American J. of Engineering and Applied Sciences 2 (4): 611-619, 2009
ISSN 1941-7020
© 2009 Science Publications

Vision-Based Obstacle Avoidance of Mobile Robot Using Quantized Spatial Model

Salaheddin Odeh, Rasha Faqeh, Laila Abu Eid and Nihal Shamasneh
Department of Computer Engineering, Faculty of Engineering, Al-Quds University,
P.O. Box 20002, Abu Dies, Jerusalem, Palestine

Abstract: Problem statement: Problem of moving a robot through unknown environment has attracted much attention over past two decades. Such problems have several difficulties and complexities that are unobserved, besides the ambiguity of how this can be achieved since a robot may encounter obstacles of all forms that must be bypassed in an intelligent manner. This research had been aimed to develop a system that was able to detect obstacles in a mobile robot's path using a single camera as only sensory input and to achieve the target point in optimized manner. For this reason, algorithm which took total path length and safety into account was developed. **Approach:** To control movement of robot from a starting to a target point inside the site where obstacles can obstruct the way of robot, real-time software-specially tailored for this purpose-was necessary to develop. To analyze and to process scene images captured by a (vision) camera, camera was installed at the top over the center of site in a way that it covered whole site through which sufficient image information could be delivered. From sequentially captured images that was manipulated through image processing and computer vision, the system built a representative site model, whose ingredients were gridded squares as a result of quantized spatial plane of site and then it began planning the desired routing path. **Results:** For building a robot path, less computing effort was necessary because grid information was much easier to deal with than pixels and only a minimum amount of stored data of symbolic site model from current and previous state was necessary. **Conclusion:** Using a quantized spatial domain, a less computational effort was necessary to control movement of robot with the ability of obstacle detection and avoidance.

Key words: Robotics, obstacle avoiding, image processing, computer vision

INTRODUCTION

Generally, vision-based robotic systems with the ability of obstacle detection and avoidance are relatively complicated since extracting information from a stream of the site images consisting of the robot and the obstacles can be a very complex task to achieve real-time performance with as little computing processing as possible. The problem of moving a robot through unknown environment has attracted much attention over the past two decades. Although at first glance, the problem goal sounds simple like “Move the robot from a start position to a desired destination”, several difficulties and complexities are unobserved, besides the ambiguity of how this can be achieved. A robot may encounter obstacles of all forms that must be bypassed in an intelligent manner. Accordingly, a great deal of our research focuses on the use of computer vision to achieve a vision-based autonomous mobile robotic system capable to be navigated within its environment through logically acting on the sensed data to avoid such obstacles. The robot tries to locate hindering obstacles, both stationary and moveable, plans ways to bypass these objects and, finally, acts according to the resulted plan.

According to Picardi *et al*.[1], computer vision is the branch of artificial intelligence that focuses on providing computers with the functions typical of human vision. Currently, there are many computer-vision based applications that have been produced in fields such as: Industrial automation, robotics, biomedicine and satellite observation of earth. Several applications in the field of robotic systems that can autonomously detect obstacles were developed and implemented such as a car-like wheeled robot, in which navigation is based on a distributed active-vision network-space approach using fuzzy logic[2]0. The scheme also includes trajectory tracking and obstacle

Corresponding Author: Salaheddin Odeh, Department of Computer Engineering, Faculty of Engineering, Al-Quds University, P.O. Box 20002, Abu Dies, Jerusalem, Palestine

avoidance, where two distributed wireless charge-coupled-device cameras individually driven by two stepping motors are constructed to capture the dynamic pose of the robotic system and the obstacle. In another investigation, the MARS project[3], a rover for exploring the Mars needs to be able to navigate autonomously. NASA particularly had been looking towards designing a rover to explore Mars and other celestial bodies before men and women are sent to the planet. One reason the rover must be intelligent is that light takes around 28 min to travel to Mars from the Earth and therefore real-time tele-operation[4] is out of question. The Jet Propulsion Laboratory (JPL) team at NASA has considered the robot vision approach to be a good solution for detecting obstacles because it is non-mechanical, non-scanning and compatible with stereographic viewing by human operators. Baker *et al*.[5] presented a vision-based tracking system of an outdoor mobile robot for the purpose of street-crossing. The system can detect and track vehicles in real time to extract motion regions to decide safe crossing timing. Some other robotic applications, whose main concern is dealing with stereo vision-based obstacle detection[6], are focused on partially sighted people. Obstacle avoidance is a major requirement for any technological aid aimed at helping such people, even for blinds or children to navigate safely. A further application, which serves impaired people, is a vision-based assistive navigation for robotic wheelchair platforms[7]. It presents an interesting example, where conventional wheelchair platforms are expanded with advanced navigational capabilities.

