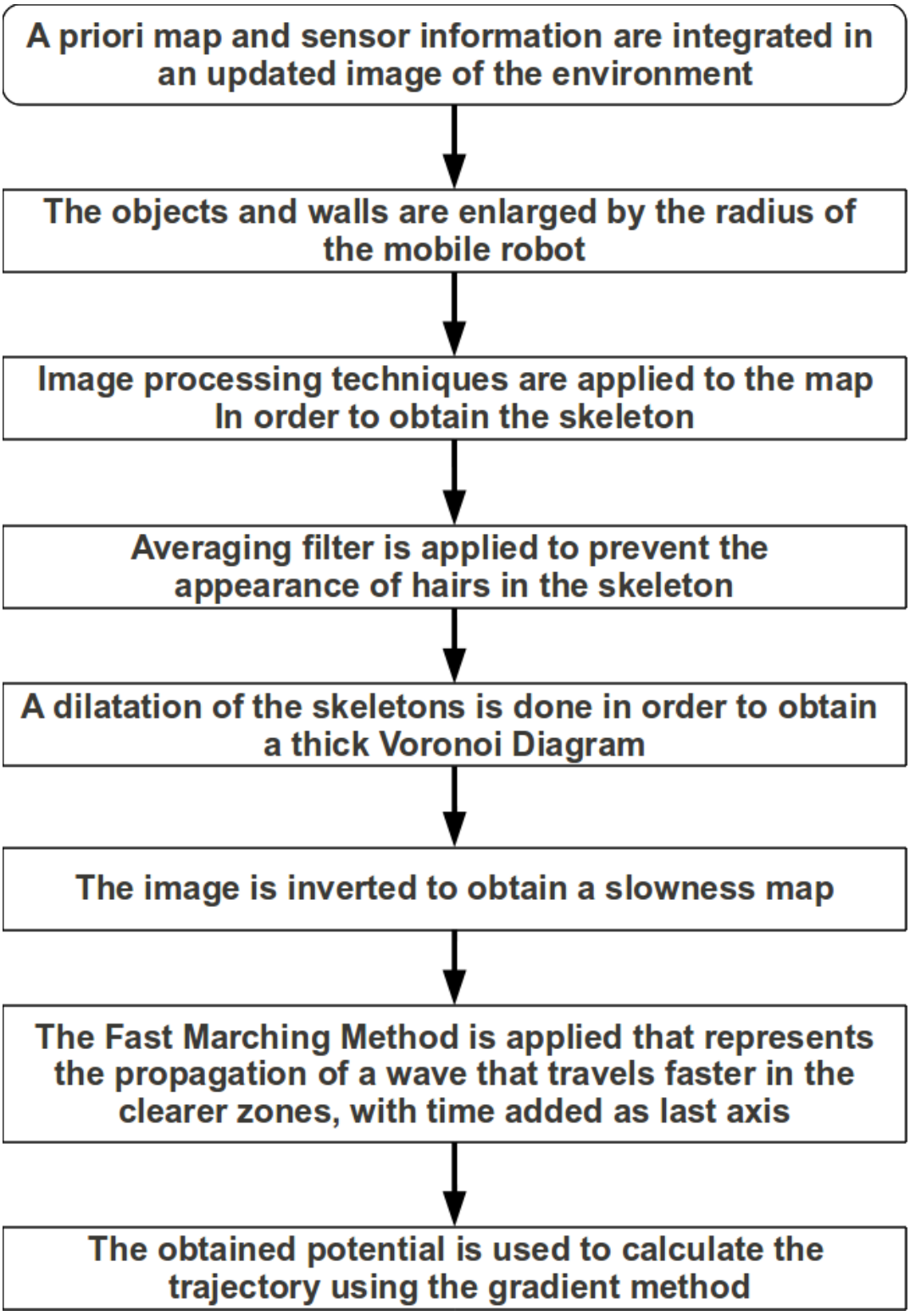
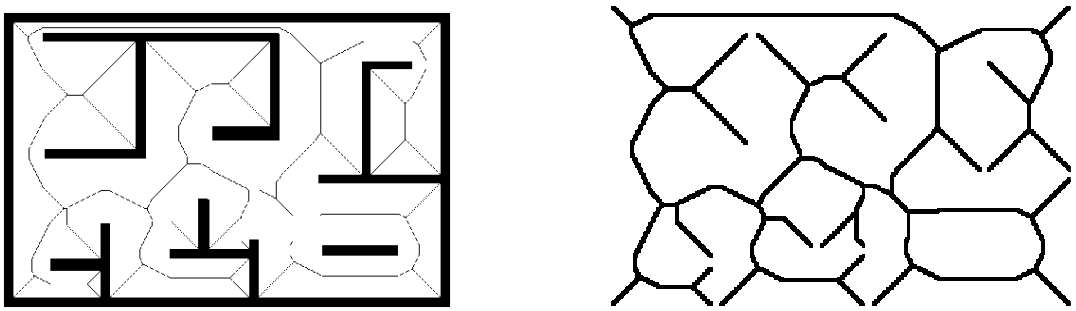



**FIGURE 7**: Flowchart of the proposed planning algorithm



(a) Voronoi Diagram (b) Thickened Voronoi Diagram

**FIGURE 8 :** Map of the room used in the first experiment

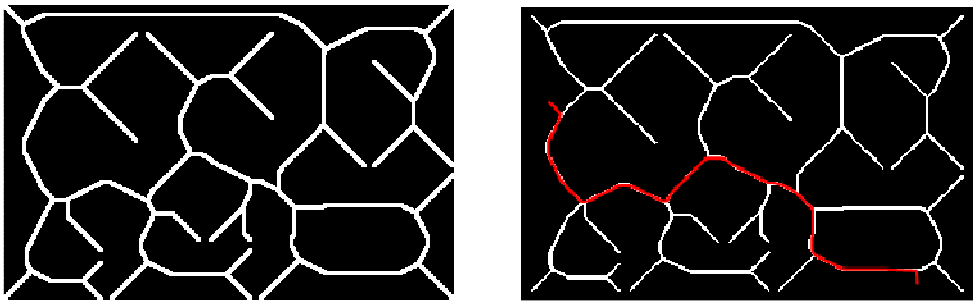


**FIGURE 9**: Trajectory calculated by Fast Marching Method in the inverted image of the thickened Voronoi Diagram of the room

The proposed method can also be used for sensor-based planning, working directly on a raw laser sensor image of the environment, as shown in Fig. 8 and 9. The data shown corresponds to the corner between two perpendicular aisles of the Carlos III University to the left of the robot, which is located in the center of the bottom of the scene. The image obtained from raw data may contain scattered pixels (see Fig. 7), and the dilatation of the image achieves the continuity and thickening of the lines representing the walls and obstacles (see Fig. 7). There are similar spurious lines behind the walls because the Voronoi diagram is made with an open image, but that path followed by the robot is correct. At this point the path planning steps described above are applied in the same sequence. The capability of using raw sensor data combined with the speed of the calculation allows this methodology to be used online to recalculate the trajectory at any time, avoiding this way dynamic obstacles and objects not present in the original map.

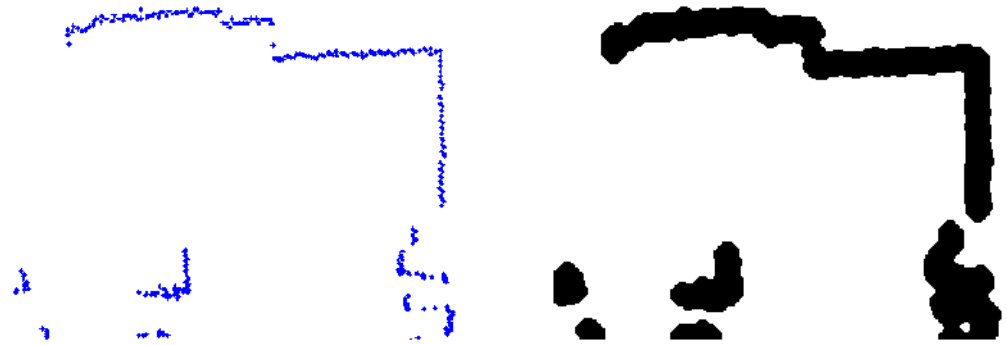


**FIGURE 10**: Data read by the robot (Local map)

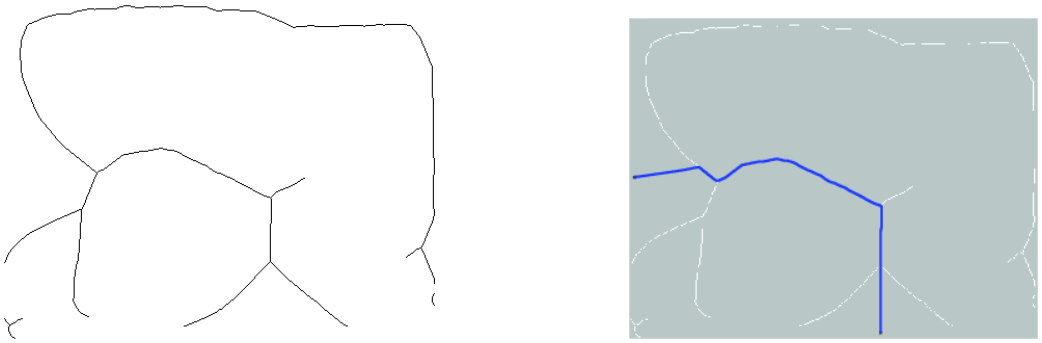


**FIGURE 11**: Voronoi Diagram and Trajectory of the Local map

## 6. RESULTS

The proposed method has been tested using the manipulator robot Manfred shown in Fig. 16. It has a coordinated control of all the degrees of freedom in the system (the mobile base has 2 DOF and the manipulator has 6 DOF) to achieve smooth movement. This mobile manipulator uses a sensorial system based on vision and 3D laser telemetry to perceive and model 3D environments. The mobile manipulator will include all the capabilities needed to localize and avoid obstacles and to navigate safely through the environment.


