Autonomous Navigation System Design for a Smart Robotic Rover

Yi Cheng, Kathleen Hayden, Zekeriya Aliyazicioglu, Tim Lin,
California State Polytechnic University, Pomona

1. Introduction

In response to the NASA Research Announcement of 2002, California State Polytechnic University, Pomona (Cal Poly Pomona) submitted a proposal for the "Partnership Awards for the Integration of Research into Undergraduate Education" (PAIR) program. The purpose of the PAIR program is to integrate cutting-edge NASA-related research into the undergraduate curriculum. Cal Poly Pomona proposed to incorporate the Jet Propulsion Laboratories (JPL) robotic technology research into the undergraduate curriculum of the Electrical and Computer Department, the Engineering Technology Department and the Computer Science Department. Our proposal, "Deep Space Exploration using Smart Robotic Rovers", was selected for funding and we established our first interdisciplinary team of students and faculty to develop a smart robotic rover.

During the last two years, students and faculty participating in this program have developed a robotic rover that has successfully accomplished the initial goals of the project. The rover is capable of climbing 30^o inclines, rotating about its center axis, strafing, and maneuvering diagonally while maintaining stability. It was also designed to protect the vital internal components from outside contaminants and provided mechanical support for all externally mounted equipment.

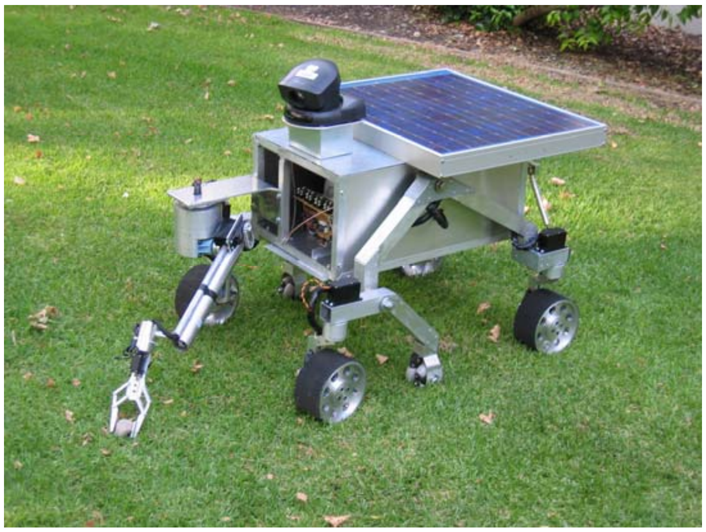


Figure 1. Smart Robotic Rover Prototype

This year we have a more ambitious goal to develop and implement autonomous navigation algorithms that can be used for navigation of the robot without human intervention. The robot will use camera images to move from one location to another and to navigate around any possible obstructions in its path. It will be capable of identifying and locating a designated target. Stereographic images will be used to garner depth information for navigational purposes.

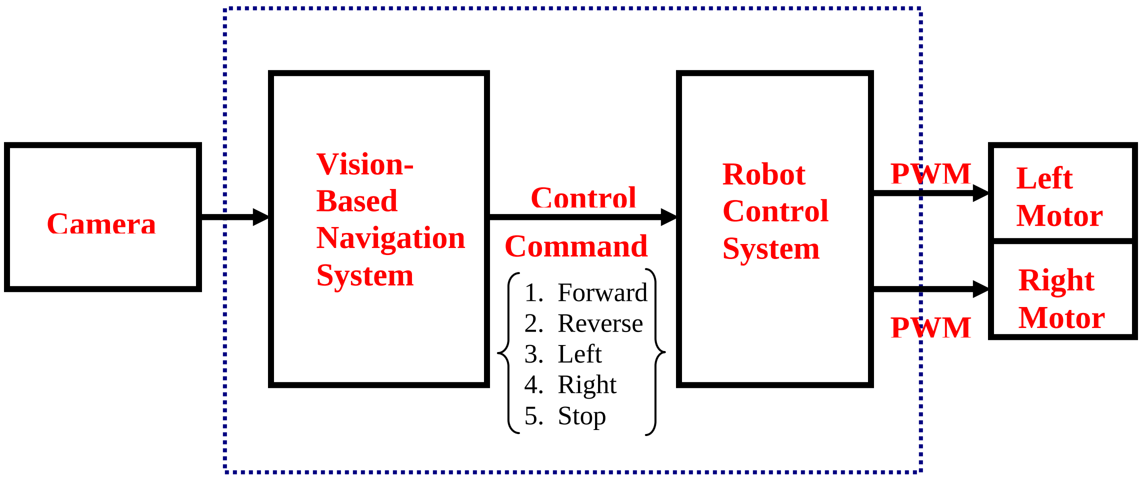


Figure 2. Vision-based Navigation & Control System

In this paper we will discuss our efforts to design a vision-based robot control system, our efforts to ascertain depth information using stereovision, and the creation of a database containing known and determined image information to support autonomous navigation.