This research project has been aimed at developing a system that not only detects obstacles in a mobile robot's path using a single camera as the only sensory input, but it also achieves the target point in an optimized manner. For this reason, an algorithm, which takes the total path length and safety into account, was developed. The implemented vision-based autonomous mobile robotic system can be applied in an industrial environment to move safely from one point to another by detecting and avoiding the stationary or movable obstacles existing in its path, which stands for a line production in the site and so on.

MATERIALS AND METHODS

Robotic system control overview: As illustrated in Fig. 1, the obstacle detection is carried out by software, which analyzes and processes scene images captured by a (vision) camera installed on the ceiling inside the site, where the robot and the obstacles are located. The following algorithm is the main procedure of the control software of the robotic system in forms of pseudo-code.

Initialization:

```
WHILE robot system is running
  BEGIN
    Image capturing // Result: Scene shot
    Obstacle detection // Result: Obstacle perception
    Obstacle modeling // Result: Obstacle model
    Obstacle avoidance // Result: Motion command
  END
```

After building a computer model of the site with its different objects (the robot and the obstacles), the system starts planning the routing path that reaches from the current point to other possible end points. The methods and techniques used for building the model are image processing and computer vision. The former involves manipulating a digital image to generate a second image that differs in some respects from the original one; whereas the latter involves extracting numerical or symbolic information from images[0]. One reason our system can be seen as an autonomous and, thus, as artificially intelligent is because of its capability to decide the movement path to be pursued without interfering of human pilots or operators. The robot tries to locate obstacles, both stationary and moveable, plans how to bypass these objects and finally acts according to the resulted plan. According to the hindering obstacles,

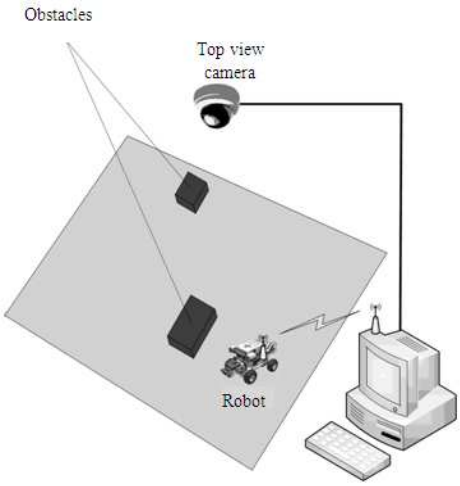

Fig. 1: Principal architecture of the vision-based autonomous mobile robotic system

the movement path can be either invariant or variant. This forces us to compute and update the route sequentially through rechecking the entire environment or site state using a sequence of scene shots.

In this investigation, the consideration and solutions for realizing the vision-based autonomous mobile robotic system leans on the following restrictions and assumptions: Firstly, the robot moves in the vertical (default) or the horizontal direction. Secondly, the obstacles might be stationary or movable with a fixed or an accelerating speed. Thirdly, the obstacles move in the horizontal direction to the left and right and vice versa.

Image capturing: Capturing images of the robot in its environment is achieved by a digital colored video camera as a visual input device, which provides the system software with the necessary visual sensory data for further analysis. The passive nature of the camera implies that it has low signal interference in the presence of other sensors, so it gives it an advantage over other types of sensors that might be used to obtain environmental information. A top view camera such as the DOME camera, which delivers a view of 360° (near bird view) is the most suitable for such kind of applications. It can be connected to the computing system via USB port. An image resolution of 680*480 pixels with 24 bit RGB color format is sufficient for such kind of application to obtain the necessary information. The computing system receives a video stream of the site and with the help of a software timer, only single shots from that stream are captured and then analyzed. Among the different software packages that are suitable to deal with digital video cameras is the Microsoft DirectX Software Development Kit, especially the dynamic library link "DShowNet.dll".

