Vision-Based Navigation of Mobile Robot with Obstacle Avoidance by Single Camera Vision and Ultrasonic Sensing

Akihisa Ohya

Intelligent Robot Laboratory
University of Tsukuba
Tsukuba, Ibaraki 305, JAPAN
ohya@is.tsukuba.ac.jp
http://www.roboken.esys.tsukuba.ac.jp/

Akio Kosaka and Avi Kak

Robot Vision Laboratory
Purdue University
West Lafayette, IN 47097–1285, USA
{kosaka,kak}@ecn.purdue.edu
http://rvl1.ecn.purdue.edu/

Abstract

This paper describes a vision-based navigation method in an indoor environment for an autonomous mobile robot which can avoid obstacles. In this method, the self-localization of the robot is done with a model-based vision system, and a non-stop navigation is realized by a retroactive position correction system. Stationary obstacles are avoided with single-camera vision and moving obstacles are detected with ultrasonic sensors. We will report on experiments in a hallway using the YAMABICO robot.

1 Introduction

In what has become a fairly well-researched approach to vision based navigation for mobile robots, a robot is provided with an environmental map and a path to follow[1][2][3][4]. The important function of vision-based processing in this case consists of 'self-localization.' In a different approach, a robot is provided with a sequences of images of the interior space[5][6]. By comparing these prerecorded images with the camera images taken during navigation, the robot is able to determine its location.

An important adjunct to the problem of navigation is the problem of obstacle avoidance. In the vision-based navigation work reported in the past, such as in [3], obstacle avoidance is carried out using ultrasonic sensors. These sensors take over the control of the robot as long as obstacles are detected in the vicinity. The control is then handed back to vision-based processing once the obstacles are no longer a factor. While it is expedient to use ultrasonic sensors in such a manner, we believe it is empirically more interesting to use vision-based processing for obstacle avoidance also. Since a primary focus of machine intelligence and advanced robotics is to capture human faculties in a computer and since humans use vision for obstacle avoidance while navigating, we are evidently interested in doing the same in a robot.

In this paper, we will present an integrated vision-based process for mobile robots that is capable of simultaneously navigating and avoiding stationary obstacles using monocular camera images. While the self-localization part of the process is the same as the *FINALE* system of [3], what distinguishes the work reported in this paper is that we are now able to give to the robot a vision-based obstacle avoidance capability at the same time. In the current implementation, this obstacle-avoidance capability is limited to the detection and avoidance of stationary obstacles. This is owing to the limitations of the computing hardware available to the robot. Therefore, moving obstacles must still be detected with ultrasonic sensors.

What is particularly noteworthy about our approach is that the self-localization and the obstacle avoidance are both carried out by processing the same image, thus eliminating what would otherwise be a redundancy in sensor data collection. While a model-based approach is used for self-localization, obstacles are detected by computing the difference between the edges estimated from the 3D environment model and the edges detected from the actual camera image. We should mention that is not the only approach to vision-based detection of obstacles. As reported by [7], an alternative approach consists of computing optical-flows from the images.

2 Vision-Based Navigation

2.1 Self-Localization

As mentioned earlier, the self-localization part of the overall navigation scheme is the same as the *FINALE (Fast Indoor Navigation Allowing for Locational Errors)* system of [3]. However, the manner in which self-localization is used in the current system is different from that in [3]. Self-localization in *FINALE* kicks in whenever the variances associated with the

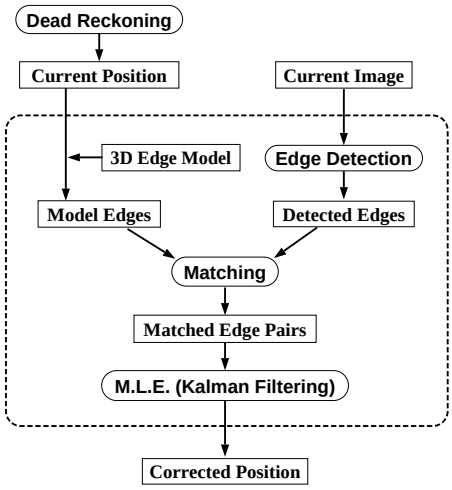

Figure 1: Flow of self-localization procedure.

positional parameters exceed a certain predetermined threshold. In our current system, self-localization is carried out on a continuous basis.

We'd like to review briefly the computations that go into self-localization. Figure 1 shows the flow of the self-localization procedure. First, the robot renders an expectation image using its current best estimate of where its present location is. Next, the model edges extracted from the expectation image are compared and matched with the edges extracted from the camera image through an extended Kalman filter. The Kalman filter automatically then yields updates values for the location and the orientation of the robot.