2. Educational Objectives

Teams composed of faculty and students are engaged in the development of a robotic rover, capable of being deployed in an unfriendly environment, equipped with an autonomous navigational system, designed to collect scientific data and communicates through a wireless network with the base station during its deployment. Juniors and seniors are recruited to participate in this effort. Participating students are asked to actively engage in the project activities for six quarters, three while they are juniors and three while they are seniors. Through their participation and contributions towards the mission of this project, students receive degree credit. Juniors are awarded "Upper Division Elective Credit" and seniors fulfill their degree capstone requirement, "Team Senior Project". Currently, we have more that 70 students and 14 faculties participating in this effort. The students are sub-divided into nine different sub-teams each with one or more faculty advisors. Each sub-team is composed of students and faculty from the engineering, technology and computer science.

Students enrolled in the project agree to commit a minimum of six hours per week to the project. Sub-team meetings are held each week of the quarter. And all teams meet together three times each quarter to present their progress towards the goals of the sub-team and to coordinate events and tasks for the project. Each spring, the College of Engineering holds a "Symposium Day" to showcase the activities of all "Team Senior Projects" that have been completed during that year.

Figure 3. NASA PAIR Team Picture

Each sub-team in the NASA PAIR group presents their accomplishments to the invited guests, students and faculty members. The event concludes with a demonstration of the rover.

Students participating in this project are aggregated into multi-disciplinary teams. The tasks for this project include hardware design (both mechanical and electrical), software design and control systems. Students are challenged to envision a system solution requiring contributions from many different disciplines. As a result they are exposed to topics, tools and technology that would not otherwise be a part of their undergraduate training. They benefit from participating as a team member on this project by applying the knowledge they have gained in the classroom to a real-world engineering problem. Additionally, they benefit by interacting with students and faculty from other disciplines such that they acquire a greater appreciation of both the technical and social complexities of participating in a team effort to solve a complex task.

3. A Vision-Based Robot Control System

The mission requirements for a vision-based control system are: (1) to move the robot to a specified target using an on-board digital camera and computer, and (2) to avoid obstacles without human intervention

A vision-based robot control system assumes that the target is visible and can be identified from the robot's digital camera images. The control system will search for the target and move towards it; it will detect obstacles along its path and make detours to avoid them. The robot will automatically stop when it nears the target.

The system does not require the position coordinates of the target, and does not require real-time position measurements. This approach does not rely on database containing geographical information. In order to move the robot to the target, the controller attempts to align and maintain the target image at the center of the camera's focus while progressing towards the target. Obstacle avoidance will be accomplished by moving to the left or right of the obstacle.

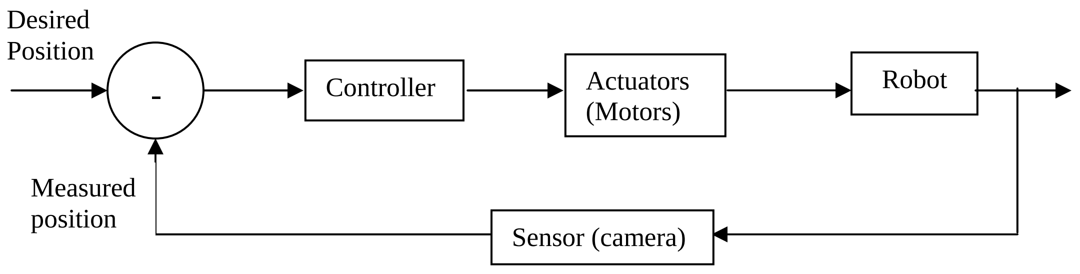


Figure 4. Block Diagram of A Vision-Based Robot Control System

The vision-based control system is implemented as a finite state machine on a Xilinx Spartan-3 FPGA board with a 1-Mbyte of static RAM to store images. There are four states as shown in Figure 5.