**FIGURE 12**: The robot Manfred has been used to test the algorithms

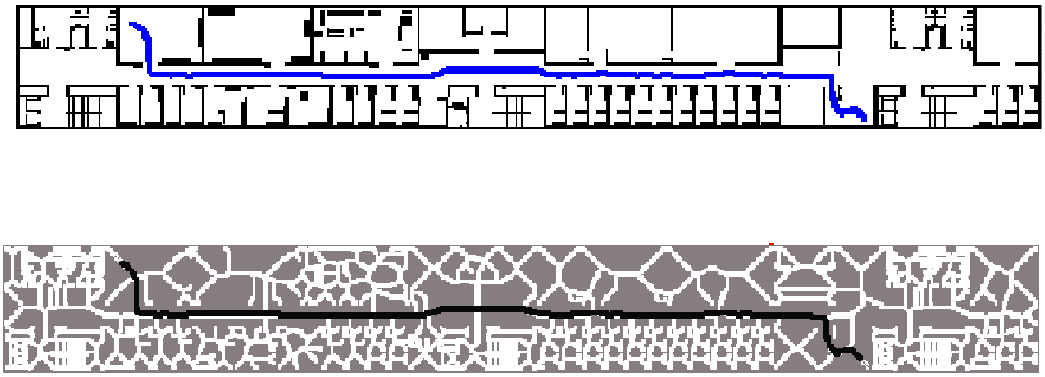


**FIGURE 13**: Trajectory calculated with Fast Marching over the Voronoi Diagram (Global map)

To illustrate the capabilities of the proposed method different tests are presented in this section. In the first test (room test with a resolution of $628 \times 412$ pixels), a difficult test room environment and the floor of the laboratory environment have been used (Fig. 6 and Fig. 7).

In the second test (laser test), the method is applied to a local environment path-planning task where the laser scanner measurement data are used to generate the trajectories, as it was explained in the previous section. Fig. 2 and 9 illustrate the achieved good trade-off between trajectory length, distances to obstacles and smooth changes in the trajectory. The Generalized Voronoi diagram of Fig. 13a goes behind the walls because the GVD is calculated on the image and in this image there is a possible path behind the walls.

The method has capabilities to generate adequate paths on a global scale as shown in Fig. 16. The results for the complete lab floor are shown in fig. 16 top (the environment map of the Robotics Lab of the Carlos III University) and fig. 16 bottom (the path obtained after applying the Fast Marching Method to the previous Voronoi Diagram image). The method provides smooth trajectories that can be used at low control levels without any additional smooth interpolation process.

The last test (lab test) is dedicated to illustrate the capability of the proposed method of adapting to a changing environment taking into account possible dynamic features of the environment such as moving obstacles and persons in the vicinity of the robot, as shown in Fig. 16. During the motion, the robot observes the environment with its laser scanner, introduces the new information in the map and plans a new trajectory. Local observations (obstacle in the middle of the corridor) generate modified trajectories in order to avoid the detected obstacles. In the bottom map of the figure the obstacles detected block the corridor and the sensor-based global planner generates a completely different safe trajectory. The dimensions of the environment are 116x14 m (the cell resolution is 12 cm). The image resolution is $966 \times 120$ pixels. For this environment (lab floor) the first step (the Voronoi extraction) takes 0.05 s in a Pentium 4 at 2.2 Ghz, and the second step (Fast Marching) takes 0.15 s for a long trajectory, which makes a total of 0.20 s to apply the Voronoi FM method, as shown in table 1.

The proposed method is highly efficient from a computational point of view because it operates directly over a 2D image map (without extracting adjacent maps), and due to the fact that Marching complexity is $O(n)$ and the Voronoi path calculation is also of complexity $O(n)$, where $n$ is the number of cells in the environment map.

A great advantage of the method presented is that it can work with any kind of shaped objects, as demonstrated in Fig. 16. The curvature in the shape of the obstacle is not a restriction for the application of the method, and nor is the presence of concavities.

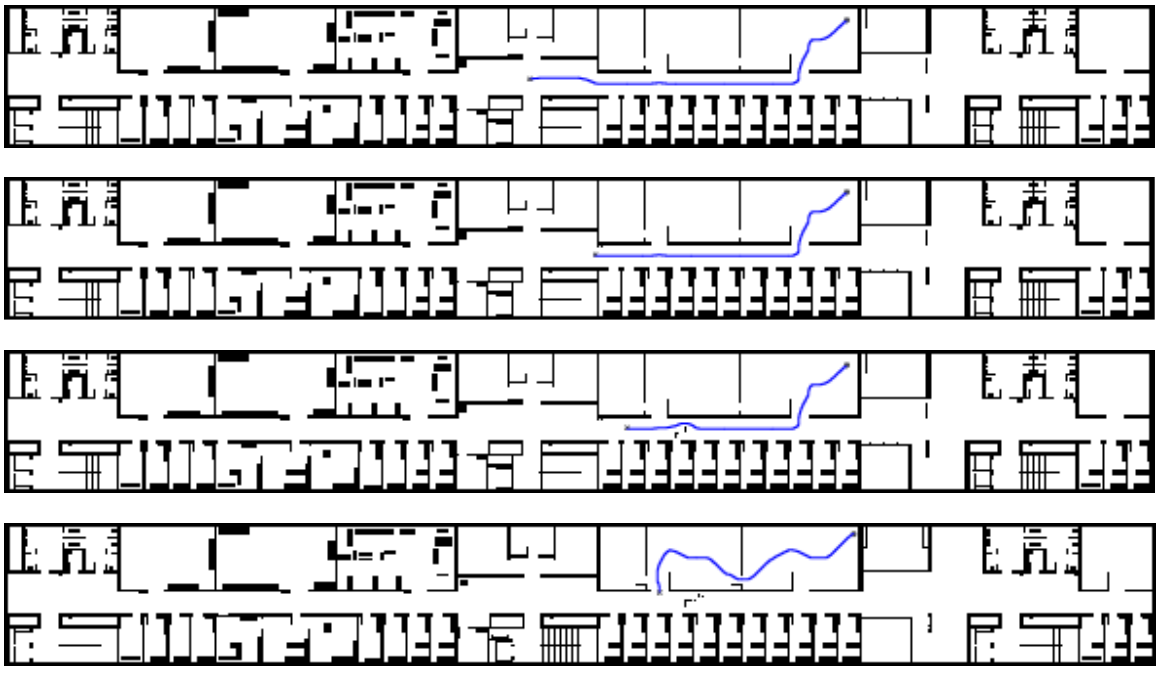


**FIGURE 14**: Evolution of the path when the robot reads information about the new obstacles that are not in the previous map and the robot can no pass throught the corridor

**TABLE 1:** Time comparison with other navigation Methods

| Method | Laser(100x100 pixels) | Room(628x412 pixels) | Lab floor(966x120 pixels) |
|---|---|---|---|
| Voronoi FMM | 0.017 s | 0.446 s | 0.20 s |
| Finite Differences | 1.72 s | 164.8 s | 55.6 s |
| Finite Elements | 1.34 s | 33.6 s | 16.2 s |

**FIGURE 15**: Voronoi Diagram and trajectory in an environment with curved shapes

In Fig. 16, the proposed method has been applied to a open environment, that is to say, to a environment map that is not limited totally by walls, in which there are circular obstacles. This map represents a corner with columns. As can be seen in the figure, the algorithm is perfectly able to make a correct planning of trajectories.

### 6.1 Comparison With Other Navigation Methods

One interesting method that calculates exact navigation functions is the Harmonic Functions method. Harmonic functions provide optimal potential maps for robot navigation in the sense of minimum length over the potential surface and they are the solutions of the Laplace equation $\nabla^2\Phi(x) = 0$. It has been implemented by using finite differences [43,44] and the finite elements method (FEM) [17] with a high level (usually 1) for the boundary conditions for walls and obstacles and a low level (usually 0) for objective. The finite differences method has been used to solve Laplace's equation with the successive over-relaxation (SOR) method, which is an iterative method and, consequently very slow. The finite elements method (FEM) is particularly useful when a robust approximation is sought to solve partial differential equations on non-homogeneous mesh. Solid mathematical foundations and a great deal of generality allow different implementations.

In table 1 a comparison between the different exact navigation methods is shown. This table compares the SOR method, the Finite Elements Method and the proposed Voronoi Fast

Marching method. As can be seen, the difference in time in comparison with the other methods is very significant.

### 6.2 Comparison With Other Navigation Method

Graph-based methods, such as Dijkstra, $A^*$, or $D^*$, can be considered to compute a navigation function constrained to movement choices along graph edges or discrete transitions between grid cells. They thus produce paths that are not optimal for execution. The Fast Marching method proposed gives a navigation function that measures distances in the continuous domain, due to its intrinsic interpolation (see Fig. 2).

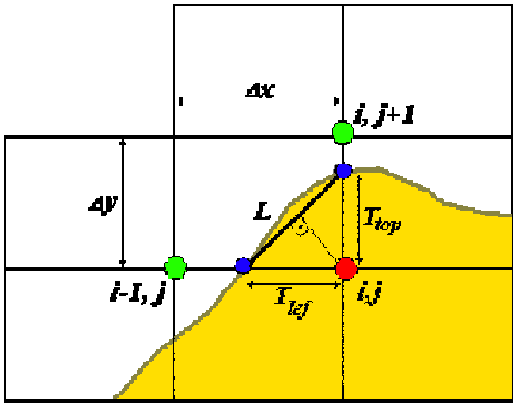



**FIGURE 17**: Calculation of Tij as distance of the point (i,j) to the line L (interpolation)

The graph methods are quite fast but they work in a graph. In our application the original map and the skeleton are binary images and they need to be transformed into its corresponding graph.
In table 2 a comparison with Dijkstra's and $A^*$ methods is shown. As can be seen, these methods need a long time for the construction of the graph. The proposed method has also the advantage that it works in the continuous domain and its paths are smoother and not close to obstacles.