Noise reduction: Before going futher, it is of great significance if we redurce or, even, remove the noise contained in an image. However, noise removing is related with information loss. Most noise removal processes are called filters that are applied to each point in an image, the so-called convolution, by using information in a small local window of pixels (the mask) and then replacing the value of the center pixel with the value determined[9]. One appropriate noise removal is the median filter[10], which, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel. However, the original value of the pixel is included in the computation of the median. Median filters are quite popular because (for certain types of random noise) they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Order-statistics filters, to which the median filter belongs, are nonlinear spatial filters whose response is based on ordering or ranking the pixels contained in the image area encompassed by the filter and then replacing the value of the center pixel with the value determined by the ranking result.

Obstacle and robot detection: Our approach depends on the instantaneous global perception of obstacles and robot positions. Distinguishing robot pixels from background and obstacle pixels is done by using distinguished colors for the background (green), the robot (red) and the obstacles (blue or others). Figure 2 shows two output images, one for the obstacles and another for the robot, are the results of applying an extracting filter such as the Euclidean color filtering on the original image. The Euclidean color filtering extracts the pixels that are closed to the desired object color (the robot or the obstacles) by using the Euclidean distance described in formula 1), through which we can obtain how much the color vector C_1 of each pixel in the image closes to the desired color C_2. In a color image, each pixel consists of the three color components: Red, green and blue modeled as integer values between 0 and 255. While 255 means full color tense, 0 means no color at all:

$$d = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2} \quad (1)$$

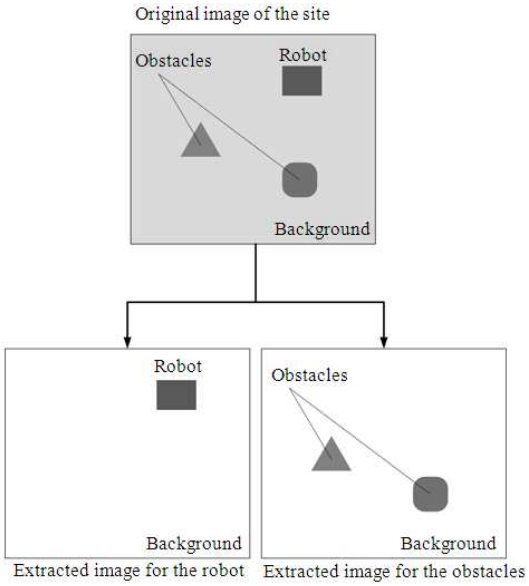

Fig. 2: Manipulating and processing of image data through the Euclidean color filter for obtaining the robot and obstacles images

It is clearly noted that if d is less than the predefined threshold of ε then the pixel keeps its original color, otherwise the pixel is set to black [0, 0, 0]. Up till this phase, two images are produced from the original image: One includes the robot and the other the obstacle, through applying the image processing techniques.

Obstacle modeling for computer vision: This phase looks into the meaning of the data that was being provided by the previous phases. The acquired data are used as input to the model and algorithm developed to deal with the obstacles and the robot positions, which come under the computer vision part of this investigation, thereby achieving the necessary representative or symbolic data through obstacle modeling based on the images previously acquired by the image processing phase. The symbolic data are needed later for the autonomous obstacle avoidance.

Converting to rectangle frames: In order to obtain a representative model of the obstacles and the robot, a converting algorithm, the so-called image-symbol conversion algorithm, determines the dimension of the obstacles and the robot available in forms of images, which has been acquired in the previous step. One way to determine these image objects is to scan the image after clustered pixels and to delimit the founded clusters with rectangle frames, each with (x, y) coordinates, height and width. Consequently, every object in the scene is provided with the appropriate loci data in the spatial domain of the site. No matter what shape the obstacles and the robot have, they are mapped into rectangle frames for the purpose of simplifying the manipulation of these objects. As Fig. 3 shows, software-technically, both images for the robot and the obstacles that are previously acquired through applying the Euclidean color filtering, will be scanned and analyzed to allocate the robot and the obstacles position within the site image. The scanning algorithm scans the image containing the obstacles and only if a fixed number of adjacent pixels of an obstacle are met, the algorithm considers these pixels as a new obstacle object.

The information achieved by the conversion process is stored in an array data structure, simplifying the symbolic representation of the robot and obstacles data. Accordingly, it is less complicated to control the robotic system within the site embracing the dynamical obstacles. Every data structure record consists of various data elements that are necessary for representing and managing our robot path problematic.