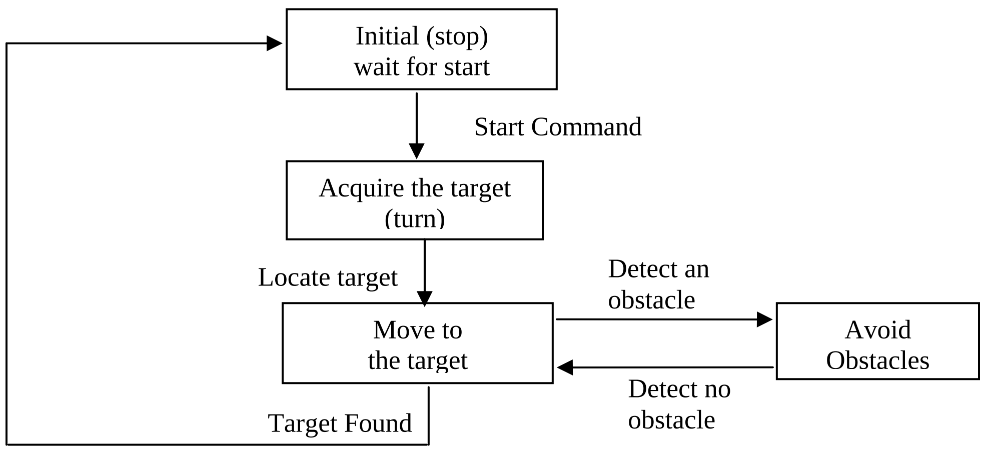


Figure 5. State Diagram of A Vision-Based Robot Control System

1. Initial state: After power-up, the controller initialize all variables, allowing a user to enter new target image in the FPGA memory, and wait for the Start command to transition to the second state. The robot will remain stationary.

2. Acquiring Target state: In this state the controller does not know the direction nor the distance to the target. The robot will be commanded to rotate 360 degrees until it locates the target in the center of the camera image. As the camera searches the surrounding area for the target it will continuously compare the image information detected by the camera to the specified target information. The controller will use color and shape information to help identify the target. If there is a good correlation between the desired target and the image detected by the camera, then a transition to the next state will occur.

3. Moving-to-the-Target state: The robot will be commanded to move toward the target while constantly aligning the target to the center of the camera image. For example, if

the target image is to the left of the center, then the robot has moved to the right of the target. In order to make proper corrections, the controller will command the right motor to increase its speed and the left motor to decrease its speed until the target image is back at the center of the camera image.

Anytime that the robot detects an object other than the target in its path, i.e. near the center of its image, it will transition to the fourth state, Avoiding-Obstacle state.
The robot controller will determine that it has reached the target by either calculating the size and location of the target contained in the camera image, or by the activation of the front bumper switch. Once the rover has reached the target, it will transition to the Initial State to wait for a new command.

4. Avoiding-Obstacle state: The robot will be commanded to turn right or left to avoid hitting the obstacle. The controller will attempt to pass the obstacle at a safe distance to the right or left by placing the aimed point at the horizontal center of the image. Thus, the controller uses the images of the obstacle to help it to navigate around the obstacle. After the obstacle is no longer in the image, the robot has moved away from the obstacle. It will return to the Moving-to-the-Target state.

4. Stereo Images

Stereovision, similar to human eyesight, is the ability to capture two different image perspectives of the same real world scene at a specific instant in time. Combining the two dimensional perspective from each camera makes it possible to compute the third dimension, depth.

In a stereovision system, depth information pertaining to objects in the scene can be obtained by calculation of the disparity. Our goal is to identify matching points in the two images, compute the disparity between the left and right image and calculate the depth [1]. To simplify this case, we used two cameras having the same focal length, mounted at the same height and spaced a fixed distance apart. Therefore, corresponding image points differ only in their x-coordinates as seen in Figure 6 [2].