To illustrate the process of self-localization, Figure 2(a) shows a typical camera image. Shown in (b) is an expectation image rendered from the wire-frame model of the environment; this expectation map is overlaid on the camera image. As the reader can see, the discrepancy between the various edges in the underlying camera image and the highlighted edges in the expectation map is caused by the error between where the robot actually is and where the robot thinks it is. Shown in (c) are the edges extracted from the camera image. Note in particular that not all gray level variations in the camera image translate into edges. As explained in [3], this is due to the fact that the system only looks for those edges in the camera image that are in proximity – both spatially and in the Hough space – to the edges in the expectation map. Shown in (d) is a re-projection into the camera frame of those model edges that were successfully used for self-localization. The fact that these re-projected edges fall exactly where they should is a testimony to the accuracy of the result produced by the Kalman filter. Although not discernible, shown in (e) are two small icons, in close proximity to each other, the bright one corresponding to the updated position and orientation of the robot and the somewhat darkened corresponding to the old

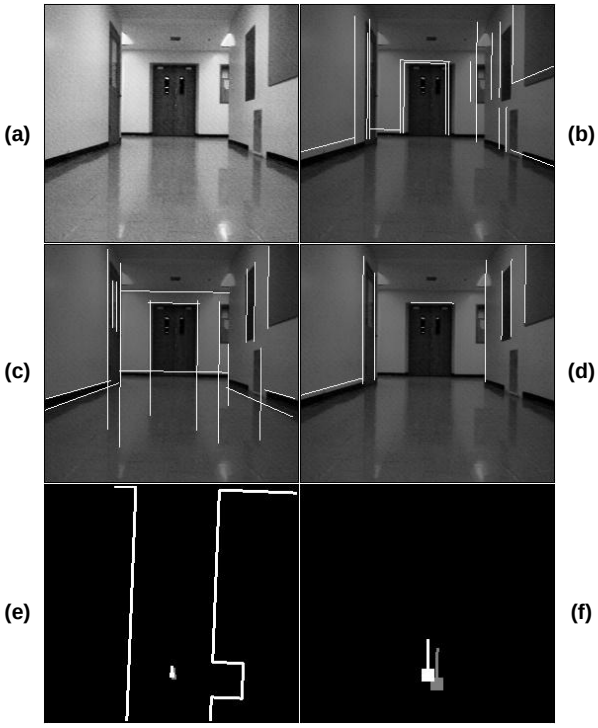

Figure 2: Sample images of self-localization. (a) Camera image. (b) Expectation map overlaid on the camera image. (c) Edges extracted from the camera image. (d) Matched model edges reprojected into the camera frame. (e) Two small icons showing the robot's old and the updated position in the hallway. (f) Enlarged version of (e). The gray icon is for the old position and the white icon is for the updated position.

position and orientation. To help the reader discern these two icons, shown in (f) is an enlarged version of the image in (e).

2.2 Non-Stop Navigation by Retroactive Position Correction

We now need to explain how the self-localization process described above fits into the overall framework for navigation. What we really want to do – and this has actually been implemented – is to navigate with dead-reckoning, meaning that we want the robot to update its position continuously on the basis of the information supplied by the wheel encoders. Unfortunately, there is always some differential slippage in the wheels that causes a purely dead-reckoning based approach to go awry if navigation is attempted over significant distances. So we want the robot to use vision-based self-localization to eliminate the errors accumulated during dead-reckoning. Due to the time delays associated with vision-based processing, this turns out to be a fortuitous combination dead-reckoning and vision-based position updating. Since we do not want the robot to come to a halt as the camera image is being processed,

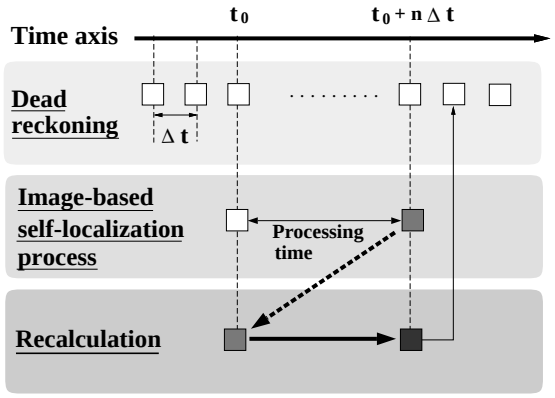

Figure 3: The timing diagram for retroactive position correction. At time t_0, an image is taken and self-localization process starts. Self-localization results become available at time $t_0 + n\Delta t$ where Δt is sampling interval for dead reckoning. The corrected current robot's position corresponding to time $t_0 + n\Delta t$ is recalculated.

what is really needed is a retroactive approach to position correction, along the lines originally proposed in [8]. As explained below, in the current system we have implemented a slight variation on the retroactive updating idea reported in [9].