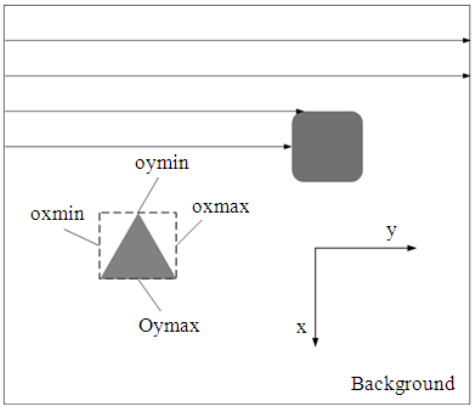

Fig. 3: Scanning an image for obstacle detection for the purpose of symbolical representation

The image-symbol conversion algorithm functions as in the following:

- Once the algorithm meets a new obstacle object, it reserves a new obstacle record and then sets the obstacle record element “oymin” (Fig. 3) to the current y-coordinate value. This value is fixed and could not be changed for the currently scanned obstacle
- Conversely, it is possible to change the variable “oxmin” since the algorithm might meet pixels with “x” values less than the current stored one
- The values of the variables “Oymax” and “Oxmax” will be incremented with each new line scan that contains pixels, which belong to a specific obstacle

As explained above, all obstacles in the image site, which can be in forms of any geometrical shape, will be mapped into representative rectangles.

Converting to gridded squares (2D-boundaries): From a computational viewpoint, it is more efficient if we quantize the image site with its different objects. As is obvious, the interplay of the robot and obstacles takes place in the spatial domain and thus, we have here to quantize the two dimensional plane, where the robot and the obstacle being. Quantization consists of selecting breakpoints in magnitude and then re-mapping any value within an interval to one of the representative output levels[0]. In order to convert the captured images of the site consisting of the robot, obstacles and background into symbolic data, the view port will be divided into gridded squares (hereafter referred to as grid elements) to enable the software system to deal

with gridded squares and not with single pixels. The grid-element size has a role key regarding the determination of the free background space and the path options to be planned. On the one hand, keeping the grid elements small gives more adequate results and more free grid-element options but this will increase the processing time, which may affect the characteristics of our real-time system; on the other hand, keeping the grid element size large reduces the free space utilization that may yields to less options when deciding the robot safe path. Pursuing of grid information is much easier regarding problem solving and demands less computational effort due to the fact that a single pixel has no significant meaning without its surrounding.

Every grid element represents a group of image pixels ordered into equally sized boundaries and has its own coordinates G (i, j) that is obtained by reading the image pixels from the image origin p (0, 0) until reaching the coordinate p (width-1, hight-1) from top left towards bottom right side. As a result, the representative data are achieved through coding the pixels into their respective 2D-boundary indices and, then, deciding the value of that grid element using the previously defined boundaries based on the pixel index, leading to fill the grid elements with the appropriate values for that boundary. If the grid element index is located within the robot boundary, then the grid element will be assigned the symbolic value R for robot. If it is located within obstacle boundaries, then the grid element value is set to the symbolic value O; while the remaining grid elements (background) is assigned a value equal to B.

The LEGO robot: Our research aimed at developing a vision-based algorithm to move a robot hindered by stationary or movable obstacles securely, A suitable robotic system for experimenting and testing our techniques is the Lego Mindstorms NXT[12], which can be remotely controlled through a wireless communication and whose controlling code easily embedded in the vision-based algorithm code. The MINDSTORMS robotic system is flexible to build since it includes various components such as an array of building blocks, motors, sensors and a central controller, known as the LEGO RCX[13]. The LEGO Mindstorms makes use of the LEGO assembly instruction set (LASM) to interpret and execute both onboard programs and PC-driven requests. LASM byte code is sent to the RCX from a PC via infrared data transfer, usually from a tower connected to the PC via USB or RS-232 (serial) cable. Two motors built in the robot are used to provide all possible motions including forward, reverse, right and left turns.