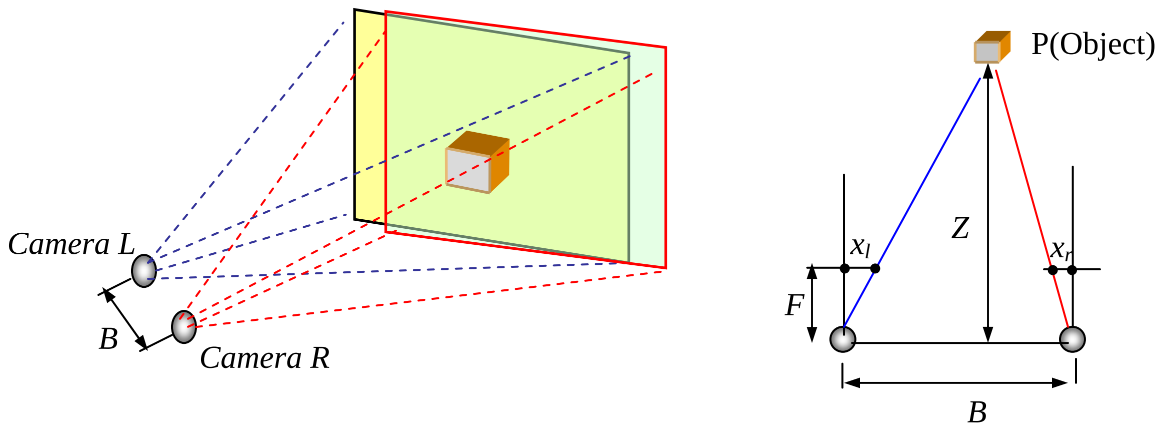


Figure 6. Parallel camera configuration

If the distance between the two cameras (B) and two cameras have the same focal length (f) are known, then the disparity vector has only a horizontal component, which is given by [2]

$$d_x = x_l - x_r = f\,\frac{B}{Z}$$

We can measure d_x using matching points in MATLAB, hence the depth can be obtained as

$$Z = f\,\frac{B}{d_x}$$

In our pilot study, we used image-capturing tools available in MATLAB to acquire simultaneous images from two Web cameras mounted in parallel, see Figures 7a and 7b. The distance between the two web cameras was set to a constant value. The two images were superimposed such that the disparity between the objects was apparent in the combined image. The larger the disparity the closer the object is to the camera, and the smaller the disparity the farther the object is away from the camera. Due to limited camera availability, our pilot study used two comparable Web cameras with unknown focal lengths. We measured the distance between the cameras, the distance from the objects, and computed the disparities for the objects to calculate the focal length.

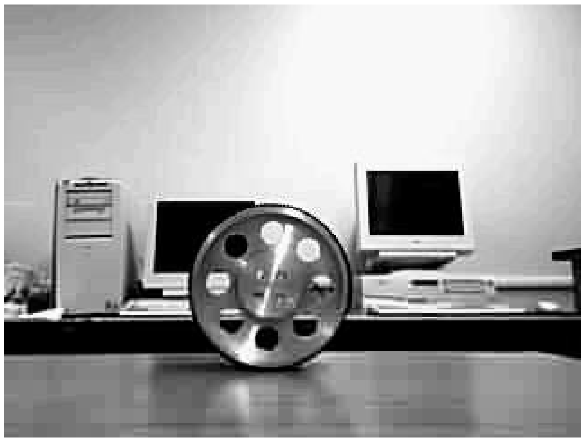


Figure 7.a. Left Camera Image

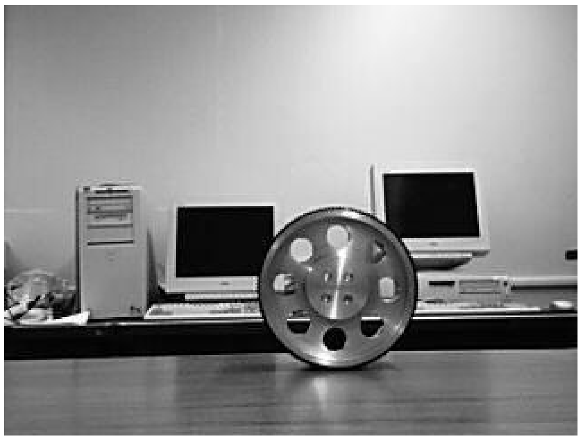


Figure 7.b. Right Camera Image

4.1 Superimposing Images and Pixel Disparities

The left and right gray scale images from Figures 7a and 7b are processed using an edge-finding function in MATLAB and then superimposed into a single image. The outlined edges help to locate exact reference points for finding pixel disparities. Figures 8a and 8b show the processed left and right edge images.