**Table 2. Time comparison with Graph Methods**

| Method | Graph Generation time | Room(628x412 pixels) | Total time |
|---|---|---|---|
| Voronoi FMM | 0.00 s | 0.446 s | 0.446 s |
| Dijkstra | 2.82 s | 0.845 s | 3.665 s |
| A* | 2.82 s | 0.001 s | 2.821 s |

### 6.3 Extension to 3D

The proposed method works also in 3D cases. But not only that the Fast Marching Method can work also in higher dimensionality cases. The process is shown in Figure 3. The fist step is to extract the 3D skeleton, that could be considered as the Voronoi transform. That is done cropping the obstacles until the skeleton is obtained. Once that is done, the Fast Marching Method is launched. Fast Marching works for cases of 3,4 or even more dimensions. As in the two-dimensional case, the skeleton is interpreted as the boundary condition for the wave propagation. The process is shown in Figure 3, and as we can see the algorithms works well in the 3D-case.

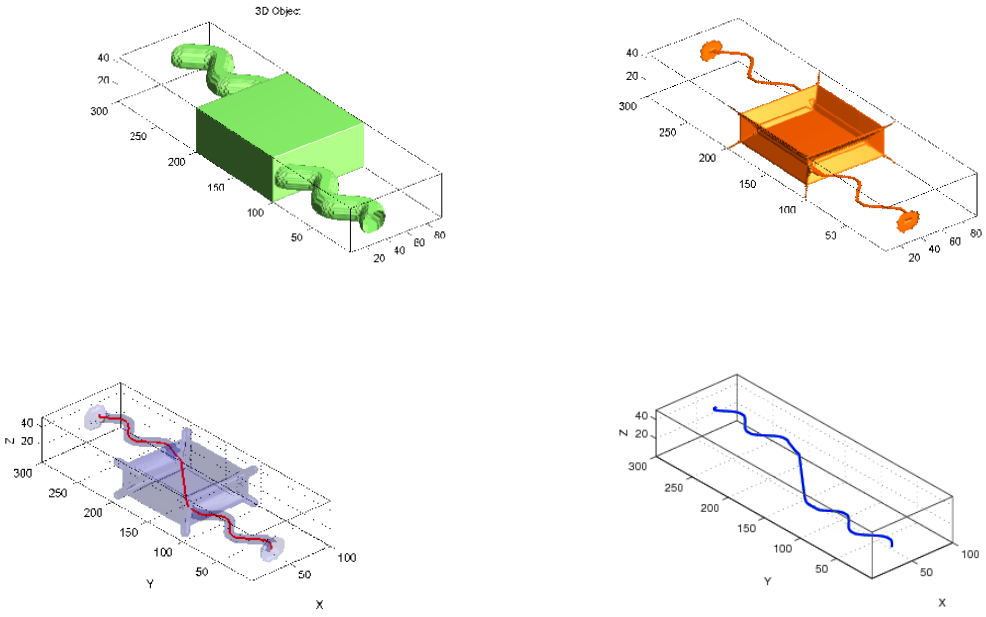



**FIGURE 17**: Trajectory calculated with the proposed method in 3D

## 7. CONCLUSION

A sensor-based path planner is presented in this paper. The proposed method is able to deal simultaneously with both global and local planning requirements. The advantages of the approach can be summarized by the fact that the trajectories obtained are smooth and safe, and at the same time, free of local traps due to the integration of the real-time sensor information in the recalculation of the path.

The method is easy to implement, the algorithm is very fast and can work online. It works in cluttered and changing environments with moving obstacles. The method is complete, i.e., the method is capable of finding a trajectory if it exists.

As demonstrated along this work, the method can perform in all types of environments without restrictions in the form of the obstacles. The planner works with curved forms, open environments (not totally enclosed by walls), and concavities.

The algorithm complexity is $O(n)$, where $n$ is the number of cells in the environment map, which let us use the algorithm on line.

## 8. REFERENCES

[1] D. Adalsteinsson and J. Sethian, "A fast level set method for propagating interfaces," *J. Comput. Phys.*, vol. 118, no. 2, pp. 269–277, (1995).

[2] F. Aurenhammer. "Voronoi Diagrams: A survey of a fundamental Geometric Data Structure," *ACM Computing Surveys*, no. 23, 345-405, (1991).

[3] F. Aurenhammer and R. Klein, *Chapter 5 in Handbook of Computational Geometry*. Eds. J.R. Sack and J. Urrutia, pp. 201–290, (2000).

[4] H. Alt and O. Schwarzkopf, "The Voronoi Diagram of curved objects," in *Proc. 11th Annual ACM Symposium on Computational Geometry*, pp. 89–97, (1995).

[5] M. de Berg, M. van Krefeld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms ans Applications, 2nd rev.*. Springer, (2000).

[6] H. Blum, "A transformation for extracting new descriptors of shape," in *Models for Perception of Speech and Visual Form*, W. W. Dunn, Ed.MIT Press, Cambridge, Mass., pp. 153–171, (1967).

[7] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman, "Linear time euclidean distance transform algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 529–533, (1995).

[8] C. S. Chiang. "The Euclidean Distance Transform," *Ph. D. Thesis, Dept. Comp. Sci., Purdue University*, (1992).

[9] H. Choset. "Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph," *PhD thesis, California Institute of Technology, Pasadena, California*, March (1996).

[10] H. Choset and Burdic. "Sensor-Based Exploration: The Hierarchical Generalized Voronoi Graph," *The International Journal of Robotics Research*, vol. 19, pp. 96–125, (2000).

[11] H. Choset et al., *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, (2005).

[12] J. L. Davis, *Wave propagation in solids and fluids*. Springer, (1988).

[13] E. Dijkstra, " A note on two problems in conexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269-271, (1959) .

[14] D. Ferguson and A. Stentz, *Advances in Telerobotics Field D*: An Interpolation-Based Path Planner and Replanner,* Springer Berlin, pp. 239-253, (2007).

[15] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Automat. Mag.*, vol. 4, pp. 23–33, (1997).

[16] N. Gagvani and D. Silver, "Parameter controlled skeletonization of three dimensional objects," *Technical Report CAIP-TR-216, Rutgers State University of New Jersey*, http://citeseer.ist.psu.edu/gagvani97parameter.html, (1997).

[17] S. Garrido, L.Moreno, "Robotic navigation using harmonic functions and finite elements," in *Intelligent Autonomous Systems IAS'9*, pp. 94–103, (2006).

[18] S. Garrido, L. Moreno, D. Blanco, and F. Martin, "Voronoi Diagram and Fast Marching applied to Path Planning." in *Proc of IEEE International conference on Robotics and Automation, ICRA'06*. pp.3049-3054. (2006).

[19] S. Garrido, L. Moreno and D. Blanco. "Sensor-based global planning for mobile robot navigation." in *Robotica*. vol. 25. pp.189-199. (2007).

[20] S. Garrido, L. Moreno, and D. Blanco, *Exploration of a cluttered environment using voronoi transform and fast marching method*, Robotics and Autonomous Systems, doi:10.1016/j.robot.2008.02.003 (2008).

[21] M. Held. "Voronoi Diagrams and Offset Curves of Curvilinear Polygons." Computer Aided Design, (1997).

[22] S.S. Keerthi, C.J. Ong, E. Huang, and E.G. Gilbert, "Equidistance diagram- a new roadmap method for path planning," *In Proc. IEEE Int. Conf. On Robotics and Automation*, pp 682-687, (1999).

[23] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, pp. 90–98, (1986).

[24] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1398–1404, (1991).

[25] S. Koenig and M. Likhachev, " Improved fast replanning for robot navigation in unknown terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (2002).

[26] J.-C. Latombe, *Robot motion planning*. Dordrecht, Netherlands: Kluwer Academic Publishers, (1991).

[27] D.T. Lee, "Medial Axis Transform of a Planar Shape," *IEEE Trans. Pattern Anal.Mach. Intell.,* PAMI-4, pp. 363-369, (1982).

[28] M. Likhachev, G. Gordon, and S. Thrun," Advances in Neural Information Processing Systems ARA*: Anytime A* with provable bounds on sub-optimality ," *MIT Press*, (2003).

[29] S. R. Lindemann and S. M. LaValle," Simple and efficient algorithms for computing smooth, collision-free feedback laws". International Journal of Robotics Research, (2007).

[30] S. R. Lindemann and S. M. LaValle, "Smooth feedback for car-like vehicles in polygonal environments". In Proceedings IEEE International Conference on Robotics and Automation, (2007).

[31] R. Mahkovic and T. Slivnik "Generalized local Voronoi Diagram of visible region. *In Proc. IEEE Int. Conf. On Robotics and Automation*, pp. 349-355, Leuven, Belgium, May (1998).

[32] S. Mauch, "Efficient algorithms for solving static Hamilton-Jacobi equations," Ph.D. dissertation, California Inst. of Technology, (2003).

[33] P. Melchior, B. Orsoni, O. Laviale, A. Poty, and A. Oustaloup, "Consideration of obstacle danger level in path planning using A* and fast marching optimisation: comparative study," *Journal of Signal Processing*, vol. 83, no. 11, pp. 2387–2396, (2003).

[34] J. Minguez and L. Montano, "Nearness diagram navigation: collision avoidance in troublesome scenarios," *IEEE Trans. Robot. and Automat.*, vol. 20, pp. 45–59, (2004).

[35] K. Nagatani, H. Choset, and S. Thrun. "Towards exact localization without explicit localization with the generalized voronoi graph," *In Proc. IEEE Int. Conf. On Robotics and Automation*, pp. 342-348, Leuven, Belgium, May (1998).