Autonomous obstacle avoidance in the quantized spatial plane: The information obtained previously can be now used to construct the robot free path. During this phase, the control program determines a motion command to control the robot with each timer tick. Up till now, the image is divided into a number of grid elements that each has its representative value and coordinates. Obstacle grid elements, the robot and the free-space grid elements are known. What we are going to use is the available information in order to determine the path and then to send the robot a motion command. For controlling the robot movement inside the environment, several parameters about each obstacle object must have been managed and stored sequentially by the control software. A summary of these parameters is discussed in the following:

- Side: This parameter is used in combination with the motion direction to decide whether it is necessary to worry about the obstacle crossing or not. The obstacle position can have the values LEFT, RIGHT and CENTER, corresponding to the current location of the robot. It is to note that the x values for the right and left side of both the robot and the obstacles are significant for taking that decision. This parameter is implemented inside a method concerning about determining the position of an obstacle relative to the robot
- Closeness: This parameter gives the number of grid elements that the obstacles are far or in contact vertically with the right or the left edge of the robot. It is used to calculate whether it is safe to cross or to go around an obstacle
- Robot motion: It can have two values, vertical and horizontal; however, the robot default motion is in the vertical direction, as long as no obstacles will be crossing its path
- Grid element: This essential parameter is used to model the environment of the vision-based autonomous mobile robotic system, where the interplay of the different objects, namely the robot and obstacles, takes place. Every grid element is definite through its discrete coordinates i and j and indicates whether it is free or occupied by a part of the robot or an obstacle. For every grid element, the controlling software stores the current and previous information, $G_c(i,j)$ and $G_p(i,j)$, with the following meanings:

$$G(i,j) = \begin{cases} B & \text{Free space background} \\ O & \text{Occupied by an obstacle} \\ R & \text{Occupied by the robot} \end{cases}$$

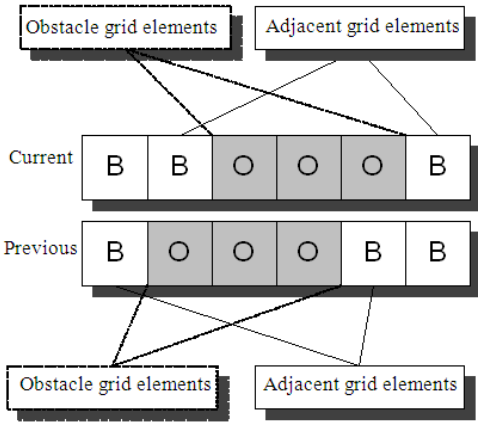

Fig. 1: Determining the obstacle motion direction through the current and previous grid elements of an obstacle and its adjacent grid elements

It is interesting to note that the previous grid elements of an obstacle, in addition to its adjacent grid elements, play a central role in finding out the motion direction of an obstacle. Technically, such software approach is a powerful approach due to the fact that it is the only history information processed or saved through two successive iterations (Fig. 4).

The adjacent grid elements are known from the current obstacle grid coordinate values, so when we know where the obstacle starts and ends, we can also know its adjacent grid elements from left and right. Since we know the adjacent coordinates, we are able to find out from the history information what the pervious value was, which is already saved in the previous iteration. If it is the first iteration step, then the previous grid values are set to zeros. From Fig. 4, the current and the previous values are used to determine the direction of motion of an obstacle, which can be to the right, to the left, or even if it is still. It can be noticed that the current value for both the right and the left adjacent grid elements are zeros.

The following pseudo-code clarifies how we can determine the motion state of an obstacle:

```
BEGIN
IF the previous value of the right adjacent grid element equals B AND the previous value of the left adjacent grid equals B
    BEGIN
        The obstacle is still
    END
ELSE IF the previous value of the right adjacent grid equals B AND the previous value of the left adjacent grid equals O
    BEGIN
        The obstacle motion is to the right
    END
ELSE IF the previous value of the right adjacent grid equals O AND the previous value of the left adjacent grid equals B
    BEGIN
        The obstacle motion is to the left
    END
END
```

The following pseudo-code includes the complete algorithm for controlling the motion and direction of the robot:

```
WHILE robot does not reach the end line
  BEGIN
    IF the robot motion is vertical
      BEGIN
        IF the robot coordinate y-value is far from the nearest obstacle
          BEGIN
            Move the robot forward until the robot is near to the obstacle
          END
        ELSE
          BEGIN
            Determine the parameters SIDE, CLOSENESS and ROBOT MOTION for the nearest obstacle to decide the robot motion command.
          END
      END
ELSE IF the robot motion is horizontal towards the left side
    BEGIN
        IF the robot is bypassing an obstacle from the left
          BEGIN
            Motion decision is to turn the robot to the right in order to return the robot to the vertical direction.
          END
        ELSE // the robot is still in contact with the
             // obstacle
          BEGIN
            Move the robot forward in the horizontal left direction.
          END
        END
ELSE IF the robot motion is horizontal towards the right side
```

```
        BEGIN
          IF the robot is bypassing an obstacle from the
          right
            BEGIN
              Motion decision is to turn the robot to the
              left in order to return the robot to the
              vertical direction.
            END
        ELSE // the robot is still in contact with the
        obstacle
            BEGIN
              Move the robot forward in the horizontal
              right direction
          END
        END
END LOOP
// The robot reaches the end line
```