Figure 3 shows the timing diagram used for retroactive position correction. At time t_0, an image is taken and the self-localization process started. The computations for self-localization come to an end at time $t_0 + n\Delta t$ where Δt is the sampling interval for dead reckoning, meaning that at the end of each Δt interval the position of the robot is recalculated based on the wheel encoder readings. Since the self-localization results correspond to the image taken at time t_0, the dead-reckoning based estimate of robot's position at the current time (the time instant $t_0 + n\Delta t$) must be recalculated. Since this recalculation is done within a sampling interval for dead reckoning, the robot is able to proceed in a non-stop manner without having to stop for processing the camera image.

3 Obstacle Avoidance

Obstacle avoidance is carried out using both vision and ultrasonic sensing. While our ultimate goal is to use only vision for obstacle avoidance, due to the limitations of the computing hardware available to the mobile robot, at this time vision can only be used for the detection of stationary obstacles. So, in the current implementation, the detection of moving obstacles is left to ultrasonic sensors.

3.1 Stationary Obstacle Avoidance

The detection of stationary obstacles is based on the premise that the camera image and the expectation map must be in near-perfect registration *immediately after self-localization.* So, any discrepancy between these two images can only be caused by the presence of obstacles in the environment, assuming of course that the edge-detection process does not result in any artifact edges in the camera image. Fortunately, as will be shown presently, the artifact edges, caused mostly by glare and other illumination dependent phenomena, can be eliminated by using adaptive thresholding.

Adaptive Thresholding. We will now explain how the adaptive thresholds are found for the edge detection operator. Recall, the edge-detection thresholds at the different locations of the robot must be such that when an edge detector is applied to a camera image immediately after self-localization, it should not yield any artifact edges. For a particular hallway with given illumination conditions, these thresholds are found through a learning procedure prior to any navigation in that hallway. The learning procedure consists of

1. clear the hallways of all obstacles
2. manually drive the robot through the different sections of the hallway
3. render expectation maps from the hallway model at regular intervals during these manual traversals by the robot
4. record the camera images at the same locations where the expectation maps are rendered
5. apply an edge detection operator to the camera images using a threshold T for the edge detection operator
6. construct a difference image between the model edge map from Step 3 and the edge map from the previous step
7. divide the difference image of Step 6 into five vertical regions and count the numbers of pixels N_1 through N_5 in each region
8. compute max N_m of these five numbers N_1 through N_5
9. go back to step 5 and by iterating Steps 5 through 8, construct a graph of the number of pixels N_m in Step 8 versus the threshold T
10. choose the threshold T_0 for which the number of difference pixels N_m is a minimum. Designate this value of N_m by N_0.

The reason why the index N_m is not just a total number of pixels in the whole difference image and is calculated in the manner mentioned above, is that it will also be used as another adaptive threshold value for determining the direction of safe passage. Figure 4 is a pictorial depiction of these steps for the determination of the optimum threshold T_0 at each location of the robot.

To appreciate why there would be a threshold T_0 that would minimize the number of difference pixels, note that for high values of T both the real hallway

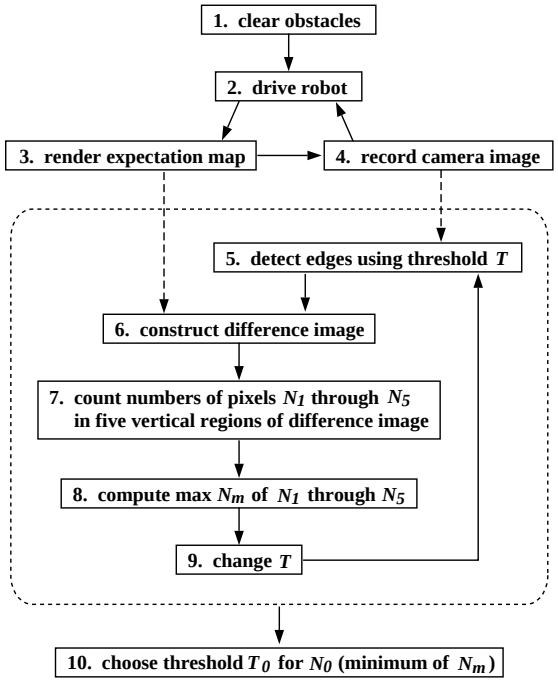

Figure 4: Flow of the learning process for determining optimum edge detection thresholds.

