Scene Reconstruction and Interpretation for Robot Navigation.*

Benedict Wong† and Minas E. Spetsakis
Dept. of Computer Science, York University
4700 Keele Street, North York, ONTARIO
CANADA, M3J 1P3
tel (416) 736-5053
fax (416) 736-5872
bwong@swi.com, minas@cs.yorku.ca

ABSTRACT In this paper, we present an approach to autonomous robot navigation in an unknown environment. We design and integrate algorithms to reconstruct the scene, locate obstacles and do short-term field-based path planning. The scene reconstruction is done using a region matching flow algorithm to recover image deformation and structure from motion to recover depth. Obstacles are located by comparing the surface normal of the known floor with the surface normal of the scene. Our path planning method is based on electric-like fields and uses current densities that can guarantee fields without local minima and maxima which can provide solutions without the need of heuristics that plague the more traditional potential fields approaches. We implemented a modular distributed software platform (FBN) to test this approach and we ran several experiments to verify the performance with very encouraging results.

1. Introduction

Autonomous Platforms that can navigate through a cluttered environment to a goal have a tremendous potential for applications. A typical autonomous platform consists of a vehicle, a set of sensors, a data communication link and on-board computer. The vehicle can be a wheeled robot, a tracked vehicle, a limbed robot, an aquatic vehicle, an aerial robot etc. The set of sensors may include sonar, infrared sensors, bumper switches, laser range finders, video cameras or even a Global Positioning System. Both the vehicle and the sensors are controlled by the on-board computer or other computers via the data communication link. The computer analyses the sensor data and based on the user inputs, guides the autonomous platform to the goal position. Other than performing tasks such as goal selection, most of the navigation is done by computer.

In this paper, we concentrate on the type of navigation in which the robot has no predefined information about the environment except that it contains a mostly empty flat floor. Here, the robot's task will be to navigate around obstacles on the floor. This kind of navigation is often called short-term navigation or obstacle avoidance.

We developed a short term navigation system that does scene reconstruction, obstacle detection and plans a path using current fields. The scene reconstruction uses a flow algorithm that we developed and which is based on the Lucas and Kanade algorithm. It incorporates a multiresolution scheme to cope with the large displacements and a smoothness factor to stabilize the solution. The accuracy of the algorithm is enough to distinguish small obstacles from the floor. We also introduced the idea of Current (or Dynamic) Fields to

*The support of the NSERC (App. No. OGP0046645) is gratefully acknowledged.

† Currently at SystemWare Inc.(http://www.swi.com)

do the navigation to correct some well known problems associated with the potential fields. We also discovered that many things get simplified if we run these algorithms in image space (retinotpic) rather than actual three dimensional space.

We have also developed a software platform to facilitate experimentation. This platform is modular, so that original components can be interchanged with more advanced ones. It is also portable to survive software or hardware upgrades. Finally, it provides multi-platform support to distribute the workload and take advantage of hardware which is installed on different machines. We named it Field Based Navigator (FBN) after its most important component. Using this software platform we developed navigation algorithms and we tested it on a RWI robot with a video camera as the source of sensor input.

Previous research on autonomous platform using camera image as input includes [4], [12] and [5]. All these examples are based on a wheeled robot platform. Other approaches on the autonomous platform are [3], [15], [11], [6], [13], [10] and [20]. Each kind has its own advantages and limitations because of the physical properties of the platform, the surrounding environment and the usage.

1.1. System Overview

The FBN system contains a moving platform, a video camera and a set of processes. The robot used in this paper is the RWI B12 mobile robot [9, 8]. This robot is equipped with sonar sensors that we do not use and bump switches which are used in the default mode. The RWI platform can perform translation and rotation. The video camera that we use is a simple camera without auto iris or auto focus. It is small enough to fit on the robot (See Picture 1.1).

Communication between the RWI robot and a host computer is via a pair of serial link cables. The cables are connected to the B12 computer and G96 board separately. The power supply is a rechargable 14V battery on board the RWI robot.

The camera used in this paper is a WAT-902A camera by Watec America Corporation. Our experiments are conducted in a room with roughly thirty five square meters ($35m^2$) so that the camera lens has a short focal length and a wide viewing angle. The platform is tethered with a cable containing the camera power, the video output, and the robot control cables.