[36] A. Nash, K. Danie, S. Koenig, and A. Felner, "Theta*: Any-Angle Path Planning," *on Grids Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, p. 1177-1183, (2007).

[37] N. Nilsson, "Principles of artificial intelligence," Palo alto, CA: Tioga Publishing Company. (1980).

[38] R. Ogniewicz and O. Kubler, "Hierarchic Voronoi Skeletons," *Pattern Reconition*, (1995).
[39] A. Okabe, B. Boots, and K. Sugihara, "Spatial Tessellations: Concepts and Applications of Voronoi Diagrams". John Wiley and Sons, Chichester, UK, (1992).

[40] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Trans. on Robotics*, vol. 23, no. 2, pp. 331–341, (2007).

[41] R. Philippsen, "An interpolated dynamic navigation function," in *2005 IEEE Int. Conf. on Robotics and Automation*, (2005).

[42] A. Oustaloup. A. Poty, and P. Melchior, "Dynamic path planning by fractional potential," in *Second IEEE International Conference on Computational Cybernetics*, (2004).

[43] E. Prestes. Jr., P. Engel, M. Trevisan, and M. Idiart, "Exploration method using harmonic functions," *Journal of Robotics and Autonomous Systems*, vol. 40, no. 1, pp. 25–42, (2002).

[44] E. Prestes. Jr., P. Engel, M. Trevisan, and M. Idiart, "Autonomous learning architecture for environmental mapping," *Journal of Intelligent and Robotic Systems*, vol. 39, pp. 243–263, (2004).

[45] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 802–807, (1993).

[46] S. Quinlan and O. Khatib. "Efficient distance computation between non-convex objects," in *IEEE Int. Conf Robot and Autom.*, pp. 3324-3329, (1994).

[47] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Nat. Acad Sci. U.S.A.*, vol. 93, no. 4, pp. 1591–1595, (1996).

[48] J. A. Sethian, "Theory, algorithms, and applications of level set methods for propagating interfaces," *Acta numerica*, pp. 309–395, (1996).

[49] J. Sethian, *Level set methods*. Cambridge University Press, (1996).

[50] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3375–3382, (1996).

[51] R. W. Smith, "Computer processing of line images: A survey", *Pattern Recognition*, vol. 20, no. 1, pp. 7-15, (1987).

[52] A. Stentz, "The focused D* Algorithm for Real-Time Replanning," *Proceedings of the Internatinal Joint Conference on Artificial Intelligence (IJCAI)*, (1995).

[53] N. Sudha, S. Nandi, and K. Sridharan, " A parallel algorithm to construct Voronoi Diagram and its VLSI architecture," *In Proc. IEEE Int. Conf. On Robotics and Automation*, pages 1683-1688, (1999).

[54] J. N. Tsitsiklis, "Efficient Algorithms for Globally Optimal Trajectories," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1528-1538, (1995).

[55] I. Ulrich and J. Borenstein, "Vfh+: reliable obstacle avoidance for fast mobile robots," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1572–1577, (1998).

[56] I. Ulrich and J. Borenstein, "Vfh*: local obstacle avoidance with look-ahead verification," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2505–2511, (2000).

[57] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap Planner with sampling on the medial axis of the free space". *In Proc. IEEE Int. Conf. On Robotics and Automation*, pp. 1024-1031, (1999).

[58] L. Yang and S. M. LaValle, "A framework for planning feedback motion strategies based on a random neighborhood graph". In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 544–549, (2000).

[59] L.Yatziv, A. Bartesaghi, and G.Sapiro, "A fast O(n) implementation of the fast marching algorithm," *Journal of Computational Physics*, vol. 212, pp. 393–399, (2005).

# A Robotic Prototype System for Child Monitoring

**Yanfei Liu** liu@engr.ipfw.edu
*Department of Engineering*
*Indiana University – Purdue University*
*Fort Wayne, 46805-1499, US*

**Abstract**

Child monitoring systems can use different technologies, such as camera systems, wireless technology using RFID sensors, and GPS based systems. These systems are capable of remotely reporting the status of the children but not able to take actions based on the interpretation of the scenarios. A robotic child monitoring system has the advantage that it can take actions to warn and possibly protect the child from danger. In this paper we present the design and experimental testing of a robotic child monitoring prototype system. The whole system consists of a Khepera robot, a host computer, the distraction/alarm circuitry and a testing table. The experimental testing results show that this prototype system fulfills the requirement of finding and following the baby prop and also taking certain actions when the baby prop approaches a danger area.

**Keywords:** Autonomous Robots, Household Robots, Child Monitoring.

## 1. INTRODUCTION

Different technologies have been used to implement child monitoring systems to release parents from continuous observation of their children. These techniques include camera systems, wireless technology using RFID sensors, and GPS systems. Cooper *et al.* [1] collected data about the outdoor physical activity of over 1000 children to study their behavior patterns. The data was collected through accelerometers and GPS receivers. Al-Ali *et al.* [2] described a Kids Tracking System using RFID sensors. The system was designed to track moving children in an area such as a large park or a shopping center. The children to be tracked were given an RFID tag to wear. Several wireless RFID readers were installed in the park. These readers sent the locations of the children to a central station through wireless LANs. Rai *et al.* [3] described a multiple camera system using FireWire web cameras. One of the potential applications for this system was to track the position of a moving person. The aforementioned systems are examples of three technologies that have been used in child monitoring systems. There has been also research work that combined some of the aforementioned techniques. Jawad *et al.* [4] presented a child tracking system that used GPS outdoors and RF signals indoors to provide the locations of children. Nakagawa *et al.* [5] developed a system that used multi-camera system and RFID to monitor children. Based on the information provided from the RFID tags, the parents can choose the camera that will take and transmit the images. Ishikawa *et al.* [6] described a system that combined an omni-directional camera and a GPS receiver to remotely monitor a child. This system allows the parents to open Google Maps using the position provided by the GPS system. The images transmitted from the omni-directional cameras would compensate for the measurement error of the GPS system. Sampath and Sundaram [7] described a mobile unit that combined an omni-directional camera, a basic health monitoring gadget module, and wireless transceivers to serve as a remote monitoring system for children. Wong *et al.* [8] proposed a child care system that combined GPS and Bluetooth technologies. The system detects the location of the child with the GPS system and the distance between the child and the parent through received the signal strength indicator (RSSI) from Bluetooth.

For decades, people have viewed robotics as a high tech topic. Today robots are widely applied in industry, military field and even in space explorations. Robots are beginning to directly impact people's everyday life. Robots have evolved into different varieties, such as surgical robots (Hockstein *et al*. [9]), and they have slowly entered the household. The household robots vary from entertaining robots (Fujita [10]; Inoue *et al.* [11]), cleaning robots (Prassler *et al.* [12]), to carebots (Salvini *et al.* [13]). Robots also have great potential for child monitoring. Integrating an autonomous robot with the aforementioned techniques, such as cameras and wireless communication, would add to the total cost of the system. However, compared with other child monitoring systems, a robotic system would have the following advantages. First, the mobility of a robot can greatly enlarge the playing area of a child both indoors and outdoors. Second, a robotic system will have more processing power to deal with complex household areas. Third, a robotic child monitoring system can communicate and take actions to warn and possibly protect the child from danger. All of the systems mentioned above are passive systems which cannot take actions based on the scenarios. This paper describes an effort in Department of Engineering at Indiana University – Purdue University Fort Wayne to bring robots closer to people. As the outcome of this project, a robotic prototype system was developed whose function is to assist parents with child monitoring tasks.

This research task was carried out by two senior electrical engineering students. Therefore the initial large scope of the project had to be simplified into an appropriate scope and level of difficulty for a senior design project. The simplifications include sizing down the whole system, using Lego pieces to simulate a child and other furniture in the room, and a basic artificial intelligent design. The objective is to set a starting point in the development of a mobile child monitoring system in the household.

In this prototype system, the robot is capable of finding and following a 7-10 month old baby prop (crawling age) in a confined space. This system also features an artificial intelligence component in order to determine the danger level that the child might be in and take actions accordingly. Various algorithms were generated to accomplish the tasks of target finding, object tracking and obstacle avoidance. The prototype includes a Khepera II robot [14], a scaled down model of a room with a total area of 2.88 $m^2$ (32 sq. ft.). This room contains a baby prop, built with yellow Lego blocks, and three obstacles created using small Lego pieces. With the equipped camera, the robot is capable of processing images to find and follow the baby prop. The artificial intelligence capabilities include activating a distraction light circuit as well as an alarm warning. The software development was conducted using KTProject [15] which allows programming in C.

The remainder of this paper is organized as follows. In Section 2, an overview of the prototype system is described. In Section 3, the detailed design of the robotic system is presented. Testing procedures and results are presented in Section 4. Finally, Section 5 has the conclusions.

## 2. SYSTEM OVERVIEW

Figure 1 shows a diagram of the initial designed prototype. Our prototype system consists of a Khepera robot II, a host computer, the distraction/alarm circuitry and a testing table. Figure 2 shows a diagram of the input/output system in our prototype. The Khepera robot receives sensing information from 8 IR (infrared) sensors mounted around the perimeter to detect any obstacle. It also uses information from the camera to find the baby prop and any obstacles. The robot uses the sensed information to determine the danger level of the baby and then activates the flashing LED/alarm circuits as needed. Figure 3 shows a picture of the actual system and the testing area. The Khepera robot II is 70mm in diameter and 100mm in height. The host computer is used for the code writing, downloading the code to the robot, and viewing the image transmitted from the camera. The host computer is connected with the robot through the interface/charger module using a standard RS232 cable. The interface/charger module converts the RS232 signal into an S serial signal to communicate with the robot. The serial cable between the robot and the host computer is only used for displaying information/images. The robot can be fully autonomous

when in action. The robot activates the distraction/alarm circuitry using a wireless transmission to the radio base.