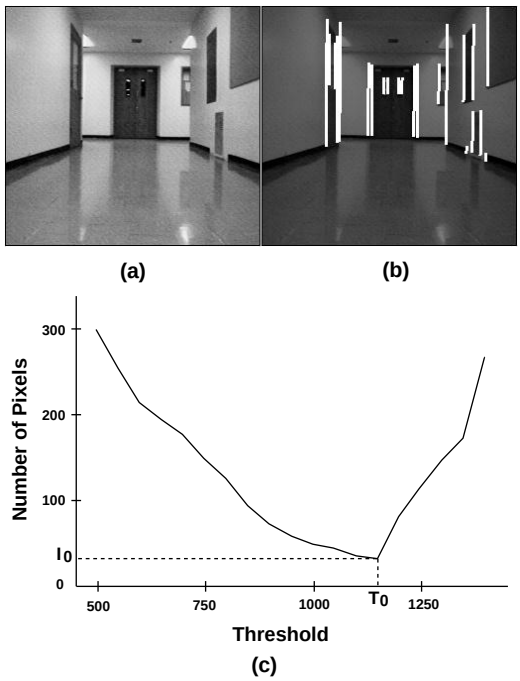

Figure 5: (a) Camera image. (b) Vertical lines in the rendered expectation map overlaid on the camera image. (c) Plot showing the number of difference pixels N_m vs. the threshold T.

edges and the artifact edges will be suppressed. As the threshold is increased, there will come a point where no edges will be extracted from a camera image. So large values of T will yield a high count for the number of pixels in the difference image. At the other extreme, when T is too small, the camera image will yield an excessively large number of edges, most corresponding to random gray level fluctuations in the camera image. So the number of difference pixels will again be large. In between these two extremes, there will be a value for T, designated T_0, for which the number of difference pixels N_m will be a minimum denoted by N_0. Shown in Figure 5(a) is a camera image taken from a position where the rendered expectation map looks like what is shown by the overlaid white lines in Figure 5(b).[1] Shown in Figure 5(c) is a plot of the difference pixels N_m obtained in Step 8 above for different values of the threshold T. Also marked in this figure are the threshold T_0 and the number N_0 of pixels that corresponds to the threshold T_0.

To illustrate a result of the 10-step procedure for determining the adaptive thresholds for an entire section of a hallway, shown in Figure 6 is a section of the hallway immediately outside our laboratory. The robot was manually driven along the trajectory marked by x's. Along this trajectory, expectation maps were rendered at regular intervals and the corresponding camera images recorded. (In order not to jumble up the display in Fig. 6, only every fourth position where images were rendered and recorded is shown in the figure.) The edge detection thresholds T_0 and the numbers of pixels N_0 corresponding to T_0 are shown in Figure 7. This graph constitutes the table used in the adaptive thresholding process during autonomous navigation.

Obstacle Detection. Figure 8 shows the flow of computations for the obstacle detection procedure by vision. During autonomous navigation, in accord with the statements made at the beginning of Section 3.1,

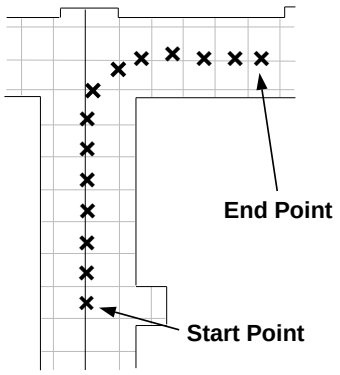

Figure 6: The robot's trajectory for the learning of adaptive thresholds.

[1] Only the vertical edges of the rendered expectation map are displayed as overlaid white lines. That's because we have found it sufficient to use just the vertical lines for the detection of stationary obstacles. However, the entire procedure can be readily extended to edges of arbitrary orientation, albeit at a higher computational cost.

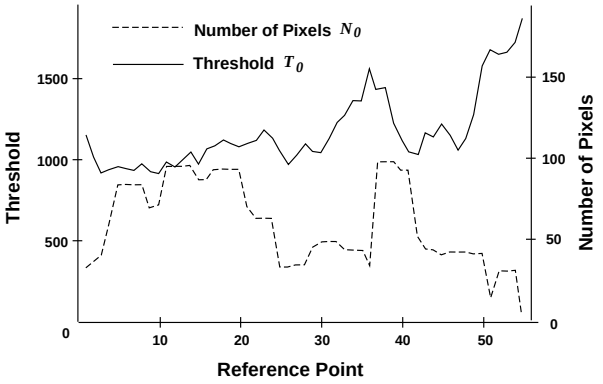

Figure 7: The edge detection thresholds T_0 and the minimum number of difference pixels N_0 for each reference point.