Figure 1.1: The RWI is a cylindrical robot with $17cm$ radius, $46cm$ in height. The mobile base B12 of the RWI robot has three wheels, rigid suspension, synchronous drive and a forty pound carrying capacity. On top of the mobile base, there are twelve sonars and six bump switches. The B12 computer, which is built around the NEC 78310 micro controller, controls the mobile base and the bump switches. The sonars are controlled by the G96 Sonar Board which contains a Motorola 68HC11 microprocessor. The control language of the B12 computer and the G96 Sonar Board uses two-letter mnemonic commands which are sometimes followed by a hexadecimal number.

The system that we built is not restricted to a single operating system or machine. It works on SGI machines and Sparc stations with two different operating systems. This distributed system is built from portable software components running on different machines and operating systems communicating through machine independent interfaces.

The distributed system consists of four main subsystems: the image processing system, the image acquisition system, the robot command system and the user interface system (Figure 1.2).

All programs in this paper are written in the C programming language or the MediaMath [18] programming language. The C compiler used is the ANSI C compiler. The image acquisition system is based on the IRIS Video Library (VL) in the SGI machines, and the display program used the *xt* library in X Windows System, Version 11. Most of the programs can be executed in more than one operating system such as SunOS 5.5.1, Sun BSD 4.1.3 and SGI IRIS release 6.2. The robot daemon is a system dependent program and it is

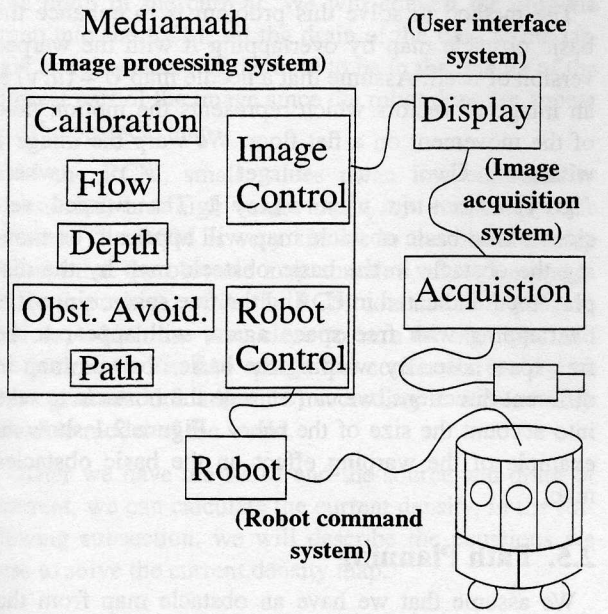


Figure 1.2: The image processing subsystem is the main process in the system which runs in MediaMath [18]. It handles the all computations and communications with other subsystem via sockets. The image acquisition system and the user interface program both work together. The image acquisition daemon captures images from the camera and sends images to the user interface program for the user to observe and for transmission to the image processing system. The robot command system is a process which controls the RWI robot and communicates to the image processing system.

separately implemented in different systems with a common communication interface. Currently, the image acquisition daemon runs only on SGI machines.

In designing the system we strove to achieve robustness and simplicity. We avoided heuristics as much as possible and we tested components with mathematically predictable behavior. Moreover the modularity of the system allows for the incremental development of individual components. Among the most interesting results that came out of this work is that this can be done with stereo/structure from motion despite the noise sensitivity in this kind of computations.

1.2. Overview of the Paper

The structure of the paper is as follows: In section 2, we describe the algorithms in our system. In section 3, we discuss the results of our system and the new navigation algorithm. In section 4, we present a summary of our contribution.

2. Architecture And Algorithms

2.1. Camera Calibration

We used manufacturer's data and assumed a distortionless perspective projection for calibration to keep things as simple as possible. It turns out that although this incomplete calibration introduced errors such as the flat floor was perceived slightly curved by the robot, we did not have any significant problem. The calibration matrix that relates a point on the image coordinate $p_{raw} = (k, l, 1)^T$ with the same point in the camera coordinate system $p_i = (x_i, y_i, 1)^T$ are

$$\mathbf{p}_i = \begin{bmatrix} \alpha_x & 0 & c_x \\ 0 & \alpha_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_{raw_i} \tag{2.1}$$

where parameters α_x, α_y, c_x and c_y are easy to compute from manufacturer's data sheets.

