A Tracked Mobile Robot with Vision-based Obstacle Avoidance

K. Varun Raj*, Sharschchandra V. Bidargaddi, K.N. Krishnanand, D. Ghose
Mobile Robotics Laboratory
Department of Aerospace Engineering
Indian Institute of Science, Bangalore-560012

* Corresponding author (email: varunrajk@gmail.com)

Abstract

This paper presents VITAR (VIsion based Tracked Autonomous Robot), a robotic test bed that consists of a tracked mobile robot equipped with a pan-tilt mounted vision system, an onboard PC, driver electronics, and a wireless link to a remote PC. A novel vision-based obstacle avoidance algorithm is implemented on the robot. The robot uses histograms of images obtained from a monocular and monochrome camera to detect and avoid obstacles in a dynamic environment. As the generation of histogram is computationally inexpensive, the robot is quick to avoid the obstacles. Several experiments conducted with different obstacle environments validate the effectiveness of the algorithm.

Keywords: Tracked mobile robot, vision-based obstacle avoidance, image segmentation.

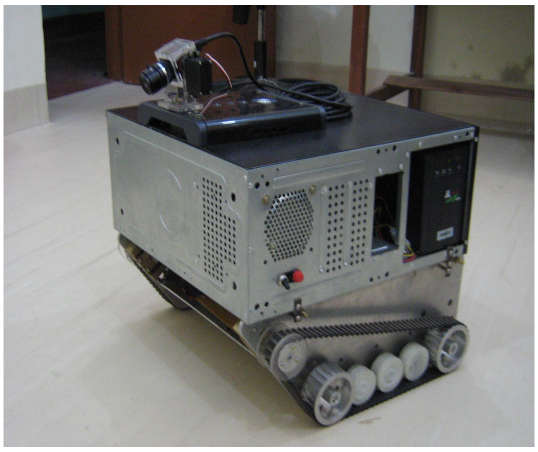


Figure 1: VITAR (VIsion based Tracked Autonomous Robot)

1 Introduction

VITAR (**VI**sion based **T**racked **A**utonomous **R**obot) is a mobile robotic system built for indoor/outdoor navigation research. VITAR uses tracks for locomotion which makes it a versatile platform to operate over diverse terrains as tracks provide a greater surface area of contact with ground than that of the wheels [1]. Wheeled vehicles demonstrate excellent mobility and speed on road. But when off-road usage is required, and where wet conditions prevail, mobility is suffered. Tracked con gurations provide signi cantly better travel times when the operation requires off-road usage, and in some cases they guarantee the best mobility for all-weather tactical operations [2]. However, it takes considerable power to steer a tracked vehicle as the leading and trailing ends of the footprint skid sideways, perpendicular to the direction in which the tracks roll [3].

One of the crucial capabilities of a mobile robot is to detect and avoid collisions with obstacles present on the robot's path. Till date, several sensor modalities like ultrasonic, infrared, laser, and vision have been used to aid mobile robots in obstacle avoidance. Vision based obstacle avoidance offers several advantages over other range based sensors: detecting at obstacles and holes, differentiating between road and the adjacent at grassy areas, and enabling navigation in rocky terrain. While vision based perception enables mobile robots to handle complex obstacle environments as mentioned above, it is faced with several major challenges. Despite over 30 years of research, there is no agreement within the computer vision community on the best approach to achieve this task (see [4] for a review of progress to date). Moreover, a precise de nition of obstacle detection is lacking [5]. Different methods have their strengths and weaknesses and no one method is universally better than the alternatives.

This paper presents a novel vision-based obstacle avoidance algorithm for mobile robots. Obstacle detection systems can be broadly classi ed into range-based and appearance-based obstacle detection systems [6]. The obstacle avoidance algorithm discussed in this paper falls under the second category.

The paper is organized as follows. In Section 2, details of the robot hardware, software architecture, and the vision library are presented. The novel vision based algorithm used for obstacle avoidance is explained in Section 3. Algorithmic implementation is discussed in Section 4. Results of real-robot-experiments with various obstacle environments are presented in Section 5. The paper concludes with some remarks and directions for future work.