A testing field was built to simulate a baby's scaled down playpen area. The field was a table with approximate dimensions of 1.2m X 2.4m (4ft. X 8ft.). There is a red outer marking to indicate the out of bounds area and a blue inner marking to indicate the warning area (Figure 3). The red marking encompasses the playpen and has a total surface area of 1.3m$^2$ (14sq.ft.). The blue marking is 5cm to the inside of the red line and it also encompasses the playpen area. Additionally, a white vertical boundary of height 0.3m (1ft) on the table's perimeter was created as a wall.

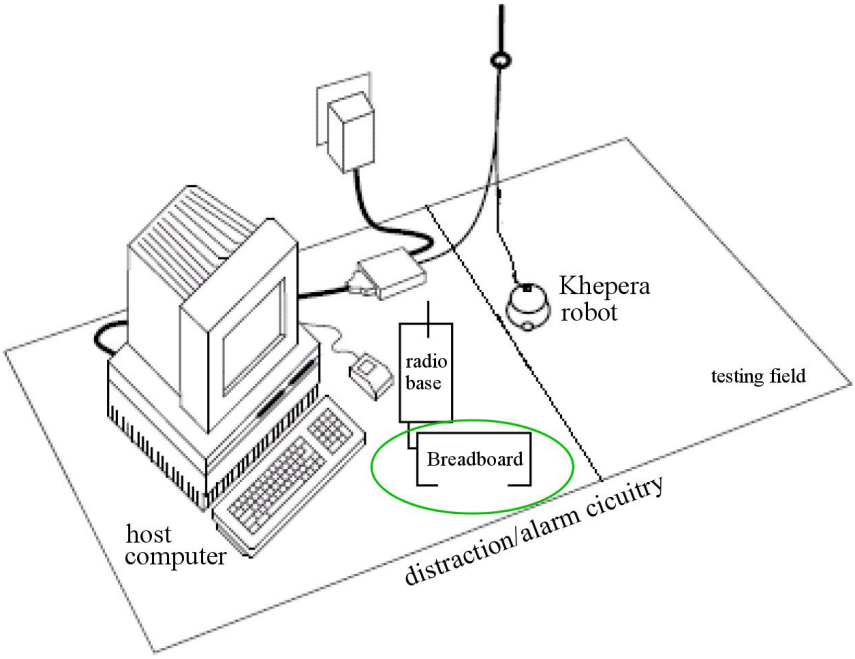



**FIGURE 1:** Robot and host computer configuration.

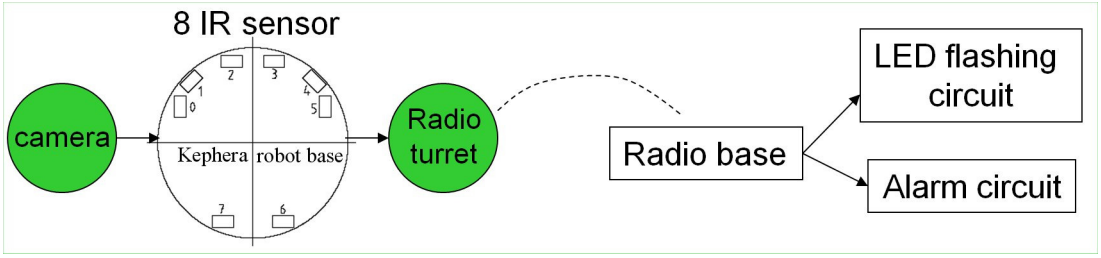



**FIGURE 2:** System diagram.

## 3. DETAILED DESIGN

### 3.1 Robot Set

In our prototype, the robot is the Khepera II [14]. The robot has a Motorola 68331 microcontroller with 512K RAM and flash memory on board. The robot was assembled with the following turrets, mounted in the given order, from bottom to top: Base unit, General I/O turret, Radio turret and the Camera turret. Figure 4 shows the pictures of each individual turret, the assembling order and the final robot. To attach these turrets to the robot base, they have to be placed on the extension

connector with all the pins seated correctly. The serial connection of the radio turret is used by the robot to communicate with the host computer. The K6300 Camera Turret comes equipped with a Freescale MC68331 processor, along with a flash memory. The camera turret holds a V6300 digital CMOS color camera and its optical elements. The color images acquired through this camera are 160 pixels wide and 120 pixels high. A detailed description of the Khepera II robot and the aforementioned turrets can be found in [16].

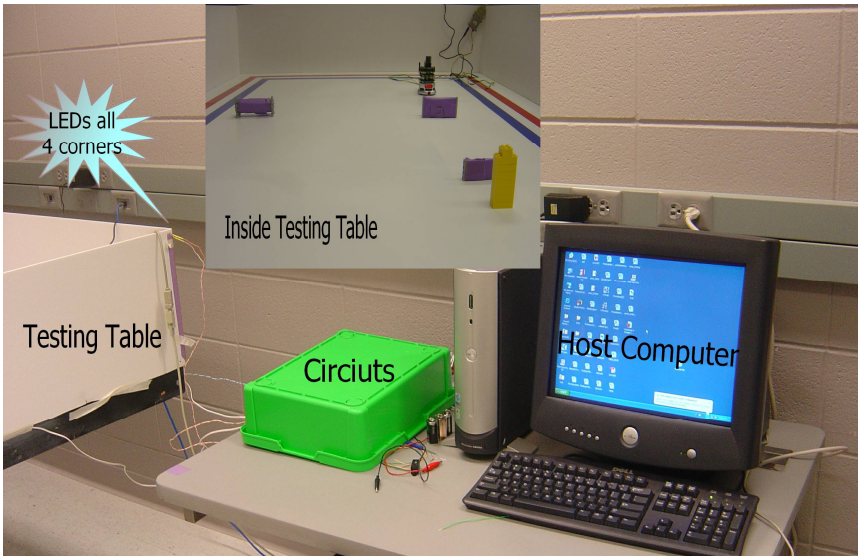



**FIGURE 3:** The actual system and testing area.

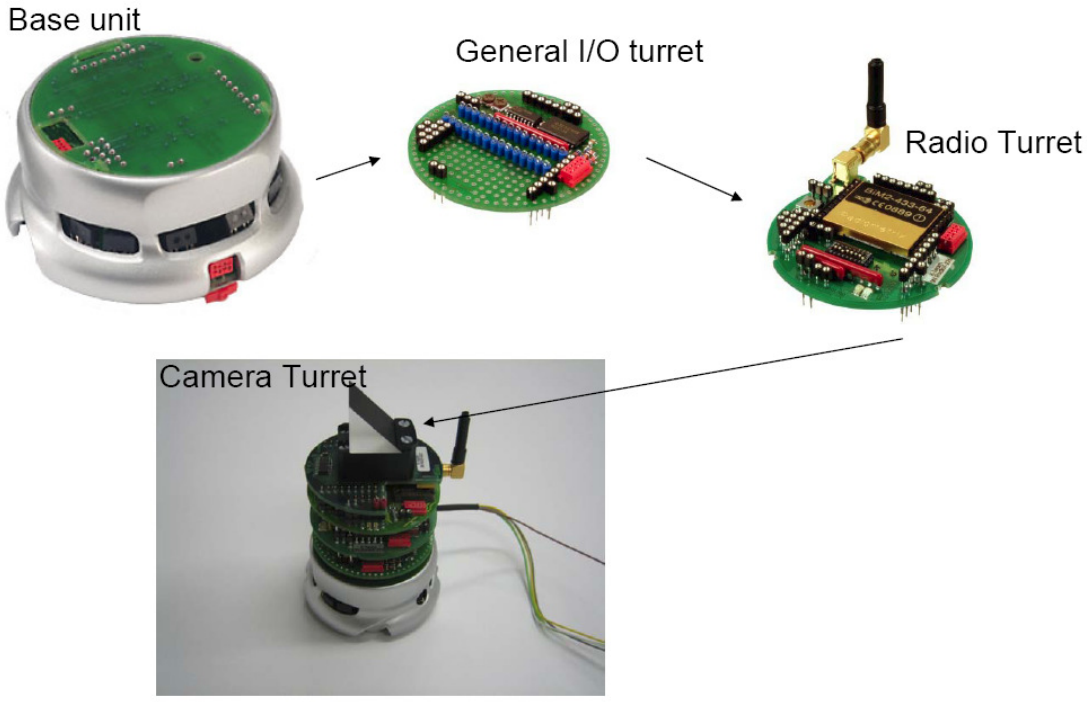



**FIGURE 4:** The composition of the Khepera II robot.

### 3.2 Hardware Design

The hardware design includes a flashing lights circuit and an alarm system. The flashing lights circuit is designed to distract the child from going into dangerous situations. Four bright LEDs are placed in the four corners of the room. This guarantees that the baby will see at least one light no matter which direction the baby is facing. The alarm system is used to alert the parents for three different purposes. First, if the robot doesn't find the baby after searching the entire room, the robot needs to alert the parents. Next, if the robot is trapped between the wall and the baby, i.e. the baby might pick the robot up and the robot has no room to back up, it needs to send an alarm to alert the parents. Finally if the baby is in the out of bounds area the robot needs to notify the parents that the child might be in danger. Figure 5 shows the complete diagram for the LED/alarm circuits. In the next several paragraphs, the function and design of each individual circuit will be briefly described. A more detailed description of these circuits can be found in [17].