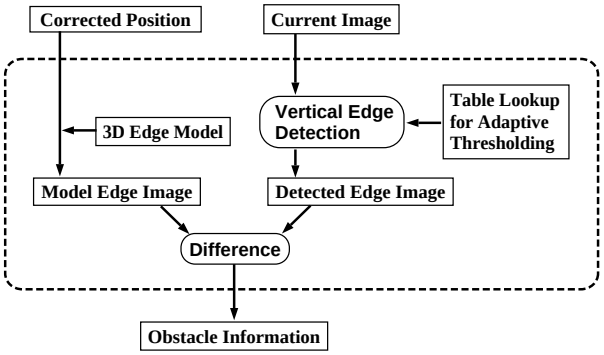

Figure 8: The obstacle detection procedure using vision during autonomous navigation.

obstacles are found from the difference of vertical edges in the camera image and the expectation map immediately after each exercise in self-localization. As shown in Fig. 8, model vertical edges in the scene are estimated from the 3D edge model of the environment using the robot's position corrected by the self-localization first. Next, vertical edges are extracted from the camera image using for the detection threshold a value that corresponds to the nearest position in the table lookup. The vertical edges thus found are compared with the model vertical edges. Any extraneous vertical edges must come from any obstacles in the environment.

Figure 9 shows sample images for a typical exercise in obstacle detection by vision. Displayed by overlaid white lines in (a) are the model vertical edges as obtained from the expectation map rendered from the environment model. The tall dark object on the left is a wooden box that serves as an obstacle. Shown in (b) are the vertical edges extracted from the camera image using the nearest optimum edge-detection threshold T_0 from the table of such thresholds for the hallway in question. Shown in (c) is a difference edge image for the vertical edges extracted from the camera image and the model vertical edges. Shown in (d)

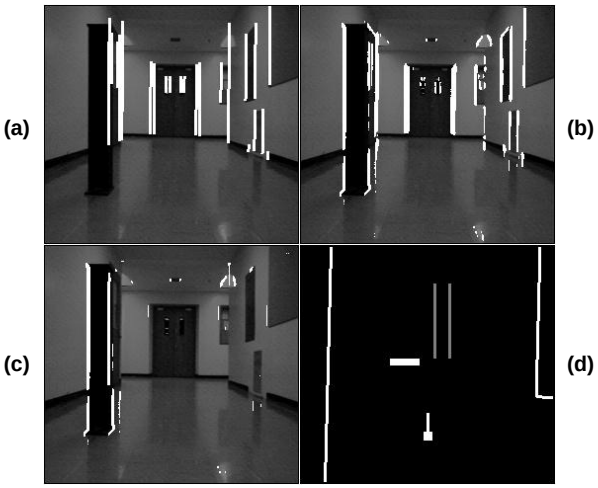

Figure 9: Sample images for obstacle detection by vision. (a) Model vertical edges overlaid on the camera image. (b) Detected vertical edges from the camera image. (c) The difference edge image. (d) Robot's position, obstacle information and passage space in the hallway.

are robot's position, obstacle information and passage space in the hallway. The robot is shown as an inverted 'T' icon, the obstacle by a small white rectangle, and the safe-passage direction by two closely spaced parallel lines.

Determining the Direction of Safe Passage. To find a direction for safe passage in the presence of obstacles, as pointed out in the ten-step procedure outlined in the previous subsection, the difference edge image (an example of which is Figure 9(c)) is divided into five vertical regions, as shown in Figure 10(a) where a tall dark obstacle is present on the left. Number of pixels in each of the five regions is summed. If this sum exceeds the threshold N_0 for this location of the robot, the directions corresponding to that vertical region in the camera image are considered to be unsafe.[2] Recall from our previous discussion, N_0 is the smallest number of N_m which is the maximum of the pixel numbers N_1 through N_5 counted in the five vertical regions of the difference edge image, and N_0 results from the application of the optimum threshold T_0. Of course, as with the application of the edge detection thresholds T_0, the location for which N_0 is available will often not coincide exactly with the current location of the robot. So, as was the case for edge detection, N_0 for the nearest entry in the table is used.

[2]The threshold that is actually used for detecting obstacles is $(1+\epsilon) \times N_0$ where a non-negative ϵ accounts for the nearly always-present discrepancy between the nearest location of the robot where N_0 is available and the location of the robot where it is needed during navigation. A value of $\epsilon = 0.5$ has worked for us in almost all our experiments. In most cases, N_0 is less than 100, hence the threshold is less than 150. This margin of 50 pixels wouldn't be a large value considering various noises on the image.

Shown in Figure 10(b) are the summed values for each of the five regions marked in (a) of Figure 10.

The view-space in front of the robot is given one of three labels: 1) obstacles, 2) unknown, and 3) open space. Initially, the directions corresponding to 'obstacles' are made to correspond to those of the five vertical regions in the camera image whose difference pixel sum exceeds N_0, as explained above. By the same token, initially the directions corresponding to 'unknown' are all those that are outside the view cone corresponding to the camera image. Both the 'obstacles' and the 'unknown' are expanded by half the width of the robot to ensure safe clearance. The rest of the directions then yield us the directions of the safe passage for the robot. Since the width of the *YAMABICO* robot is assumed to be twice as large as one of the vertical regions in Figure 10 at a distance of one meter in front of the camera, free passage space is narrowed to the size of one vertical region on the right of the obstacle in Figure 10(c). Distance to obstacles cannot be computed from the visual information, since only a single camera is used in this method. It is for this reason we make the conservative assumption that all obstacles and unknown regions are situated 1 meter in front of the robot. After obstacle avoidance, the robot seeks to approach its originally planned path to the destination.