2.2. Flow Estimation

The algorithm we use is based on the modified version from Lucas and Kanade [2]. The modifications include the initial guess of flow, hierarchical computation [14, 1, 19] and the smoothness of the optical flow [7] to make the solution more stable.

The Lucas and Kanade's region matching algorithm is a weighted least-squares fit of local first-order gradient constraint. It assumes a locally constant model for the estimated velocity field $\mathbf{v}$ in small spatial overlapping neighborhoods ω by minimizing

$$\sum_{x \in \omega} W^2(\mathbf{x})[\nabla I(\mathbf{x}, t) \cdot \mathbf{v} + I_t(\mathbf{x}, t)]^2 \tag{2.2}$$

for every neighborhood ω, where $W(\mathbf{x})$ is a Gaussian window function, $I(\mathbf{x}, t)$ is the image intensity and I_t is the partial derivative with respect to time.

Horn and Schunck developed a point matching algorithm which used a smoothness term $\sum(u_x^2 + u_y^2 + v_x^2 + v_y^2)$ where u, v are the two components of the flow and x, y subscripts denote differentiation. The smoothness term is a essential part of the point matching equation to achieve a more stable solution.

Based on Lucas and Kanade's flow algorithm, we implemented a gradient based least-squares region matching multi-resolution flow algorithm using Horn and Schunck's smoothness which leads to a second order differential equation that we solved using the Conjugate Gradient method.

The initial guess of flow is calculated from the estimated depth maps. We estimate the depth map by finding the intersection of the roughly known flat floor and each image point vector p_i.

$$Z_i = \frac{\mathbf{N} \cdot \mathbf{P}_{floor}}{\mathbf{N} \cdot \mathbf{p}_i} \tag{2.3}$$

where Z_i is the z component of the 3-D object point, N is the floor normal and P_{floor} is a point on the floor plane. The initial guess of flow is horizontal and inversely proportional to Z.

2.3. Depth Recovery

The depth map is an image data structure which represents the distance between the camera and the objects at each pixel. This distance is equivalent to Z_i. The calculations are based on [17] and the equation is

$$Z_i = \frac{\left(\mathbf{T} \times \mathbf{p}_i'\right) \cdot \left(\mathbf{p}_i' \times R\mathbf{p}_i\right)}{\|\mathbf{p}_i' \times R\mathbf{p}_i\|^2} \tag{2.4}$$

where the robot moves with rotation R and the translation $\mathbf{T}$, and $p_i' = p_i + \mathbf{u}(p_i)$ where $\mathbf{u} = [u, v]^T$ is the flow.

2.4. Obstacle Detection

The error introduced by the imperfect camera calibration and the irregularities of the floor surface prevents us from using straight subtraction to compare the known floor depth map with the computed depth map to detect the obstacles so instead we compare the estimated floor normal with the image surface normal. First, calculate the actual distance of the image surface from camera ($Z_i \cdot \mathbf{p}_i$) using the depth map, then compute the surface normal from the cross product of the derivative of this image with respect to x with the derivative with respect to y and then normalize. After we have the image surface normal, we calculate the dot product of it and the estimated floor normal. If the dot product is very different from 1, then this indicates a high probability of an obstacle being present. On the other hand a value 1 indicates a high probability of free space. We call this image the *basic obstacle map*.

The basic obstacle map indicates only the location of possible obstacles; it does not consider the physical size of the robot. Since the basic obstacle map is in image coordinates, objects at the bottom of the image are scaled differently than objects at the top. Using any simple enlargement method will result in improper enhancement due to different scales in the image.