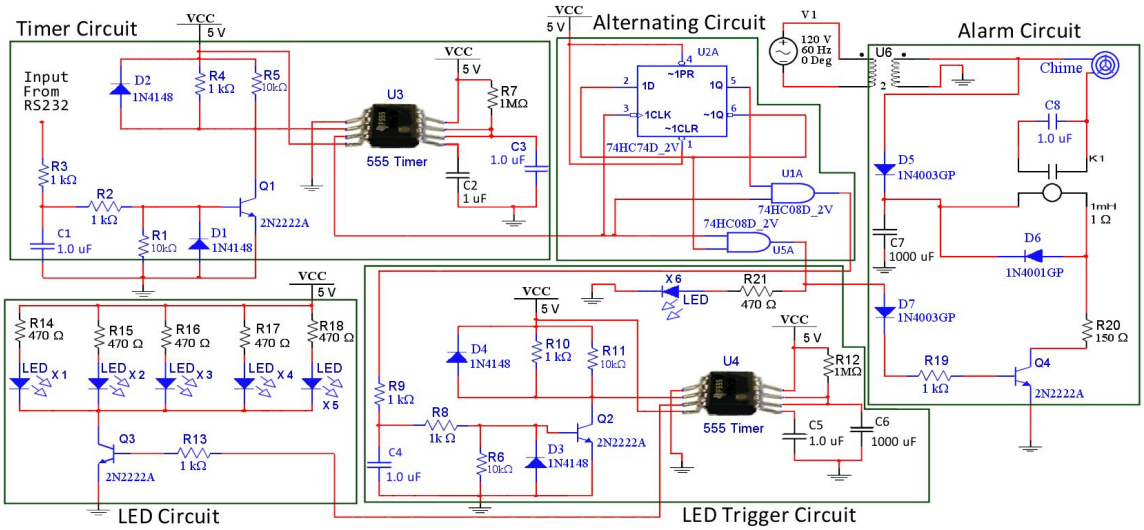



**FIGURE 5:** Complete diagram of the LED/alarm circuits.

The LED/alarm circuits are triggered by the robot sending a signal from the radio turret to the radio base. After the robot sends a signal to the radio base, the same signal is then instantaneously transmitted out from the RS232 port to the radio base. Since the signal goes high for only 10ms, there is not enough time to turn on a transistor and activate necessary hardware components. So, in order to extend the time of the input signal to a few seconds a timer circuit (shown in Figure 5) utilizing a 555 timer IC (integrated circuit) is used.

The output of the timer goes to a 7474 D- flip-flop. This set-up is to properly alternate the circuits (shown in Figure 5) between the LED circuit and the alarm circuit. The $\overline{Q}$ output of the flip-flop is reversed back to the D input and the timer output acts as the clock. The $Q$ and $\overline{Q}$ outputs are ANDed with the clock. This ensures that only one circuit is ON at any particular time.

In the alarm circuit, a high signal is sent to trigger the transistor to start conducting and thus a current begins to flow from the transformer and through the switch. This current flow causes the normally open switch to close and create a connection. The circuit is completed creating a ground connection to the door chime. A current flows through the encasing, retracting the solenoid. When the pulse stops flowing through the transistor, the current stops flowing, and the solenoid is released quickly, hitting the metal strips, creating a chime.

### 3.3 Image Processing and Obstacle Avoidance

The processor on the camera turret is a VV6300 manufactured by VLSI Vision Limited. The camera has an integrated CMOS (complementary metal oxide semiconductor) color image sensor with an on-chip ADC (analog/digital converter). The image resolution is 120X160 (row x column) and stored in the Bayer pattern color pixel array showed in Figure 6. In this pattern each pixel is divided into red, green and blue values that can range from 0 to 255. From Figure 3 we can see that there are four colors needed to be distinguished, yellow (the baby prop), purple (obstacles), and the blue and red lines on the table. Due to the low quality of the images received from the camera, the colors of the actual objects are distorted. So the filtering ranges of these four colors are experimentally determined. With only one of the color object present in front of the camera, the ranges of red, green and blue values in the Bayer pattern were determined. Results of the filtering ranges for the four colors can be found in [17].

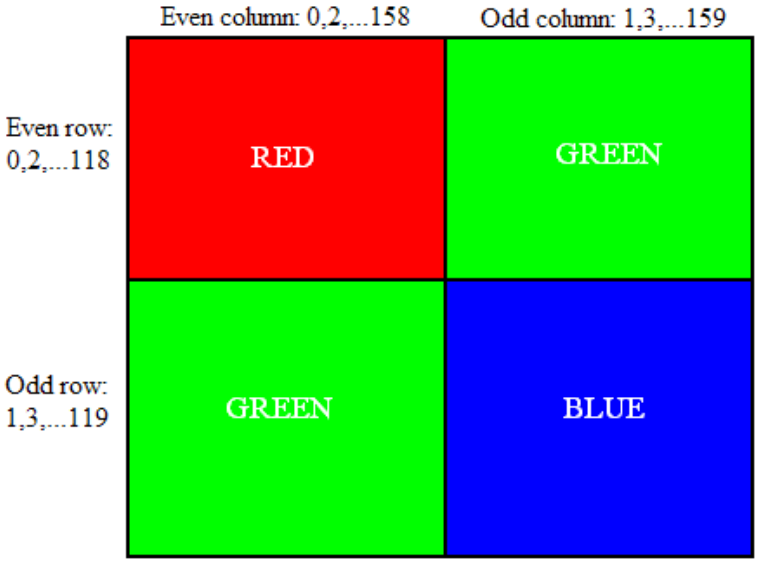



**FIGURE 6:** Bayer filter pattern

The obstacle avoidance was accomplished via both image processing and the IR sensor readings. The obstacle was first detected by using image filtering. Once an obstacle is found, the robot will keep moving forward and checking the results from the IR sensors located around the robots. Once the IR sensors detect the obstacle, the robot will turn right and move forward for 20cm(max length of the biggest obstacle), then turn left and move forward 10cm(max width of the biggest obstacle), then turn left and go forward 20cm again to reach the same distance as started before the obstacle was reached, and then turn left to face in the same direction as was before the obstacle was reached.

## 4. TESTING PROCEDURES AND RESULTS

The baby prop was created by stacking up eight Lego blocks and is 3cm long, 1.5cm wide and 7.5cm high. The dimensions of the obstacles are given in Table 1. Figure 7 also shows a picture of the three obstacles.

| Obstacle # | Length (cm) | Height (cm) | Width/diameter (cm) |
|---|---|---|---|
| 1 | 7.5 | 3 | 0.9 |
| 2 | 10 | 5 | 4 |
| 3 | 10 | 5 | 0.9 |

**TABLE 1:** Dimensions of the obstacles.

Three different scenarios were tested for the prototype system. Each scenario represents one level of complexity of the environment, ranging from no obstacle, only one obstacle to three obstacles.

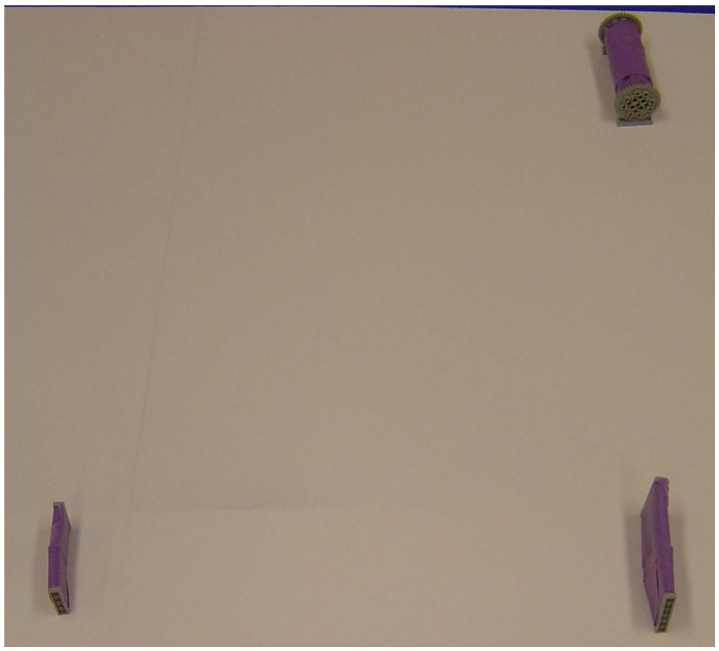


**FIGURE 7:** The obstacles.

### 4.1 Scenario I – no Obstacle

In this scenario there were no obstacles placed in the room. During this test the baby prop was present in the room 5 out of the 10 times. Each time the test was conducted the position and direction of the robot was changed. Figure 8 illustrates the approximate positions of the robot and the baby prop during each of the tests. Figure 9 shows a picture of one of the testing positions of the robot and the baby prop. During the first 5 trials the baby prop was not present, and then the baby prop was placed in different locations in the room for the other 5 trials. The testing results are presented in Table 2. Out of the ten tests, only one false result was produced. Errors are produced because the objects in the image become out of focus when the baby prop is more than 60 cm away from the robot.

| | **Baby found** | **Baby not found** |
|---|---|---|
| **No baby present** | 0 | 5 |
| **Baby present** | 4 | 1 |