RESULTS

Figure 5 clarifies a scenario, in which the autonomous robotic system based on the previous and current obstacle information its next motion course concludes, thereby avoiding to collide with the obstacles. In this example, the robot moves in the vertical direction. Within this state, the parameters for the most adjacent obstacle (Fig. 5) that are necessary for controlling the robot movement inside the environment have these values: "side = obstacle to the right of the robot", "closeness = contact" and "obstacle motion = to the right". Accordingly, the motion decision for the robot is "Turn to the left to bypass the obstacle".

According to Kamon and Rivlin[0], total path length and path safety are the two evaluation criteria that can be used to measure the performance of algorithm for robotic path planning. The path safety consideration takes place while evaluating path quality. By default, the algorithm for deciding the motion direction of the robot guides the robot in the vertical path as long as there are no obstacles crossing its path. Accordingly, this leads to a minimized path length. Otherwise, it controls the robot to move in the horizontal path if an obstacle crosses its track. In other words, the robot doesn't wait for that particular obstacle to clear the vertical path by going around it until it will become clear. Moreover, a reverse motion is applied before the robot needs to turn right or left in case of lack of enough space for achieving that, so making the motion safer. Besides using the grid approach for obtaining a less computational effort, it provides a safe margin that prevents contact between the robot and the obstacles.

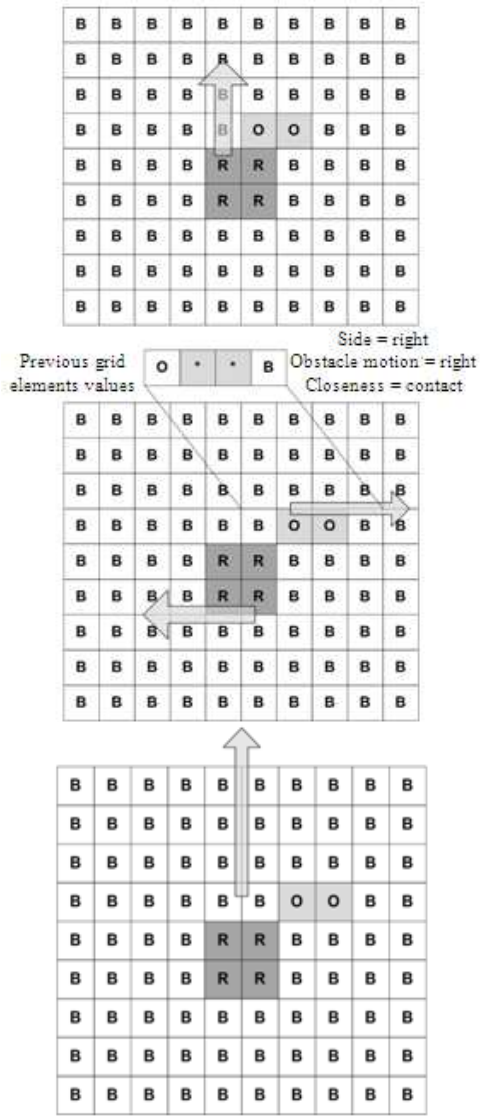

Fig. 5: A scenario within the site, in which the autonomous robotic system based on the previous and current obstacle information its next motion course, concludes, thereby, avoiding to collide with the obstacles

There are several difficulties encountered while carrying out this research:

- Controlling the movement of the robot: In order to rotate the robot 90° to the right or to the left, the Lego robot must use a time parameter. This is achieved through motion turning by powering one motor and turning off the other for a predefined time. Consequently, it is difficlut to have exactly

90° of turn besides, this procedure is aggravated through the poor motor performance, which depends on the battery level. The robot turning requires space so the motion command must be initiated in advance, which makes the programming harder as the turning space must be included in any situation that faces the robot

- Sizing the Scene: The used camera is a web cam, which provides a very narrow view, resulting in reducing the available scene size. To maximize the scene size either the camera must be heightened or a larger degree view camera must be used such as the DOME camera
- Calibrating the camera: The light tense affects the color levels of the scene objects that influneces the color segmentation negatively. Thus, this results in a variation of the value of the grids