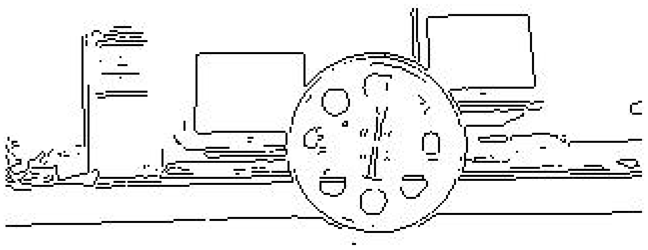


Figure 8.a. Left Edge Image

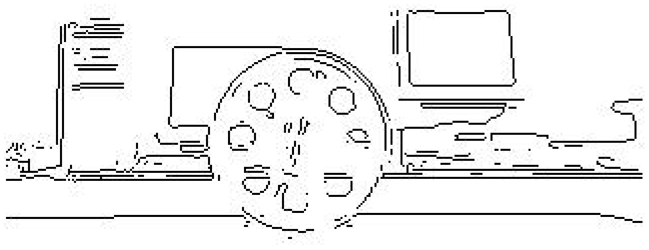


Figure 8.b. Right Edge Image

To determine the disparity between objects shown in the left and right images, a superimposed view containing both images can be used. In the resulting image, it is simply to locate exact points on the same object and measure the disparity in pixels. Using the MATLAB function "pixval", the user may manually drag a horizontal line between any two points on the image to calculate the distance in pixels. Figure 9 shows the superimposed image and the distance between reference points in pixels as determined by MATLAB function "pixval".

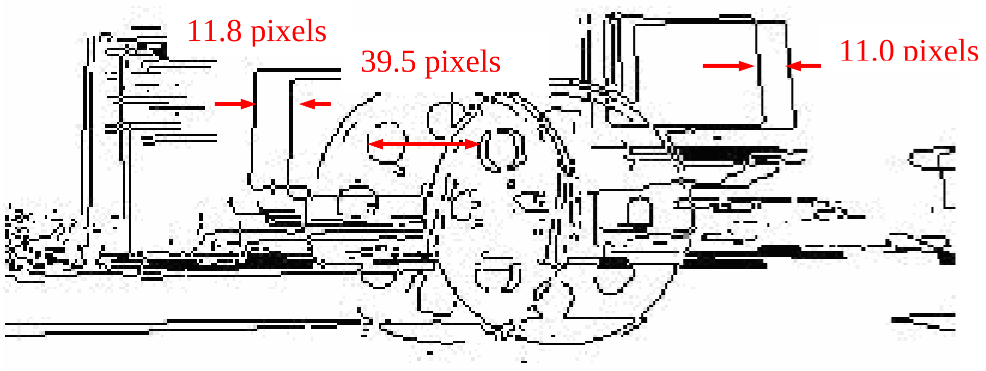


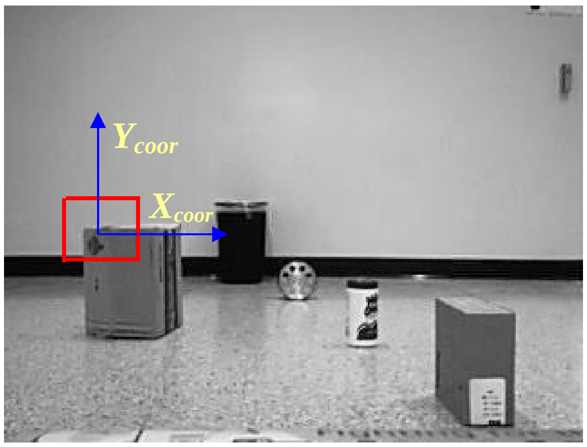
Figure 9. Superimposed Edge image and disparity

	Disparity (pixels)	Distance (cm)	Focal length
Computer-left	11.8	227	334.8
Computer-right	11	242	332.75
Wheel	39.5	62.5	308.59

Table 1: Measurements on objects in image (B=8cm)

4.2 Windows Disparity

To automate the measurement of disparity in the image we chose a small region in the left image, called "correlation window" and located the same group of pixels in the right image. Then we measured the distance between the windows representing position (x, y) in the left image and the one representing (x+d, y) in the right images. We repeat this process for every correlation window in the left image (See Figures 10.a and 10.b). After completing all windows in the left image, we capture new images to repeat same process. The size of the correlation windows affects the processing time. In this experiment gray scale images were used to determine the disparity. To improve the signal to noise ratio, color images can be used.