2 Vehicle Description

The mechanical sturcture of VITAR (Figure 1) is constructed around an aluminium chassis. Double-sided timing belts are used as tracks for the robot's locomotion. A tensioner system is tted to the side plates of the chassis in order to adjust the tension of the tracks. The vehicle can carry sensors and a payload which allows users to con gure it with various hardware modules depending on the application requirements. The mechanical speci cations of the platform are shown in Table 1 .

Table 1: Platform mechanical speci cations.

Dimension	*Length*	420*mm*
	Width	450*mm*
	Height	350*mm*

2.1 Onboard Computing

The robot is tted with an onboard PC which hosts the control software. The system runs on a Pentium IV 3.6 GHz processor and is equipped with an 80-GB hard drive, FireWire, and WiFi support. The system features the RedHat Fedora operating system and runs the Linux Kernel version 2.6. The onboard computer, which runs high-level control tasks, interfaces with the vehicle through Parallel, Serial, and USB ports. The high-level tasks are serviced by a middle-level microcontroller unit (MircroChip Pic 16F877a) which runs closed loop programs to control the motor drives and other sensors.

2.2 Power System

The vehicle electronics, including the onboard computer and the drive motors, are powered by an uninterrupted power supply (UPS) running on a 12V dry battery. On a full charge, the battery can keep the robot running for approximately 30 minutes. An AC/DC converter is used to provide 9V input to the drive motors. As the UPS comes with a surge protection socket, no additional battery is required to protect the sensitive electronics from the surges generated by the motors.

2.3 Software Architecture

In this section, we describe the higher-level software used for VITAR. Figure 2 shows the block diagram of the vehicle software architecture.

A wireless interface (WiFi) is used to set up communication between the robot and a remote PC. The images/data taken by the camera are transmitted to the remote PC. The robot control software and the computer vision algorithms are coded in C language. We used OPENCV library as a platform to develop vision algorithms. To acquire good quality images at high speed, we use a IEEE 1394 FireWire camera (GUPPY F033b by Allied Vision Technologies). The images grabbed by the camera are processed and the appropriate control signal are generated in order to achieve obstacle avoidance maneuvers.

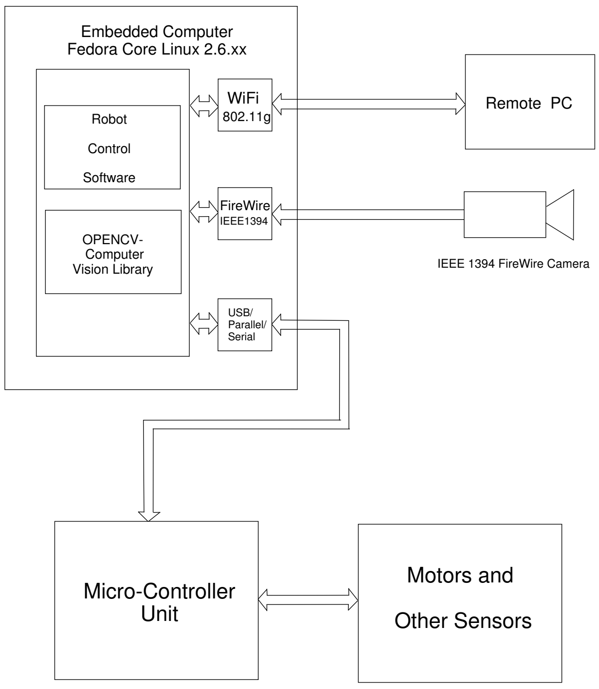


Figure 2: Vehicle Software Architecture

3 Obstacle Avoidance Algorithm

Our obstacle avoidance algorithm uses the principle of appearance-based obstacle detection. According to this principle, any pixel that differs in appearance from the ground is classi ed as an obstacle. This method is based on three assumptions that are reasonable for a variety of indoor and outdoor environments:

1. Obstacles differ in appearance from the ground.
2. The ground is relatively at.
3. There are no overhanging obstacles.