The method to solve this problem is to enhance the basic obstacle map by overlapping it with the warped version of itself. Assume that a needle map $\mathbf{U} = (u, v)$ is an image of vectors which represents the motion field of the movement on a flat floor. We warp the image I with $\mathbf{U}$ to get I' where $I'[x, y] = I[x + u[x, y], y + v[x, y]]$. The warped versions of the basic obstacle map will appear to be moving the obstacle in the basic obstacle map by the displacement indicated in $\mathbf{U}$. And the free space, since it is overlapping with free space again, will appear to be free space also. By warping the basic obstacle map in different directions, we can enlarge the obstacle to take into account the size of the robot. Figure 2.1 show an example of the warping effect on the basic obstacles map.

2.5. Path Planning

We assume that we have an obstacle map from the previous steps and that the user specified the target. The method we use to find the path is based on Dynamic Field (DF). We need the RCM (Retinotopic

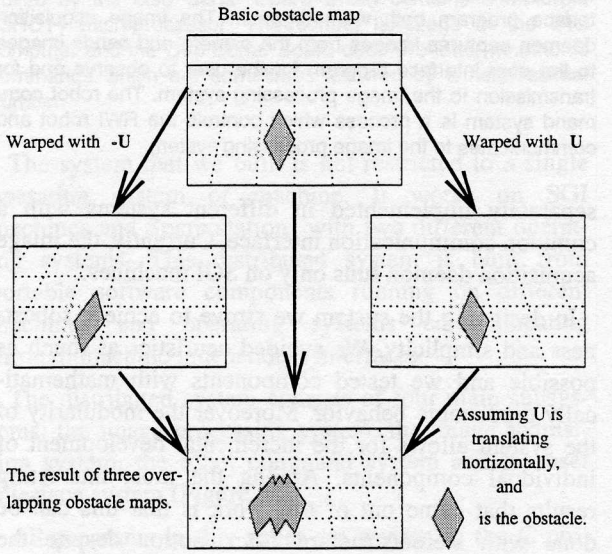


Figure 2.1: The basic obstacle map is warped with sideway to left and right using **U** and -**U**. The closer part of the obstacle is warp father and bigger then the far away part. The combined obstacle is enlarged correctly. With an extra forward warp, the result is taken into account of the size of the circular robot and can be use for path planning.

Conductivity Map) and the location of the source and the drain of the current. We will convert the obstacle map into the RCM, set the drain of the field to the target and the source of the field to be in the middle of the lowest part of the image since the robot does not appear on the obstacle map.

In the RCM, small values mean low conductivity (obstacles) and high values mean low resistance (free space). The RCM is very similar to the obstacle map except that the obstacle map contains values from negative to positive, while the RCM contains only positive values, which are very close to zero on obstacles and very close to 1 in free space. The conversion involves a histogram modification [16] according to a reasonably flexible model of the world.

After we have the RCM and the source and drain of current, we can calculate the current density. In the following subsection, we will describe the equations we use to solve the current density map.

2.5.1. Current Density Equations

Let the conductivity on the RCM be ρ which is a scalar. The potential ϕ is another scalar and the electrostatic field intensity $\mathbf{E}$ and the current density $\mathbf{J}$, where $\mathbf{J} = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}$, are two dimensional vectors. The potential and the intensity are related by

$$\nabla\phi = \mathbf{E} \tag{2.5}$$

and the field intensity and the current density by

$$\mathbf{J} = \rho\mathbf{E}\,. \tag{2.6}$$

If we assume a static field, then the divergence of the density is zero everywhere except at the source and the drain, otherwise charge carriers would accumulate unboundedly.

$$\nabla\mathbf{J} = S \tag{2.7)a}$$

$$\nabla\mathbf{J} - S = 0 \tag{2.7)b}$$

where S is 0 everywhere except at the source and the drain. Also the integral of S over the whole space is 0 so that the current flowing in is the same as the current flowing out.

To guarantee the existence of a potential we impose the constraints:

$$|\nabla\times\mathbf{E}| = 0\,. \tag{2.8}$$

Note that the result of the cross product in equation (2.8) is a vector, but we need only the magnitude of the vector.