CONCLUSION

It has been shown how we developed the autonomous mobile vision-based robotic system, which demands a less computational effort through quantizing the image site consisting of the robot, obstacles and the background, leading to a powerful computing approach as we only save the state information of the current and previous steps. The robotic system is able to direct the robot between two points safely, by acquiring the spatial states using a top view camera as the only sensor used; then applying image processing followed by computer vision in an artificial manner. Introducing additional parameters, the robot motion can take place in a more complicated environment, in which the obstacles movement is not only restricted to a horizontal movement, but rather in every direction. One might approach such solutions and bypass the previously supposed limitations by applying more advanced computer vision techniques such as model-based tracking using Kalman filter[0] for pursuing the obstacles with unrestricted motion.

REFERENCES

1. Picardi, M. and T. Jan, 2003. Recent advances in computer vision. Ind. Phys., 9: 18-21. http://radio.weblogs.com/0105910/2003/03/22.html
2. Hwang, C.L. and C.Y. Shih, 2009. A distributed active-vision network-space approach for the navigation of a car-like wheeled robot. IEEE. Trans. Ind. Elect., 56: 846-855. http://ieeexplore.ieee.org/xpl/preabsprintf.jsp?arnumber=4608745
3. Katz, G., 1993. The MARS project: Robot navigation using a vision system. MA thesis, School of Electrical Engineering and Computer Science and Engineering. The University of New South Wales. http://www.geocities.com/SiliconValley/Lakes/7156/t2.htm
4. Baard, S., 1994. Teleoperation and Robotics in Space. The American Institute of Aeronautics and Astronautics, ISBN: 1563470950, pp: 502.
5. Baker, M. and H.A. Yanco, 2005. Automated street crossing for assistive robots. Proceeding of the IEEE 9th International Conference on Rehabilitation Robotics, June 28-July 1, IEEE Xplore Press, Chicago, Illinois, pp: 187-192. DOI: 10.1109/ICORR.2005.1501081
6. Se, S. and M. Brady, 1997. Stereo vision-based obstacle detection for partially sighted people. Proceeding of the Asian Conference on Computer Vision N°3, Jan. 8-10, Hong Kong, pp: 144-151. http://cat.inist.fr/?aModele=afficheN&cpsidt=2038765
7. Tranhanias, P.E., M.I.A. Lourakis, A.A. Argyros and S.C. Orphanoudakis, 1996. Vision-based assistive navigation for robotic wheelchair platforms. Proceedings of the IAPR TC-8 Workshop on Machine Perception Applications, (MVA'96), Graz, Austria, pp: 43-57. http://portal.acm.org/citation.cfm?id=251041
8. Kubota, H., K. Fukui, M. Ishikawa, H. Mizoguchi and Y. Kuno, 1990. Advanced vision processor with an overall image processing unit and multiple local image processing modules. Proceedings of the IAPR Workshop on Machine Vision Applications, Nov. 28-30, Tokyo, pp: 401-404. http://www.cvl.iis.u-tokyo.ac.jp/mva/proceedings/CommemorativeDVD/1990/papers/1990401.pdf
9. Gonzalez, R.C., R.E. Woods and S.L. Eddins, 2004. Digital Image Processing Using MATLAB. Prentice Hall, ISBN: 0130085197, pp: 609.
10. Gonzalez, R.C. and R.E. Woods, 2002. Digital Image Processing. Prentice Hall, pp: 793. http://books.google.com.pk/books?id=iW0dAQAACAAJ&dq=Digital+Image+Processing.
11. Li, Z.N. and M.S. Drew, 2004. Fundamentals of Multimedia. Prentice Hall, ISBN: 0130618721, pp: 560.
12. Astolfo, D., M. Ferrari and G. Ferrari, 2007. Building Robots with LEGO Mindstorms NXT. Syngress, ISBN: 1597491527, pp: 447.

13. Bishop, O., 2008. Programming Lego Mindstorms NXT. Syngress, ISBN: 1597492787, pp: 187.
14. Kamon, I. and E. Rivlin, 1997. Sensory-based motion planning with global proofs. IEEE Trans. Rob. Autom, 13: 814-812. DOI: 10.1109/70.650160
15. Morris, T., 2003. Computer Vision and Image Processing. Palgrave, Macmillan, ISBN: 0333994515, pp: 300.