The rst assumption allows us to distinguish obstacles from the ground, while the second and third assumptions allow us to estimate the distances between detected obstacles and the camera [6]. However, our algorithm uses only the rst assumption. We don't make use of any distance measurements between the detected obstacles and the camera. This makes the robot detect obstacles even on an uneven terrain. It is just the mere appearance of the obstacle in front of the camera which is being considered in the decision making process. The algorithm uses histograms to achieve obstacle detection. As the generation of histograms is computationally inexpensive, the algorithm is fast and the robot can detect the obstacle in real time. This makes the robot to perform a quick maneuver when an obstacle pops in front of it.

However we make an assumption that there is no obstacle right in front of the robot.This assumption is required during the initial phase when the robot begins to move and later it can be relaxed.

Images are captured by the camera at 30 fps (frames per second) and are input to the algorithm. The basic approach of the algorithm is explained using the following example. Figure 3 shows an input image. For each image that is grabbed by the camera, a normalized histogram *H1* of that frame is generated using a mask *M1*. This mask is mainly used to set the robot's horizon by restricting its eld of view. The width of the mask determines the minimum distance from the obstacle before the robot starts to perform the avoidance maneuver.

The histogram of a digital image with ***L*** total possible intensity levels in the range **[0,G]** is de ned as the discrete function:

$$H(r_k) = n_k \tag{1}$$

where r_k is the k^{th} intensity level in the interval **[0,G]** and n_k is the number of pixels in the image whose intensity level is r_k.

Mask M1 determines what pixels of the source image are used for counting. The mask is so selected which includes only the bottom side pixels in the image that are towards the camera side as we are not interested about the obstacles that are far away from the robot. Figure 4 illustrates mask M1 used in the algorithm. This mask is of the same size of the image. The pixels having 255 intensity value (i.e. white pixels) in the mask M1 are the only pixels in the source image that are used for generating a histogram. The histogram generated of the example image is shown in Figure 5. Mask M1 forms an important parameter which determines the minimum distance before taking an evasive turn avoiding obstacle. This mask M1 is a function of the robot size. This enables the algorithm adjust to different sizes of the robots.

Next, the peak of the histogram is located and its corresponding intensity value is found. The peak corresponds to largest region occupied by a single intensity value. This could be of the obstacle or the oor . The colors of all the pixels satisfying the following condition (3) are changed to white and the rest to black. Let $f(x,y)$ denote the intensity value at a pixel (x,y). Let *P* denote the peak intensity value of the histogram.

$$|f(x,y) - P| \leq \delta \tag{2}$$

where δ is a threshold parameter.

We assumed a convenient δ value to be 40. Thus, a new binary image *BI* is generated showing the path and the obstacles. There may be multiple local peaks in the image denoting the oor and the obstacles. By doing the operation that is mentioned above, we segment the image into two regions, where one region denotes the obstacles and the other region denotes the path. Note that painting white does not indicate that the corresponding pixel is of the oor nor painting black indicates that the pixel is of the obstacle. Figure 6 shows the binary image generated for the example image.