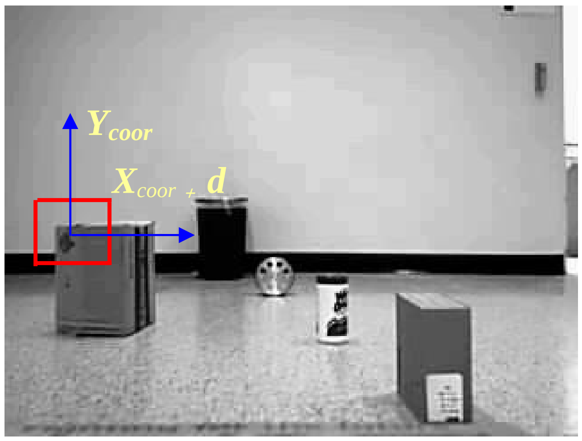
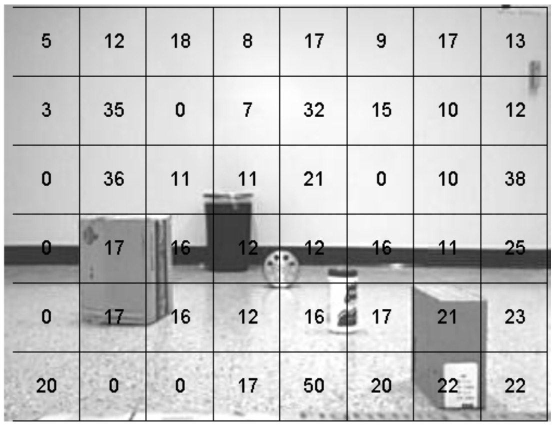


Figure 10.a Left Camera Image

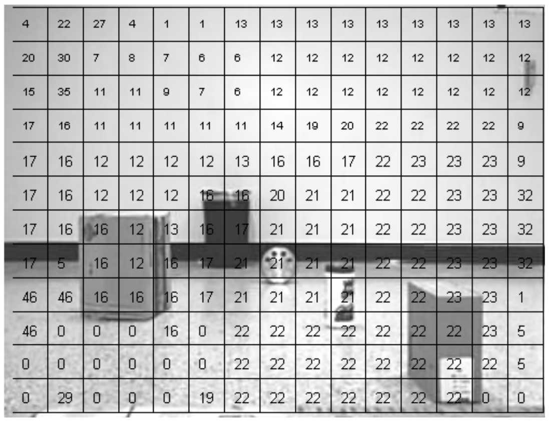
Figure 10.b Right Camera Image

For this experiment the "corresponding window" size was selected as a square of 40x40 pixels. Stepwise from the upper left corner of the image, the "corresponding window" of the left image is compared to its counterpart in the right image and the disparity value calculated. This cycle is repeated until every window section of the left image has been sampled and the disparity value determined. The program returns an array of disparity values that can be superimposed on the original image. Figure 11a shows the result of this experiment. If more accurate measurements are needed, the program can be modified to take image square of 20x20 pixels (Figure 11.b).



5	12	18	8	17	9	17	13
3	35	0	7	32	15	10	12
0	36	11	11	21	0	10	38
0	17	16	12	12	16	11	25
0	17	16	12	16	17	21	23
20	0	0	17	50	20	22	22

Figure 11.a Disparity Distribution for "corresponding window"= 40x40 pixels



4	22	27	4	1	1	13	13	13	13	13	13	13	13	13
20	30	7	8	7	6	6	12	12	12	12	12	12	12	12
15	35	11	11	9	7	6	12	12	12	12	12	12	12	12
17	16	11	11	11	11	11	14	19	20	22	22	22	22	9
17	16	12	12	12	12	13	16	16	17	22	23	23	23	9
17	16	12	12	12	16	16	20	21	21	22	22	23	23	32
17	16	16	12	13	16	17	21	21	21	22	22	23	23	32
17	5	16	12	16	17	21	21	21	21	22	22	23	23	32
46	46	16	16	16	17	21	21	21	21	22	22	23	23	1
46	0	0	0	16	0	22	22	22	22	22	22	22	23	5
0	0	0	0	0	0	22	22	22	22	22	22	22	22	5
0	29	0	0	0	19	22	22	22	22	22	22	22	0	0

Figure 11.b Disparity Distribution for "corresponding window"= 20x20 pixels

The pixels disparities are then compared with distances from the camera. A summary of these calculations is shown below in Table 2.