From Eq. (2.6) and Eq. (2.8), we have

$$\left|\nabla\times\left(\frac{1}{\rho}\mathbf{J}\right)\right| = 0\,. \tag{2.9}$$

These two independent equations we need to keep are (2.7)b and (2.9). We also introduce a smoothness term $\lambda({J_{1x}}^2 + {J_{1y}}^2 + {J_{2x}}^2 + {J_{2y}}^2)$, where λ is a small coefficient, to stabilize the result. We solve our equations by the least-squares method. The standard procedure to solve the least-square problem is to take the sum of the squares of the equations and the smoothness term

$$\int_{image}(\nabla\mathbf{J} - S)^2 + \left(\nabla\times\left(\frac{1}{\rho}\mathbf{J}\right)\right)^2 + \lambda\left({J_{1x}}^2 + {J_{1y}}^2 + {J_{2x}}^2 + {J_{2y}}^2\right) = 0 \tag{2.10}$$

and differentiate it with each and every one of the unknowns, in our case $J_1[k,l]$ and $J_2[k,l]$ for all k, l. Then we express the equations in a matrix-vector product and use the Conjugate Gradient method to solve the equations. All our functions with spatial arguments are images.

Let $M[i,j] = (\nabla\mathbf{J} - S)[i,j]$ and $N[i,j] = (\nabla\times(\frac{1}{\rho}\mathbf{J}))[i,j]$. The least-square equations for (2.10) are

$$\frac{\partial\sum_{i,j}(M[i,j])^2}{\partial J_1[k,l]} + \frac{\partial\sum_{i,j}(N[i,j])^2}{\partial J_1[k,l]} + \frac{\partial\sum_{i,j}\lambda({J_{1x}}^2[i,j] + {J_{1y}}^2[i,j] + {J_{2x}}^2[i,j] + {J_{2y}}^2[i,j])}{\partial J_1[k,l]} \tag{2.11}$$

and

$$\frac{\partial\sum_{i,j}(M[i,j])^2}{\partial J_2[k,l]} + \frac{\partial\sum_{i,j}(N[i,j])^2}{\partial J_2[k,l]} + \frac{\partial\sum_{i,j}\lambda({J_{1x}}^2[i,j] + {J_{1y}}^2[i,j] + {J_{2x}}^2[i,j] + {J_{2y}}^2[i,j])}{\partial J_2[k,l]}\,. \tag{2.12}$$

By solving the above equations, we have the final equations for the coefficient matrix. (Here the index $[k,l]$ are omitted.)

$$\left(J_{1x}+J_{2y}-S\right)_{x'}+\frac{\left(\left(\frac{J_1}{\rho}\right)_y-\left(\frac{J_2}{\rho}\right)_x\right)_{y'}}{\rho}+$$

$$\lambda\left(J_{1xx'}+J_{1yy'}\right)=0 \qquad (2.13)$$

and

$$\left(J_{1x}+J_{2y}-S\right)_{y'}-\frac{\left(\left(\frac{J_1}{\rho}\right)_y-\left(\frac{J_2}{\rho}\right)_x\right)_{x'}}{\rho}+$$

$$\lambda\left(J_{2xx'}+J_{2yy'}\right)=0\,. \qquad (2.14)$$

where y' is the reflection of y and x' is the reflection of x.

2.5.2. Path Searching

After we have the current density **J**, we can calculate the motion path for our robot. To minimize mechanical error in the robot movement*, the robot will always move in straight lines. The method that we use is an incremental approach. Here the solution will always try to follow the strongest density field from the initial position towards the goal position by a set of straight lines connected with turning points. Note that the density field might cross through obstacles but **J** will be zero or almost zero there mainly due to round off error. So the solution will not contain any path which goes through an obstacle.

The path searching is done in the image pixel coordinates, so before we use the turning points to calculate the moving distance and angle between each line segment, we will need to convert the image coordinates into actual 3-D space coordinates. The coordinate mapping is using the initial guess calculated in the depth recovery step.

The algorithm involves the following steps.

(1) We start the path from the source of the current. This position is conventionally where the robot is located. However, since the robot is not displayed in the image, we will use the next nearest position, which is directly ahead of the robot. We select a position near the lower middle of the image as the source of the current.

(2) From the current position, we find the most suitable next position by choosing among the next three points in the direction of the current density vector. We select among the three by computing the dot product of the previous and the next density vector, then select the biggest value. See Figure 2.2.