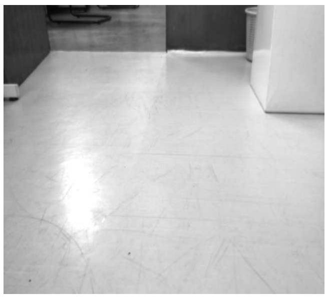


Figure 3: Sample Image

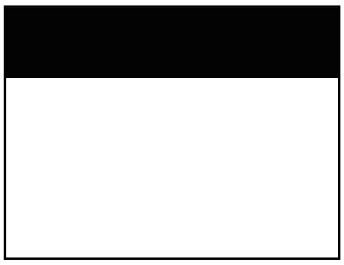


Figure 4: Mask M1

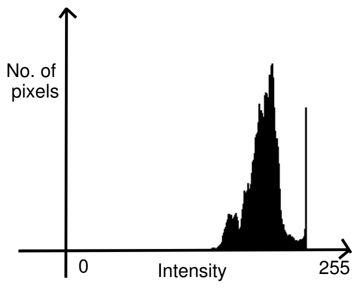


Figure 5: Histogram generated using mask M1

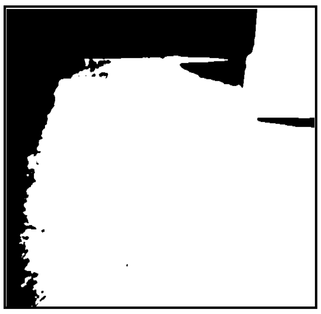


Figure 6: Binary image after thresholding

We introduce another mask M2, which helps the robot to successfully avoid the obstacles. This mask is applied on the binary image BI obtained using (3) and a normalized histogram H2 of the output of the mask M2 is generated. Note that the new histogram will have only two intensity levels i.e.0 and 255. This histogram is illustrated in the Figure 7. Selection of M2 is the criteria in nding out the heading direction. The mask area of M2 is set smaller than that of M1 and it includes those pixels which are closer to the camera side. The assumption stated earlier that no obstacles are present right in front of the robot helps us in not having any obstacles in the region of mask M2. So the size of this mask is made as small as possible so as to decrease the nearest distance to detect close by obstacles. The histogram H2 generated using M2 determines whether the robot has to turn or go straight. In the histogram H2, if the maximum occurs at 255, then there are no obstacles close by and the robot is made to go straight. If the maximum occurs at 0, then there is an obstacle in front of the robot, so the robot is made to turn in order to avoid the obstacle.

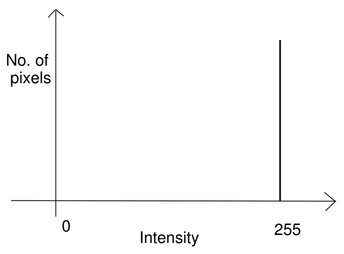


Figure 7: Histogram generated using mask M2

We now discuss a situation when the robot actually moves straight even after getting the turn signal (i.e. after the maximum shifts to 0). This could happen because of slippage in an uneven terrain or by some other error. We nd that the algorithm will still work and has time to recover until it moves a distance d which is determined by Masks $M1$ and $M2$. This distance that the robot can safely travel towards the obstacle in spite of not executing a turn command is termed as the critical zone.

4 Implementation

There are three possible locations of an obstacle with respect to robot and they are :

1. Obstacle located far away from the robot.
2. Obstacle located close to the robot but beyond the critical distance from the robot.
3. Obstacle located very close to and less than the critical distance from the robot.

In the rst case, the no. of pixels occupied by the obstacle are less than that of the pixels occupied by the oor . So, the histogram using mask M1 peaks at intensities closer to that of the oor . By using (3) all those pixels whose intensities are within ± 40 range about the peak are painted white and the rest are painted black. Now that the oor is painted white (i.e. intensity value 255), the histogram generated using mask M2 will now show maximum at intensity value 255. This is illustrated in the Figure 8. This is the indication given to the robot to move straight.

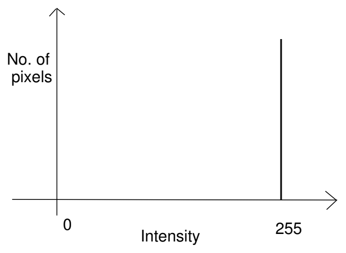


Figure 8: Histogram-I

As the robot moves closer towards the obstacle, the obstacle size in the image plane increases. Therefore, the number of pixels occupied by the obstacle in the masked region of M1 increases.So the histogram slowly changes its shape peaking at some other intensity value. Now that it has peaked at a different intensity value that corresponds to the obstacle, pixels with ± 40 intensities about the peak are painted white and rest black. So this time, the obstacle is painted white and oor is painted black. Now, the histogram generated by the mask M2 will shift its maximum from 255 intensity value to 0 intensity value. This is illustrated in the Figure 9. This is the indication given to the robot to turn and avoid the obstacle.