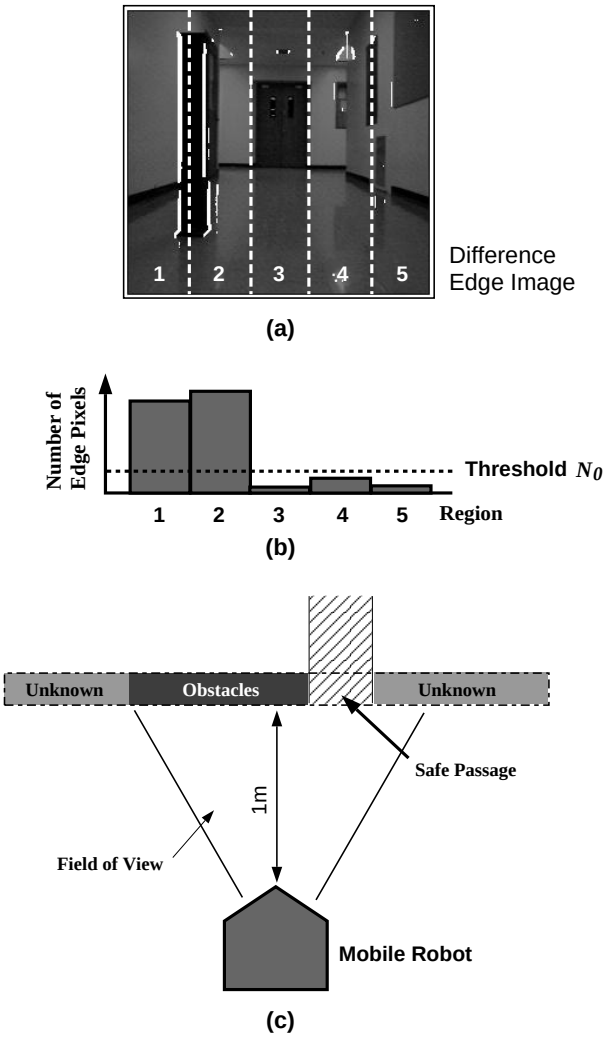

Figure 10: Determining the direction of safe passage in the presence of obstacles. (a) Division of the difference edge image into five vertical segments. (b) Total number of pixels in each of the five vertical segments of the difference edge image. (c) Direction of the safe passage obtained by excluding 'obstacles' and 'unknown' regions. Both the 'obstacles' and the 'unknown' are expanded by half the width of the robot to ensure safe clearance.

3.2 Moving Obstacle Detection

As mentioned before, the currently implemented vision system is effective only for detecting stationary obstacles because of limitations on the computing power available to the robot. The ultrasonic sensors, which can measure the distance to the object in almost real time by the pulse-echo technique, are therefore used for detecting moving obstacles. Of course, since the ultrasonic sensors have no intrinsic way of distinguishing between moving and stationary obstacles, the interaction between the two sensor systems – the vision sensor and the ultrasonic sensors – becomes important. How the robot is controlled by the two sensor systems is predicated on the following attributes of the sensors:

1. The view angles of the two sensor systems are nearly identical (60 degrees).
2. By using appropriate time gating of the received echos, the ultrasonic sensors do not detect an obstacle if it is farther than 50cm from the robot.
3. The vision sensor is capable of detecting stationary obstacles at ranges far exceeding 50cm.

Shown in Figure 11 are the view angles for the two sensor systems. In the next section, we will explain how the two sensor systems interact in light of the attributes listed above.

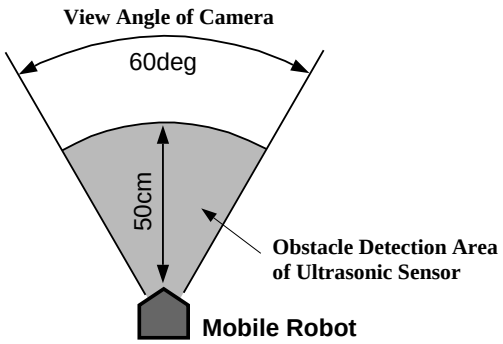

Figure 11: View angle of the camera and detection area of the ultrasonic sensor.