	Disparity (pixels)	Calculated Distance (cm)	Measured Distance (cm)	Error (cm)
Box 1	24.464	124.26	118	-6.264
Wipes	17.08	177.98	177	-0.985
Box 2	16.204	187.60	188	0.392
Wheel	12.701	239.35	264	24.648
Trash Box	10.073	301.80	318	16.203

Table 2: Measurements on objects in image

5. Database in Robot Navigation as one of Future Approaches

5.1 Background

Typically, photos are taken either using a mono-vision or stereovision camera during the robot's navigation to a desired target as discussed previously in this document. Developing a 3D image database for navigational purposes is one of our viable future objectives. It will be discussed below with the results to be presented in the future.

In general as the rover is progressing towards the target, new images are taken to determine the rover's path and distance to the desired target. This approach is fine if there is line of sight between the robot and the target. However, if the rover encounters obstacles in its path that obscures its ability to see the desired target then the rover must rely on previously taken images, objects in the images, and the landscape data in order to determine the required path to the target. A database subsystem will seek to maintain historical image data collected by the rover and save related information, such as the objects identified in the image, the rover's location when the image was taken, the orientation of the robot, etc.

5.2 Database Definition

The database will be a relational database (RDBMS) that consists of several tables linked by relations. The database schema is not yet defined, but will consist of the following data, which will be defined, in more detail at a later date:

Image Data:

- Whole Image
- Objects in the image using pattern recognition algorithm
- Landscape and geography data

Text Data:

- Time in hours, minutes, seconds etc. (This data is dependent upon on how fast the robot moves and length of time it needs to track. So it has not been decided whether coarser data like days, weeks or finer data like centi-seconds, milliseconds need to be defined). Time data can be used to determine how long it takes a robot moves along a path.
- Location of the robot using some coordinate system. It remains to be specified whether the location is 2D or 3D and also what unit of location will be used (foot, inch, or grid unit etc.). Location data can be used to compute the distance traveled by the robot.
- Speed: It remains to be determined whether speed data will be derived from distance data divided by time of travel or speed data can be measured physically from the rover.
- Orientation: Does the robot face east, west (on Mars)? This answer depends upon the coordinate system selected for the mission.

5.3 Database Implementation

Initially, MySQL has been selected for the database implementation together with an appropriate computer language. Due to familiarity, C++ is the language of choice. Alternate database implementations, such as DBMS, will be further examined for future consideration.

5.4 Database Operations

The followings database operations will be considered.

- Database storage
- Database retrieval
- Database backup
- Database compression
- Database transmission – send and receive data from location of computer to location of the rover

5.5 Database Issues

There are many issues to be considered or resolved for the database to be constructed. Some of the issues to be resolved are stated below.

- Frequency of picture taking: It is not clear how often the pictures should be taken. The robot speed is a factor in this consideration.
- File size of every picture: Will picture be color or black and white? What resolution should be used to have the best and fastest results (for the camera supporting multiple resolutions, the finest resolution will produce biggest picture and cause slowest processing but may be the best decision for autonomous navigation)?
- Picture processing: What kind of pattern recognition algorithm(s) will be used?
- Image Processing before storage: Is it necessary to modify the image to throw away unnecessary “noise” in the pictures?
- Communication: Is it helpful to transmit the image from the robot database to the base station? If so, how often and how much information should be sent?
- Database security: Is security a concern?

5.6 Database Usage

Once implemented a database (together with all the associated algorithms can be very useful in autonomous navigation:

- Find the distance to the target (to adjust the robot speed)
- Remember the locations, directions, and speed data (for future navigation planning such as more complicated paths).
- Navigation autonomously in the area from point to point or backward and also repeats the same path with faster speed.
- Build a map

Once a database system to support robot navigation has been developed other interesting possibilities exist, such as "Can the robot learn?" With the database recording past successes and failures, the robot can avoid the paths leading to failures and when confronted with multiple possible paths can compute the most economic path.

References:

[1] Sethuraman, S. Jodan, A. Siegel, M. "*Multiresolution based hierarchical disparity estimation for stereo image pair compression*"
[2] M. Sonka, V. Hlavac, R. Boyle , "Image processing Analysis and Machine Vision", Second edition, PWS Publishing