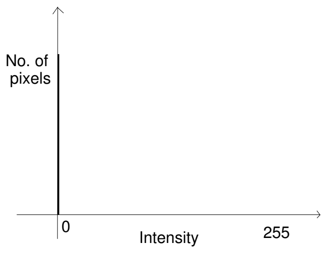


Figure 9: Histogram-II

If there is some slippage due to uneven surface or some other error due to which the robot moves forward even though the turn command was given, then the algorithm will still work as it has time to recover until it reaches the extreme end of the critical zone (i.e. the critical distance from the robot). As explained above, the pixels of the obstacles are painted white and the oor are painted black when the obstacle is close to the robot. But when it is too close, the pixels of the obstacle enter into the mask M2 region and then the histogram generated by this mask slowly shifts its maximum from intensity value 0 back to 255. This shift will occur when the obstacle pixels occupy more of this masked region than those occupied by the oor pixels. Once this shift occurs, then robot assumes that there is no obstacle present (as like in the rst case), moves forward, and makes a collision.

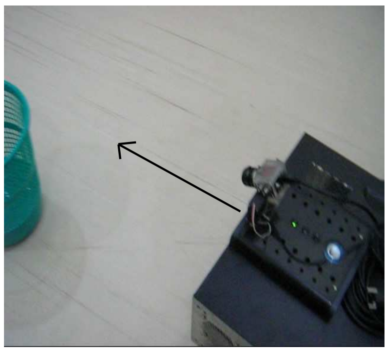
Let us consider an obstacle whose size is relatively small and even after coming close to the robot, does not occupy more pixels than that of the oor . Then the switching of intensities will not occur and the robot will collide with the obstacle. We can overcome this problem to some extent using a small variation in the algorithm. The mask M1 is divided into three columns forming three new masks. The whole algorithm is executed taking each mask into account at a time. Using each of the masks, the approximate position of the obstacle is known. Now that we are just comparing the obstacle's image size to that of the new smaller mask, the switching occurs. Once the algorithm is run for each mask, the obstacle map is generated. Obstacle map is a three bit binary number. For example, a value of 101 implies that there are obstacles in the left and right of the robot.The mask can be partitioned more to account for thin or small obstacles, but the downside of this kind of partitioning is that the obstacle detection becomes slow.

5 Experimental Results

We conduct a safe-wandering experiment using VITAR to test the ef cac y of our obstacle avoidance algorithm. Images are acquired from a AVT Guppy F033b CCD camera, which is equipped with wide-angle lens. The camera is mounted on the robot using a pan-tilt servo mechanism. The camera is connected to the computer through a FireWire link. Figures 10-16 show the snapshots taken from a video where VITAR safely wanders in an obstacle-infested environment. In Figure 10(i) we can see that the robot is moving towards the obstacle (dustbin). As the obstacle comes into its eld of view, it detects the obstacle and then avoids it by turning right (Figure 10(ii)). After avoiding the rst obstacle it moves forward till it nds another obstacle (Figure 11(iii)). In Figure 11(iv) the robot nds an obstacle on the right, and therefore avoids it by turning left. Figures 12-14 show some of the snapshots during the rest of the experiment. Figure 15 shows an image where a person is standing in front of the robot. The robot detects the person as an obstacle and therefore switching of the intensities occurs (oor with intensity 0 and the obstacles with intensity 255). This is illustrated in the Figure (16). The block diagram of the implementation is shown in Figure 17. It is a closed loop system where in the camera acts as a feedback element. These images were taken indoors at the Mobile Robotics Laboratory, Department of AeroSpace Engineering, Indian Institute of Science.