**TABLE 2:** Testing results for scenario I.

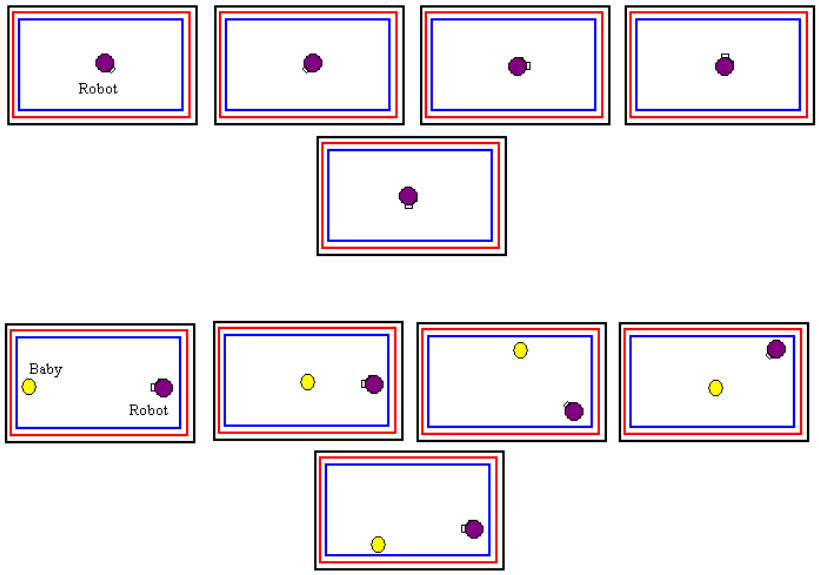



**FIGURE 8:** The locations for the robot and the baby in scenario I

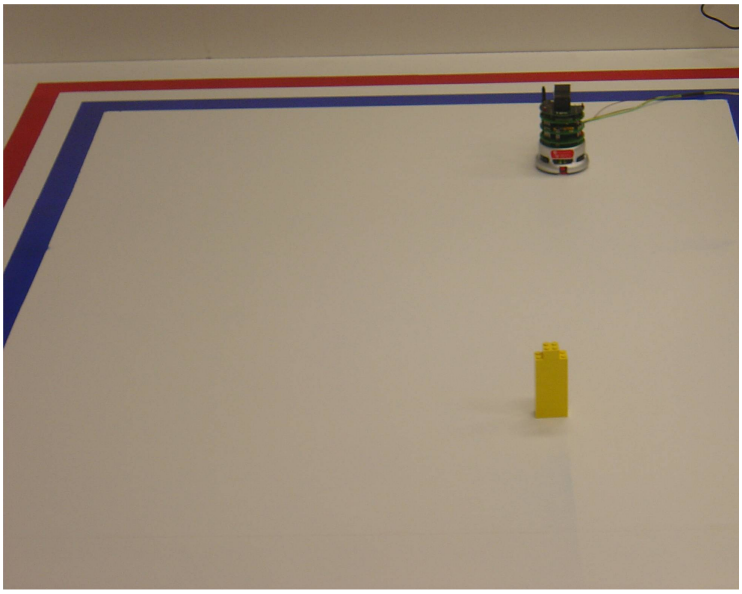


**FIGURE 9:** The robot and baby prop in scenario I.

### 4.2 Scenario II – With one Obstacle

This setup is of an intermediate difficulty level. Thus, even after the robot has located the baby prop there is an obstacle that the robot has to avoid in order to get to the child. The rectangular box shaped obstacle with dimensions 10 cm long, 5 cm high and 0.9 cm wide was placed between the robot and the baby prop. Also, the robot has to use the camera to detect whether the obstacle is present or not. The test was performed 5 times with the obstacle present and 5 times with the obstacle absent. Figure 10 illustrates the approximate positions of the objects during the tests. Figure 11 shows a picture of the testing scenario. Table 3 shows the testing results of whether the obstacle was found. Out of ten tests, only two false results were produced. During each test it is important to determine whether the robot maintains its course or deviates

from it. This determination is needed to assert whether the robot will reach the baby prop after it has passed around the obstacle. Such situation was also tested in this scenario. Once the robot has successfully avoided the obstacle it still needs to determine the baby prop's location in the room and reach it successfully. Table 4 shows the results of whether the robot can reach the baby. Out of ten tests, there were two instances when the robot couldn't reach the baby prop. This is due to two reasons. First, the two motors on the robot do not always move at the same speed. The left motor turns faster than the right motor all the time. This speed difference causes the robot to compute inaccurate results regarding the distance traveled across the table. Second, the low quality camera creates difficulties for the vision algorithm to distinguish between the color of the obstacle, the baby prop, and the blue or red borders. This problem causes the robot to sometimes assume that the baby prop or the boundary lines is an obstacle which make the robot go around the baby prop instead of following it around the room.

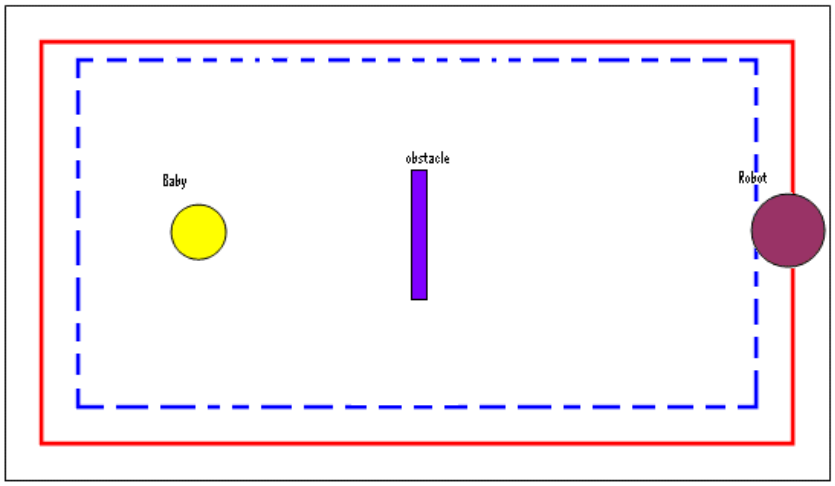



**FIGURE 10:** The top view of scenario II.

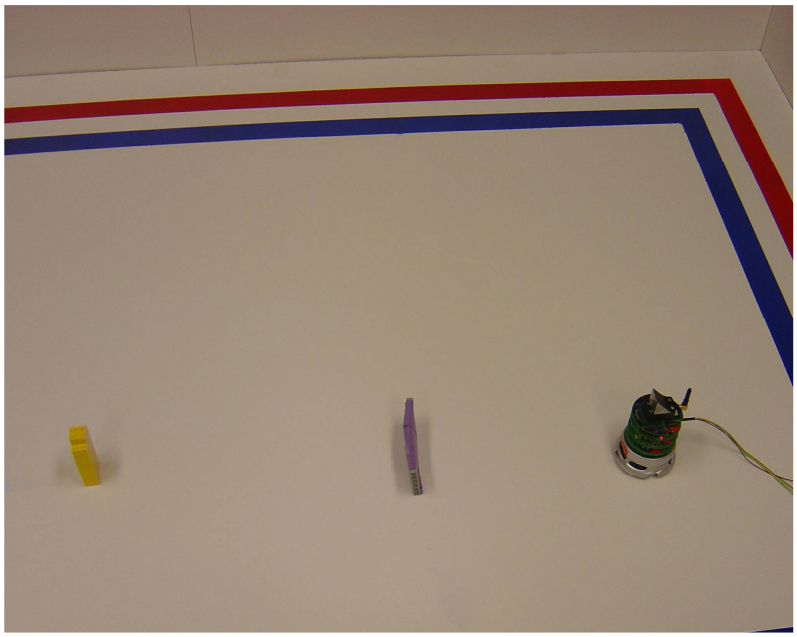


**FIGURE 11:** Testing environment in scenario II.

| | Obstacle found | Obstacle not found |
|---|---|---|
| **Obstacle present** | 5 | 0 |
| **Obstacle absent** | 2 | 3 |

**TABLE 3:** Testing results of finding the obstacle in scenario II.

| | Reached baby | Didn't reach the baby |
|---|---|---|
| **Obstacle present** | 5 | 0 |
| **Obstacle absent** | 3 | 2 |

**TABLE 4:** Testing results of reaching the baby prop in scenario II.

### 4.3 Scenario III – Complex Environment

This is the most difficult situation for the robot. Three obstacles were placed between the robot and the baby prop. Each time the test was run the location of the obstacles stayed the same. But sometimes the obstacle exactly in front of the child was switched with another of a different size and shape. Figure 12 shows a sample layout of one of the test variations. Two out of the 5 times, obstacle 3 was in front of the baby prop. Two times obstacle 2 was present and finally once obstacle 1 was present. The rest of the five times the test was run, there was no obstacle between the robot and the baby. Figure 13 shows a picture of the testing scenario. Table 5 shows the testing results of whether the obstacle was found. Out of ten tests, only one produced a false result. Table 6 shows the results of whether the robot can reach the baby prop. Out of ten tests, there was only one case when the robot couldn't reach the baby prop. The reason is similar as the one for scenario II.

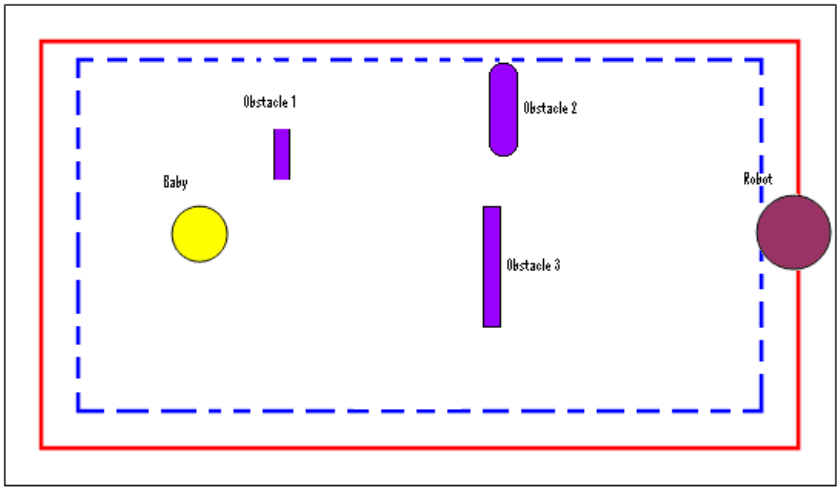



**FIGURE 12:** Top view of scenario III.

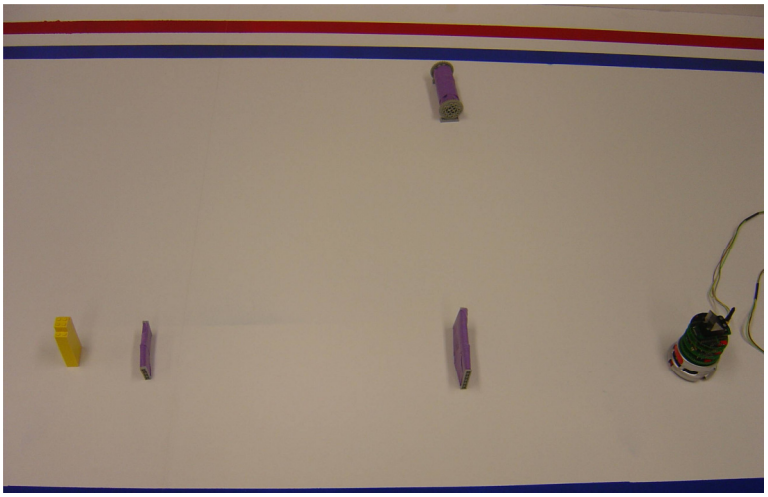


**FIGURE 13:** Picture of the testing environment in scenario III.

| | Found Obstacle | Obstacle not found |
|---|---|---|
| **Obstacle present** | 4 | 0 |
| **Obstacle absent** | 1 | 5 |