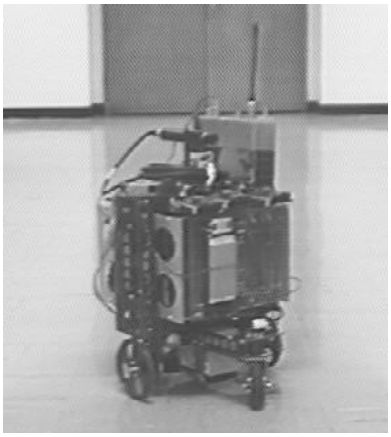

Figure 12: Autonomous mobile robot *YAMABICO*.

4 System Architecture

The navigational system described in this paper was implemented on the *YAMABICO* robot[10] shown in Figure 12. Because this robot doesn't have a powerful image processing module, a workstation is used as an off-board image processing unit at this time. The robot communicates with the workstation over two links, a video link for the transmission of images and a two-way RF link for data. An image is taken by the camera mounted on the robot and it's sent to the workstation over the video link. The received image is processed on the workstation and the resulting data sent to the robot over the RF link. The camera used here is XC-999 (SONY) and its iris was fixed. As was shown previously in Figure 11, the horizontal angle of viewing field is 60 degrees approximately. The image is digitized into 256 $\times$ 240 pixels. *YAMABICO* has ultrasonic sensors and three pairs of ultrasonic transmitters and receivers can monitor about 60 degrees of the view-space in front of the robot, in accordance with Figure 11. Since we wanted to predominantly use vision for collision avoidance, time gating was used to limit the range of ultrasonic collision avoidance to 50cm.

We will now explain how the two sensor systems interact for collision avoidance. When the robot faces a stationary obstacle, the obstacle is first detected by vision since the range of ultrasonic sensors is limited to 50cm. In the manner explained previously, the vision data is used for the calculation of the direction of safe passage and the robot turns toward this direction. For those obstacles that are avoided on the basis of vision data, the ultrasonic sensors will usually not record any echos for the simple reason that the robot has already steered away from them. If per chance one or more of the ultrasonic sensors do receive echos from such an obstacle, the robot comes to a halt and uses its vision again for calculating afresh the direction of safe passage. In case of a moving obstacle, of course,

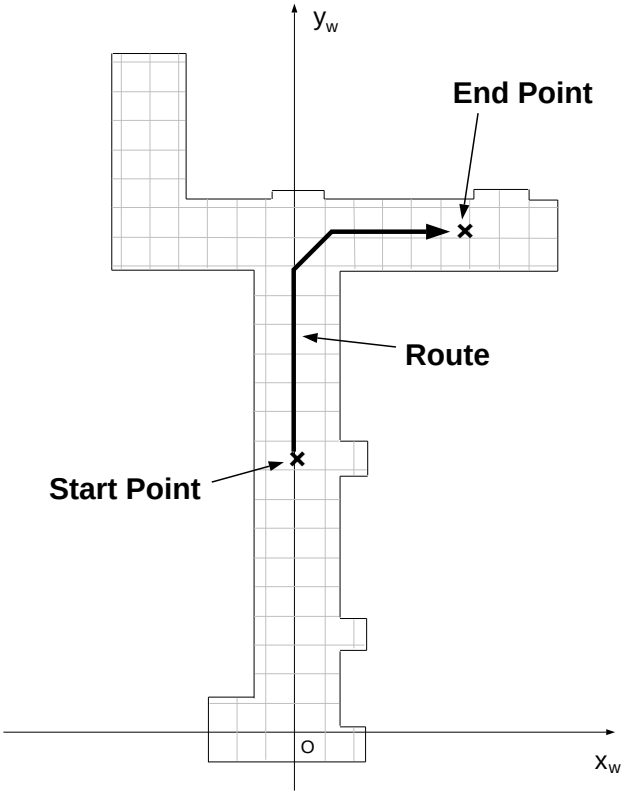

Figure 13: Map of hallway. The black line denotes planned path.

it could be detected by vision if the obstacle makes its appearance just at the moment the camera image is recorded. Such an obstacle would be treated in the same manner as a stationary obstacle. Of course, the obstacle such as this would have moved to a different location by the time the robot arrives in its vicinity. In such cases, the robot would seem to be avoiding phantom obstacles, but not to any great detriment of the overall navigational performance. Note that after each exercise in collision avoidance, the robot seeks to go back to its planned path to the destination. If the moving obstacle gets too close to the robot, the ultrasonic sensors take over and bring the robot to a momentary halt. Subsequently, the vision based processing is re-initiated for further navigation.

5 Experimental Results in Autonomous Navigation

Using the system presented in the previous section, several experiments were performed in the hallway shown in Figure 13. The size of the grid shown by the dotted lines is 1 meter on each side.