6 Further Improvements

This algorithm ensures a very less robots response time to detect and avoid an obstacle. The critical zone helps to decrease any errors in detection of the obstacle. It doesn't involve any range measurement which is very unreliable in uneven terrain.


(i)

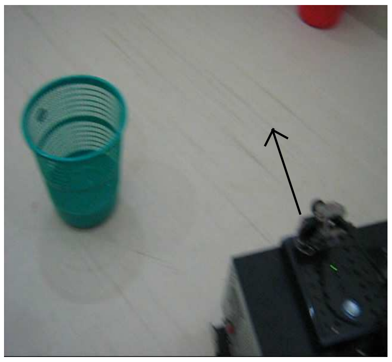
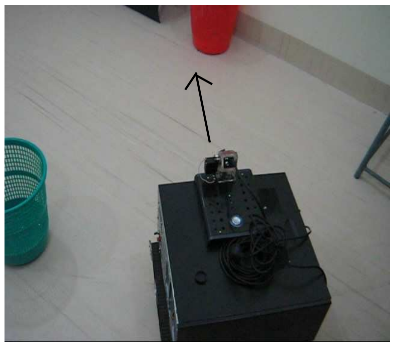

(ii)

Figure 10: Safe-wandering experiment: Snapshots i and ii


(iii)

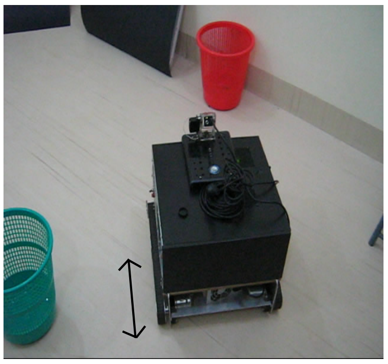
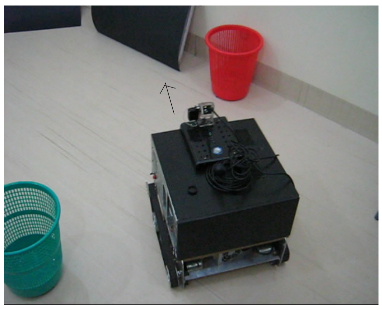
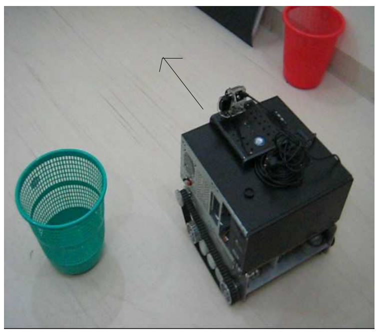

(iv)

Figure 11: Safe-wandering experiment: Snapshots iii and iv

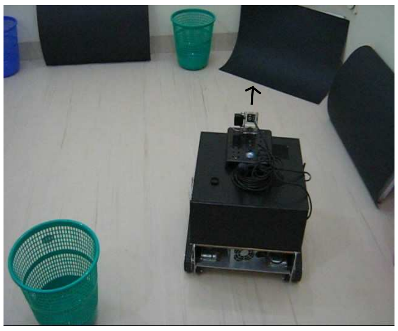


(v)

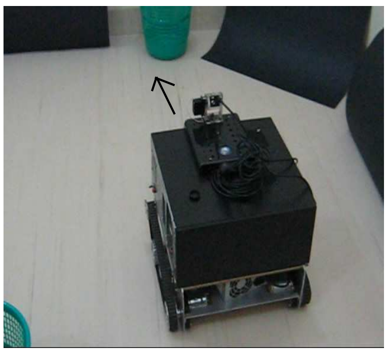


(vi)

Figure 12: Safe-wandering experiment: Snapshots v and vi

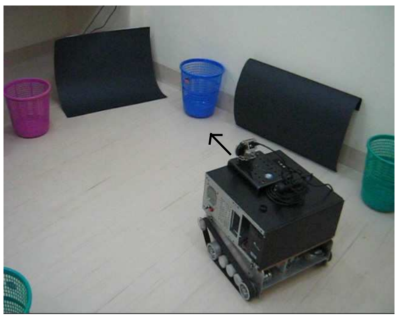


(vii)