**TABLE 5:** Testing results of finding the obstacle in scenario III.

| | Reached Baby | Didn't reach the baby |
|---|---|---|
| **Obstacle present** | 4 | 0 |
| **Obstacle absent** | 5 | 1 |

**TABLE 6:** Testing results of reaching the baby in scenario III.

### 4.4 Distraction and Alarm Circuits

Once the robot finds the baby prop, the robot should follow it around the room. If the baby prop gets close to the blue line the robot should send a signal to distract the baby prop with the help of the flashing LEDs. Also, if the baby prop gets into the out of bounds area the robot should send another signal and activate the alarm to alert the parents. Table 7 and 8 show the results.

| | Lights on | Lights off |
|---|---|---|
| **Within range** | 3 | 0 |
| **Out of range** | 2 | 5 |

**TABLE 7:** Testing results of the LED circuit.

| | Alarm on | Alarm off |
|---|---|---|
| **Within range** | 4 | 1 |
| **Out of range** | 1 | 4 |

**TABLE 8:** Testing results of the alarm circuit.

Other experiments were conducted to test the system's capability. These experiments include the estimation of the camera's field of view, the infrared (IR) sensor's capability and the autonomous behavior of the robots. A detailed description of all of these experiments and the software design can be found in [17].

## 5. CONCLUSIONS

In this paper, the design and experimental testing of a robotic child monitoring prototype system is presented. The whole system consists of a Khepera robot, a host computer, the distraction/alarm circuitry and a testing table. The design of each component was described in detail. Compared with the existing passive child monitoring systems, our system does not require the child to wear any sensor, which means it's safer. Also, our system can warn the parents when the child is in danger. With further development using a more advanced robotic platform, the system can even take actions to prevent the children from danger.

Human interactive systems are always challenging; this system is only a prototype which aims at shedding some lights on a child monitoring robotic system. Therefore it has limitations and drawbacks. For example, the low quality of the image sensor affects the performance in scenarios where more color objects are present. Therefore it is suggested that a better quality camera be used when this type of system is adopted in other robots. Also one robot might not be enough when the environment is more complex than what was experimented in our system. For future work, a network of robots needs to be considered for better performance.

## 6. REFERENCES


1. A. Cooper, A. Page, B. Wheeler, M. Hillsdon, P. Griew, and R. Jago, "*Patterns of GPS measured time outdoors after school and objective physical activity in English children: the PEACH project*," International journal of Behavioral/Nutrition and Physical Activity 2010, 7:31

2. Al-Ali, A.R.; Aloul, F.A.; Aji, N.R.; Al-Zarouni, A.A.; Fakhro, N.H.; "*Mobile RFID Tracking System*," In Proceedings of the 3rd International Conference on Information and Communication Technologies (ICTTA): From Theory to Applications, April 7-11, 2008

3. P. Rai, K. Tiwari, P. Guha, A. Mukerjee, "*A Cost-effective Multiple Camera Vision System using. FireWire Cameras and Software Synchronization*," In Proceedings of the 10th Annual International Conference on High Performance Computing (HiPC 2003)

4. S. Jawad, A. M. Yousef, B. Al-Shagoor, "*A Multipurpose Child Tracking System Design and Implementation*," International Journal of Soft Computing Applications, Issue 4 (2009), pp. 57-68.

5. S. Nakagawa, K. Soh, S. Mine, and H. Saito, "*Image Systems Using RFID Tag Positioning Information*," NTT Tech Rev, Vol.1, NO.7, pp.79-83, 2003

6. T. Ishikawa, K. Yamazawa, T. Kurata, and N. Yokoya, "*Mobile Omnidirectional Monitoring System for Remote Personal Security*," In Proceedings of the 4th International Conference on Collaboration Technologies, August 2008

7. D. Sampath and S. Sundaram, "*Telecommunication system for remote monitoring and access in child care*," in 2008. IET International Conference on Wireless, Mobile and Multimedia Networks, pp. 81-84

8. Kok Sun Wong; Wei Lun Ng; Jin Hui Chong; Chee Kyun Ng; Sali, A.; Noordin, N.K, "*GPS based child care system using RSSI technique*," in 2009 IEEE 9th Malaysia International Conference on Communications (MICC),, pp. 899-904.

9. N. G. Hockstein, , C. G. Gourin, , R. A. Faust, and D. J. Terris, "*A history of robots: from fiction to surgical robotics*," Journal of Robotic Surgery, Vol. 1, No. 2, pg. 113-118, 2007

10. M. Fujita, "*On Activating Human Communications with Pet-Type Robot AIBO*," Proceeding of IEEE, Vol. 92, No. 11, pg. 1804-1813, 2004

11. H. Inoue, and H. Miagoshi, "*Behavior Evolution of Pet Robots with Human Interaction*," 2007 Second International Conference on Innovative Computing, Information and Control, pg. 23-23

12. E. Prassler, A. Ritter, C. Schaeffer, and P. Fiorini, "*A Short History of Cleaning Robots*," Autonomous Robots, Vol. 9, No. 3, pp. 211-226., 2000

13. P. Salvini, C. Laschi, and P. Dario, "*Roboethics in biorobotics: discussion of case studies*," International Conference on Robotics and Automation (ICRA 2007) Workshop on Robo-Ethics, Rome, Italy, 2007

14. K-team home page, http://www.k-team.com/mobile-robotics-products/khepera-ii, last accessed December 2010
15. Khepera II programming manual (version 1.1, March 2000), K-team http://ftp.k-team.com/khepera/documentation/Kh2ProgrammingManual.pdf, last accessed December 2010

16. Y. Liu, C. Griffith, and P. Reddy, "*Baby Bot, a Child Monitoring Robotic System*," in Proceedings of the ASEE Illinois-Indiana and North Central Joint Section Conference, Fort Wayne, IN, USA, March 2006

17. C. Griffith, and P. Reddy, "The BabyBot – Robotic Child Monitoring System," Capstone Senior Design Report II, http://www.engr.ipfw.edu/capstone/sd_2_fall06/Baby_Bot_FINALReport.pdf , last accessed December 2010

# INSTRUCTIONS TO CONTRIBUTORS

Robots are becoming part of people's everyday social lives - and will increasingly become so. In future years, robots may become caretaking assistants for the elderly, or academic tutors for our children, or medical assistants, day care assistants, or psychological counselors. Robots may become our co-workers in factories and offices, or maids in our homes.

The International Journal of Robotics and Automation (IJRA), a refereed journal aims in providing a platform to researchers, scientists, engineers and practitioners throughout the world to publish the latest achievement, future challenges and exciting applications of intelligent and autonomous robots. IJRA is aiming to push the frontier of robotics into a new dimension, in which motion and intelligence play equally important roles. IJRA scope includes systems, dynamics, control, simulation, automation engineering, robotics programming, software and hardware designing for robots, artificial intelligence in robotics and automation, industrial robots, automation, manufacturing, and social implications.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJRA.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 2, 2011, IJRA appears in more focused issues. Besides normal publications, IJRA intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

**IJRA LIST OF TOPICS**

The realm of International Journal of Robotics and Automation (IJRA) extends, but not limited, to the following:

- Automation Control
- Autonomous Robots
- Emergence of The Thinking Machine
- Household Robots and Automation
- Jacobian and Singularities
- Nanotechnology & Robotics (Nanobots)
- Robot Controller
- Robotic & Automation Software Development
- Robotic Surgery
- Robotic Welding
- Robotics Programming
- Robots Society and Ethics
- Spatial Transformations
- Unmanned (Robotic) Vehicles
- Automation Engineering
- Biotechnology & Robotics
- Forward Kinematics
- Inverse Kinematics
- Methods for Teaching Robots
- Orientation Matrices
- Robot Structure and Workspace
- Robotic Exploration
- Robotic Surgical Procedures
- Robotics Applications
- Robotics Technologies
- Software and Hardware Designing for Robots
- Trajectory Generation

**CALL FOR PAPERS**

**Volume:** 2 - **Issue:** 3 - May 2011

**i. Paper Submission:** May 31, 2011 **ii. Author Notification:** July 01, 2011

**iii. Issue Publication:** July /August 2011

## CONTACT INFORMATION

**Computer Science Journals Sdn BhD**
M-3-19, Plaza Damas Sri Hartamas
50480, Kuala Lumpur MALAYSIA

Phone: 006 03 6207 1607
006 03 2782 6991

Fax: 006 03 6207 1697

Email: cscpress@cscjournals.org

CSC PUBLISHERS © 2011
COMPUTER SCIENCE JOURNALS SDN BHD
M-3-19, PLAZA DAMAS
SRI HARTAMAS
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6207 1607
006 03 2782 6991

FAX: 006 03 6207 1697
EMAIL: cscpress@cscjournals.org