An example of autonomous navigation on the planned path, as displayed by the black line in Figure 13, is shown in Figure 14. In this figure, the robot's positions corrected by each self-localization exercise are shown as x's. The total length of the path is approxi-

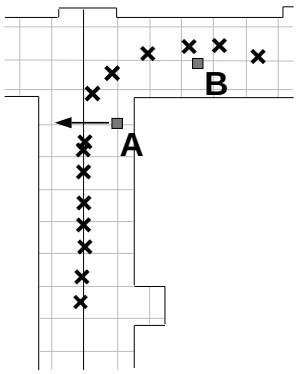

Figure 14: An example of autonomous navigation. The robot's actual trajectory is shown as x's. A moving obstacle 'A' crossed the path and there was a stationary obstacle 'B'.

mately 12.5 meters. During this experiment, a moving obstacle 'A' (a human) suddenly crossed the path just in front of the robot. The ultrasonic sensors brought the robot to a halt, followed by the invocation of vision sensors for determining the direction of safe passage. The robot also successfully avoided a stationary obstacle 'B' (a wooden box) by using vision.

The speed of the robot was set at 10cm/s for this experiment. The processing time for one image was approximately 10 seconds on the workstation (a SUN SPARCstation 20).

6 Discussions and Conclusions

We presented in this paper a vision-based navigational system for mobile robots that is also capable of avoiding at least the stationary obstacles using vision data. By using a combination of model-based vision for self-localization; retroactive position updating to cope with the time delays associated with image processing; using vision data for not only self-localization but also for the calculation of directions of safe passage in the presence of obstacles; and ultrasonic sensors for the detection of close-range moving obstacles; we have created a navigational system that makes optimum use of all the sensors for smooth and continuous navigation in indoor environments.

As with all such systems dealing with higher-level robotic intelligence, the performance can never be expected to be completely foolproof. The best that one can do is to devise appropriate automatic error correction and detection strategies. To briefly discuss the various failure modes of our system, the vision-based collision avoidance capability depends obviously on the visual contrast between the obstacle and the interior of the hallway. The size of the obstacle will also play an important role in its detectability by vision. To gain an understanding of these limitations, we performed experiments using two small cardboard boxes of two different colors, brown and white, each of height 35cm and width 24cm, as test obstacles. As for the results, the robot was able to detect the white box in all cases, but in 50% of the cases failed to detect the brown box. It is entirely possible that superior image processing strategies would enhance the performance of vision-based collision avoidance. Our future research will address this issue.

Acknowledgments

The authors would like to thank very much Dr. T. Tsubouchi, Mr. S. Maeyama and Prof. S. Yuta of the Intelligent Robot Laboratory of the University of Tsukuba for their useful suggestions and support of this research.

References

[1] T. Tsubouchi and S. Yuta: "Map Assisted Vision System of Mobile Robots for Reckoning in a Building Environment," *Proc. 1987 IEEE Intl. Conf. on Robotics and Automation*, pp. 1978-1984, 1987.

[2] K. Sugihara: "Some Location Problems for Robot Navigation Using a Single Camera," *Compt. Vision Graphics Image Process.*, **42**, pp. 112-129, 1988.

[3] A. Kosaka and A. C. Kak, "Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties," *CVGIP–Image Understanding*, **56**(3), pp. 271-329, 1992.

[4] Muńoz, A. J. and J. Gonzalez: "Localizing Mobile Robots with a Single Camera in Structured Environments," *Intl. Symposium on Robotics and Manufacturing in the World Automation Congress '96*, 1996.

[5] Y. Matsumoto, M. Inaba and H. Inoue: "Visual Navigation Using View-Sequenced Route Representation," *Proc. 1996 IEEE Intl. Conf. on Robotics and Automation*, **1**, pp. 83-88, 1996.

[6] T. Ohno, A. Ohya and S. Yuta: "Autonomous Navigation for Mobile Robots Referring Pre-recorded Image Sequence," *Proc. 1996 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, **2**, pp .672-679, 1996.

[7] T. Camus, D. Coombs, M. Herman and T. Hong: "Real-Time Single-Workstation Obstacle Avoidance Using Only Wide-Field Flow Divergence," *Proc. 13th Intl. Conf. on Pattern Recognition*, 1996.

[8] A. Kosaka, M. Meng and A. C. Kak: "Vision-Guided Mobile Robot Navigation Using Retroactive Updating of Position Uncertainty," *Proc. 1993 IEEE Intl. Conf. on Robotics and Automation*, **2**, pp. 1-7, 1993.

[9] S. Maeyama, A. Ohya and S Yuta: "Non-Stop Outdoor Navigation of a Mobile Robot –Retroactive Positioning Data Fusion with a Time Consuming Sensor System–," *Proc. 1995 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, **1**, pp. 130-135, 1995.

[10] S. Yuta, S. Suzuki and S. Iida: "Implementation of a small size experimental self-contained autonomous robot – sensors, vehicle control, and description of sensor based on behavior–," R. Chatila et al. Eds, *Experimental Robotics II*, pp. 344-359, Springer-Verlag, 1993.