(3) After we select the next position, we update the current line segment. If the point is not close to the current line segment, we will mark this point as the turning point, reinitialize the line segment, and then go back to step 2 until we reach the target. We use the standard deviation of all the points on the line segment as the measurement of nearness. If the standard deviation is bigger than a threshold, then the point does not lie near the current line segment.

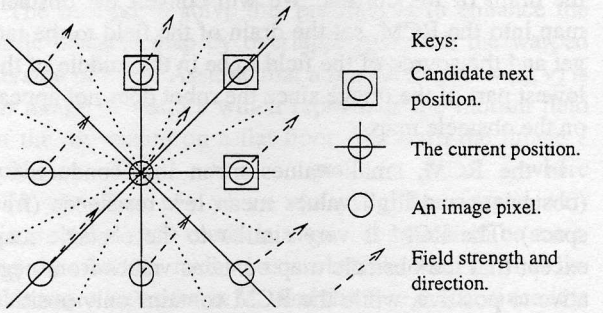


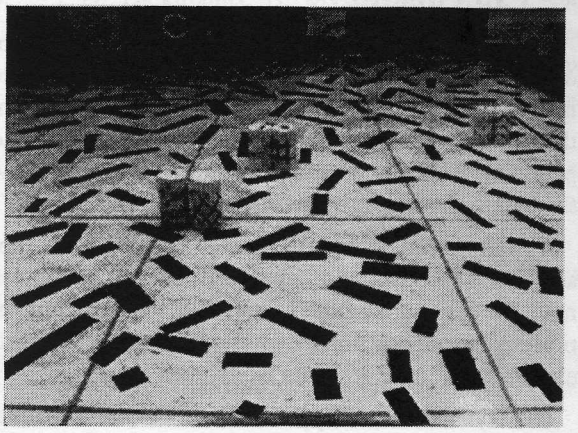
Figure 2.2: The pixel at the centre is the current position. Following the field direction, the next three suitable positions are: the top pixel, the upper-right left hand pixel and the right pixel.

3. Experimental Results

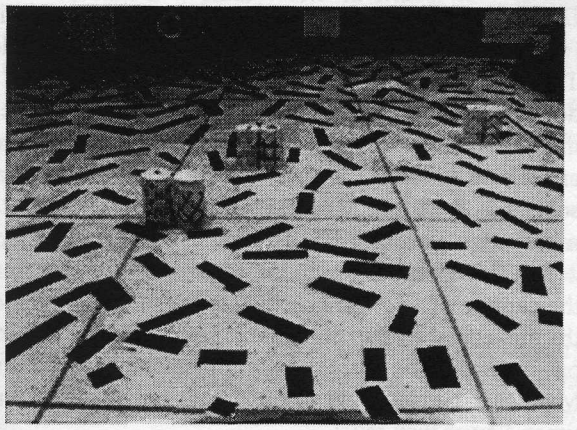
In this section, we will describe a test run with real data. The experiment is using a simple scene with three sets of obstacles.

The FBN system will first capture two video images, we named it left image and right image. Using these two images, we can find the motion flow. The needle map of motion flow and the depth map will be displayed on the FBN user interface window. Then FBN system will calculate the obstacle map and the Retinotopic Conductivity Map (RCM) and display them on the FBN user interface.

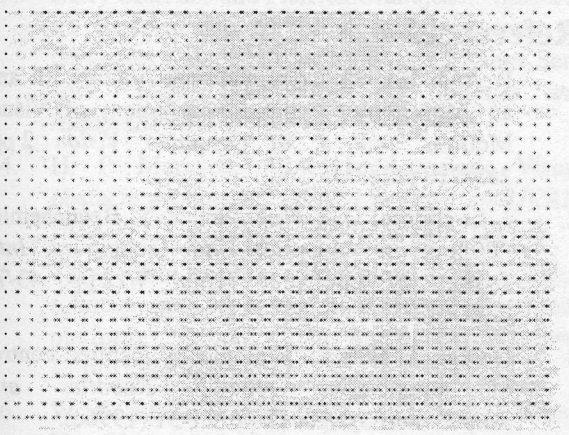
* The uncertainty of socket delays can interfere with the simultaneous translation and rotation ability of the robot.