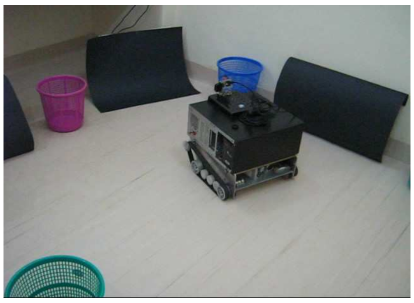


(viii)

Figure 13: Safe-wandering experiment: Snapshots vii and viii



(ix)



(x)

Figure 14: Safe-wandering experiment: Snapshots ix and x

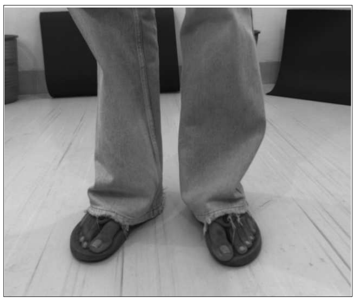


Figure 15: Image of a person standing in front of the robot

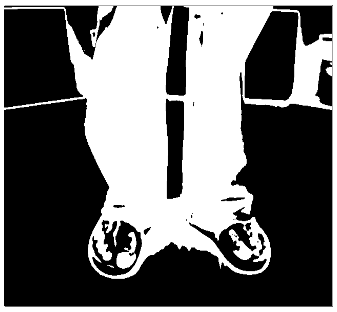


Figure 16: Binary Image generated by the algorithm

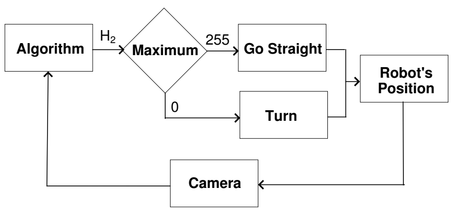


Figure 17: Block diagram of the implementation

The current algorithm can be improved in many ways. The value 40 we used in (3) can sometime be more or less. Instead, we could make δ variable which changes according to the width of the histogram peak.

Another possible improvement is to make the robot detect very small obstacles. Although a method was used in the algorithm to avoid smaller obstacles, it might not detect thin rods. However, it is unclear at this time about how to go about solving this problem without using range measurement.

7 Conclusions

This paper presented a mobile robotic test bed called VITAR (VIsion based Tracked Autonomous Robot) that was built for indoor/outdoor mobile robot navigation research. A new method for vision based obstacle avoidance using a single monochrome camera is presented. Experimental results show that the method is fast and avoids obstacles in real time the system performs well in a variety of obstacle environments.

References

[1] R F. Unger, "Mobility analysis for the TRADOC: Wheeled versus track vehicle study, Final Report," Geotechnical Laboratory, Department of the Army, Waterways, Corps of Engineers, Vicksburg, Miss., September 1988.

[2] P. Hornback, ""The wheel versus track dilemma," *ARMOR* - March-April 1998, pp. 33-34.

[3] Shraga Shoval "Stability of a Multi Tracked Robot Traveling Over Steep Slopes", *Proceedings of the 2004 IEEE International Conference on Robotics 8 Automation* New Orleans, LA April 2004.

[4] Guilherme N. DeSouza and Avinash C. Kak, "Vision for Mobile Robot Navigation: A Survey," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* VOL. 24, NO. 2, FEBRUARY 2002

[5] Z.Zhang, R.Weiss and A.R.Hanson, "Obstacle Detection Based on Qualitative and Quantitative 3D Reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 19, 15-26, 1997.

[6] Iwan Ulrich and Illah Nourbakhsh, "Appearance-Based Obstacle Detection with Monocular Color Vision," *Proceedings of the AAAI National Conference on Artificial Intelligence* , Austin, TX, July/August 2000.