3.1 Left Image



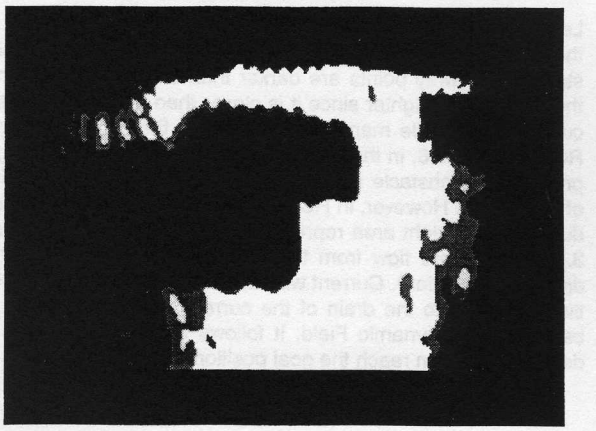
3.2 Right Image



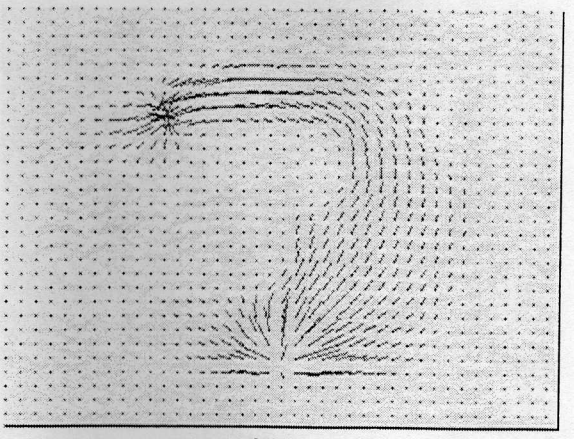
3.3 Motion Flow

3.4 Depth Map

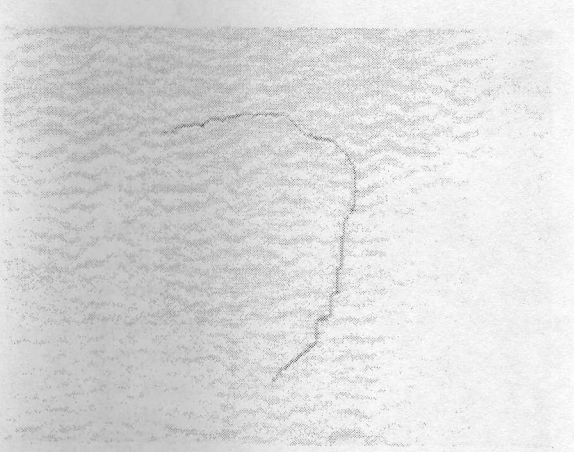
3.5 Obstacle Map



3.6 RCM



3.7 DF



3.8 Motion Path

Legend: Figure 3.1 and 3.2 are the video images. Figure 3.3 is the motion flow. Figure 3.4 shows clearly the location of the obstacles. Far away points are darker than close-by points, and the obstacle is brighter since it is closer than the floor. We calculate the obstacle map Figure (3.5) and then convert it into RCM in figure 3.6. In the obstacle map a dark area means high probability of obstacle and bright area means high probability of free space. However, in RCM, dark area represents low conductivity and bright area represents high conductivity. In figure 3.7, currents are flow from the source (starting point) to the drain (goal position). Current will select area have high conductivity and flow to the drain of the current. The motion path is based on the Dynamic Field, it follows the strongest current density which can reach the goal position.

The FBN system will wait for user input to select the target position. After the user enter the goal position, the FBN system will continue the execution and find the Dynamic Field. Then using the Dynamic Field, we can find the motion path for the moving platform.

4. Conclusion

In this paper, we summarized the experience from development of a modular multi-platform distributed navigation system and a new path planning algorithm that uses dynamic fields instead of the more common potential fields and overcomes some of the problems of the latter without use of heuristics.

A commercial miniature camera provided the input and scene reconstruction algorithms were used to estimate the structure of the scene. The scene reconstruction had enough accuracy to allow us to detect obstacles on the floor as small as whiteboard markers.

The Dynamic Field approach is based on the current field. This approach overcomes most of the problems in the more traditional potential field approaches. The Dynamic Field is computed by solving the partial differential equations. The motion path follows the field. The advantage of the Dynamic Field approach is that it does not have any unsolvable configurations or local minima and maxima and the only cases that it can fail are due to round off errors which further research can substantially reduce. The execution time is roughly 2.5 minutes long in 168 MHz UltraSparc CPU with Solaris 2.5.1. The complete execution of the FBN system from the capture of two camera images to the completion of the movement of the robot is roughly 5 minutes running on an Ultra. The system was implemented on MediaMath that allows rapid development but is about four times slower than raw C.

References

1. P. Anandan, "A Computational Framework and an Algorithm for the Measurement of Visual Motion," *Intl' J. of Computer Vision* **2** pp. 283-310 (1989a).
2. J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of Optical Flow Techniques," *Int'l Journal of Computer Vision* **V.12** pp. 43-77 (1994b).
3. George A. Bekey, "Biologically inspired control of autonomous robots," *Robotics and Autonomous System* **18** pp. 21-31 (1996c).
4. R. Chapuis, A. Potelle, J.L. Brame, and F. Chausse, "Real-Time Vehicle Trajectory Supervision on the Highway," *The International Journal of Robotics Research* **14** pp. 531-542 (1995d).

5. E.D. Dickmanns, B. Mysliwetz, and T. Christians, "An Integrated Spatio-Temporal Approach to Automatic Visual Guidance of Autonomous Vehicles," *IEEE Transaction on System, Man, and Cybernetics* **20(6)** pp. 1273-1990 (1990e).

6. Cynthia Ferrell, "A comparison of three insect-inspired locomotion controllers," *Robotics and Autonomous Systems* **16** pp. 135-159 (1995f).

7. B. K. P. Horn, *Robot Vision,* McGraw-Hill Book Company, New York (1986g).

8. Real World Interface, Inc., *G96 Sonar Board Guide to Operations Version 1.1,* Real World Interface, Inc., Dublin (1990h).

9. Real World Interface, Inc., *B12 Base Manual Version 2.4,* Real World Interface, Inc., Dublin (1994i).

10. D. Langer, J.K. Rosenblatt, and M. Hebert, "A Behavior-Based System for Off-Road Navigation," *IEEE Transactions on Robotics and Automation* **10(6)** pp. 776-783 (1994j).

11. D. B. Marco, A. J. Healey, and R. B. McGhee, "Autonomous Underwater Vehicles: Hybrid Control of Mission and Motion," *Autonomous Robots* **3** pp. 169-186 (1996k).

12. Larry Matthies, "Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-Time Implementation," *International Journal of Computer Vision* **8**(1) pp. 71-91 (1992l).

13. Larry Matthies and Pierrick Grandjean, "Stochestic Performance Modeling and Evaluation of Obstacles Detectability with Imaging Range Sensors," *IEEE Transactions on Robotics and Automation* **10**(6) pp. 783-792 (1994m).

14. H. P. Moravec, "Towards Automatic Visual obstacle avoidance," pp. 584 in *Proc. 5th IJCAI.*, IJCAI-77, Cambridge, Massachusetts (1977n).

15. Takao Okui and Yoshiaki Shinoda, "An outdoor robots system for autonomous mobile all-purpose platform," *Robotics and Autonomous Systems* **17** pp. 99-106 (1996o).

16. W. K. Pratt, *Digital Image Processing,* Wiley, New York (1991p).

17. M. E. Spetsakis and J. Aloimonos, "Optimal Estimation of Structure from Motion from Point Correspondences in Two Frames," in *Proc. ICCV*, , Tampa, Florida (1988q).

18. Minas Spetsakis, "MediaMath: A reasearch environment for vision research," pp. 118-126 in *Vision Interface*, , Banf, Alberta (1994r).

19. J. Weber and J. Malik, "Robust computation of optical flow in a multi-scale differential framework," *ICCV*, pp. 231-236 (1993s).

20. C. R. Weisbin, Mel Montemerlo, and W. Whittaker, "Evolving directions in NASA's planetary rover requirements and technology," *Robotics and Autonomous Systems* **11** pp. 3-11 (1993t).