Proceedings, SPIE Real-Time Image Processing II: Algorithms, Architectures, and Applications, Vol. 1295, Orlando, Florida, April 1990.

Real-Time Differential Range Estimation Based on Time-Space Imagery Using PIPE

Ramesh Rangachar[1], Tsai-Hong Hong[1,2], Martin Herman[1], Randall Luck[3] and Jasper Lupo[4]

[1]National Institute of Standards and Technology (NIST)
Bldg. 220, Rm B124, Gaithersburg, MD 20899

[2]Department of Computer Science and Information Systems
The American University, Washington, D.C. 20899

[3]Aspex Incorporated, 530 Broadway, New York, NY 10012

[4]Defense Advanced Research Projects Agency (DARPA)
1400 Wilson Blvd., Arlington, VA 22209

ABSTRACT

Image flow, the apparent motion of brightness patterns on the image plane, can provide important visual information such as distance, shape, surface orientation, and boundaries. It can be determined by either feature tracking or spatio-temporal analysis. We consider spatio-temporal methods, and show how differential range can be estimated from time-space imagery.

We generate a time-space image by considering only one scan line of the image obtained from a camera moving in the horizontal direction at each time interval. At the next instant of time, we shift the previous line up by one pixel, and obtain another line from the image. We continue the procedure to obtain a time-space image, where each horizontal line represents the spatial relationship of the pixels, and each vertical line the temporal relationship.

Each feature along the horizontal scan line generates an edge in the time-space image, the slope of which depends upon the distance of the feature from the camera. We apply two mutually perpendicular edge operators to the time-space image, and determine the slope of each edge. We show that this corresponds to optical flow. We use the result to obtain the differential range, and show how this can be implemented on the Pipelined Image Processing Engine (PIPE). We discuss several kinds of edge operators, show how using the zero crossings reduces the noise, and demonstrate how better discrimination can be achieved by knowledge of range.

1. INTRODUCTION

Image understanding is concerned with the determination of visual information such as distance, shape, surface orientation, boundaries, etc. One method of determining the visual information using monocular vision is to use optical flow, which is defined as the apparent motion of brightness patterns in the image plane. There are two methods to find the optical flow.

1. Feature matching based method. In this method, identifiable features from a sequence of images are extracted and correspondence is established. The corresponding features are used to calculate a set of disparity vectors for the sequence. Any identifiable entity can be used as a feature, but sharp, localized features give the best accuracy.

2. Spatio-temporal methods. These methods use the spatial and temporal relationships of image intensities to determine the optical flow. The time history of the position of a feature in the image can be represented in a 3-dimensional coordinate system (x,y,t). This is often called a spatio-temporal solid or spatio-temporal volume. The spatio-temporal area is defined as an $x-t$ slice of the solid, i.e., a slice taken with $y = constant$. The slope at any point (x,t) in the spatio-temporal area gives the horizontal component of its velocity. Therefore, the horizontal component of optical flow can be obtained by determining the orientation of the edges.

In section 2, we define optical flow, and show the equation for optical flow using gradient methods. In section 3, we give a brief introduction to edge operators, and show how the magnitude and direction of edges can be extracted using edge operators. In section 4, we first define an Epipolar Plane Image (EPI), and show how distance of objects from the camera is encoded in the EPI. The results of sections 2 and 3 are used to show how optical flow can be

extracted from the spatio-temporal image. Many authors consider the *gradient method* of determining optical flow as a separate method. Contrary to this classification, we show that gradient methods and spatio-temporal methods are the same. The algorithm and its implementation on PIPE are given in section 5. We describe our experimental setup and the experiments in section 6.

2. OPTICAL FLOW AND ITS ESTIMATION

The equation for optical flow using the gradient based method can be written as

$$I_x u + I_y v + I_t = 0 \tag{1}$$

where u and v are the components of flow velocity in the x and y directions respectively, I_x and I_y are the spatial gradients, and I_t is the temporal gradient.[4] Let us define two types of motion: *local motion*, which is the motion of the objects in the environment, and *global motion*, which is the motion of the camera itself. If we assume that the local motion is zero and that the global motion is only along the x direction, the component of motion in the y direction is zero. Under these assumptions, equation (1) becomes

$$I_x u + I_t = 0$$

The magnitude of velocity in the x direction is given by

$$u = \frac{|I_t|}{|I_x|} \tag{2}$$

Equation (2) holds for a camera moving parallel to the image plane along the x direction. If the velocity V of the camera and its focal length f are known, the depth d of an object from the camera can be obtained using the following equation.

$$d = \frac{-fV}{u} \tag{3}$$

If there are two objects O_1 and O_2 at depths d_1 and d_2, respectively, from the camera, with $d_1 > d_2$, and they induce optical flow values u_1 and u_2, then we can derive the following from Equation (3):

$$\frac{d_1 - d_2}{d_1} = \frac{u_2 - u_1}{u_2} \tag{4}$$

The reasons why we calculate this function are

(i) Equation (4) is independent of V and f so that calibration of the camera is not required, and

(ii) we would like to determine the minimum distance between two objects that can be discriminated using optical flow.

We define Equation (4) as the differential range.

3. EDGE DETECTION

An edge is a geometric form whose gray level is consistent on each of the two adjacent, extensive regions and changes abruptly as the border between the regions is crossed.[14] The pixel locations where this abrupt gray-level change occurs are called edge points. The derivative of a continuous image $I(x,y)$ has a local maximum in the direction perpendicular to the edge. The magnitude M and the direction Φ are given by the following equations.[5]

$$M = \left\{ \left[I_x\right]^2 + \left[I_y\right]^2 \right\}^{\frac{1}{2}} \tag{5}$$

$$\Phi = \tan^{-1}\left[\frac{I_y}{I_x}\right] \tag{6}$$

There are several different methods by which edges can be detected, and research is being continued to determine alternative methods. In this work, which is an application of edge operators to estimate optical flow, we have used gradient operators. The gradient operators have two mutually perpendicular masks H_x and H_y which are numerical

RHINEHART

approximations to I_x and I_y, respectively, and are used to determine M and Φ using equations (5) and (6). These operators give high signal to noise values at the edge points and zeroes for uniform regions. There are several gradient operators such as Prewitt, Sobel, isotropic, stochastic, etc. All of these can be easily implemented on digital hardware.

4. SPATIO-TEMPORAL IMAGE ANALYSIS

As discussed above, the time history of each feature point in the image gives the spatio-temporal volume, and a slice in the temporal direction gives the spatio-temporal area. Figure 1(a) shows a scene containing a railroad car and a tree. Assume that the camera moves in the horizontal direction, that the optical flow axis points perpendicular to the direction of motion, and the optical flow is in the negative direction. A horizontal scan line j of the image is selected and the spatio-temporal area for this scan line is obtained. When the scan line is coincident with the direction of motion of the camera, the spatio-temporal area is called the Epipolar Plane Image (EPI).[2] The EPI of the line j is shown in Figure 1(b). The slope due to the tree (closer object) is smaller than the slope due to the railroad car. Thus, depending upon the distance of the objects falling on the scan line j, there will be lines of different slopes in the EPI. If the slope can be measured, the distance between the camera and the object can be calculated.

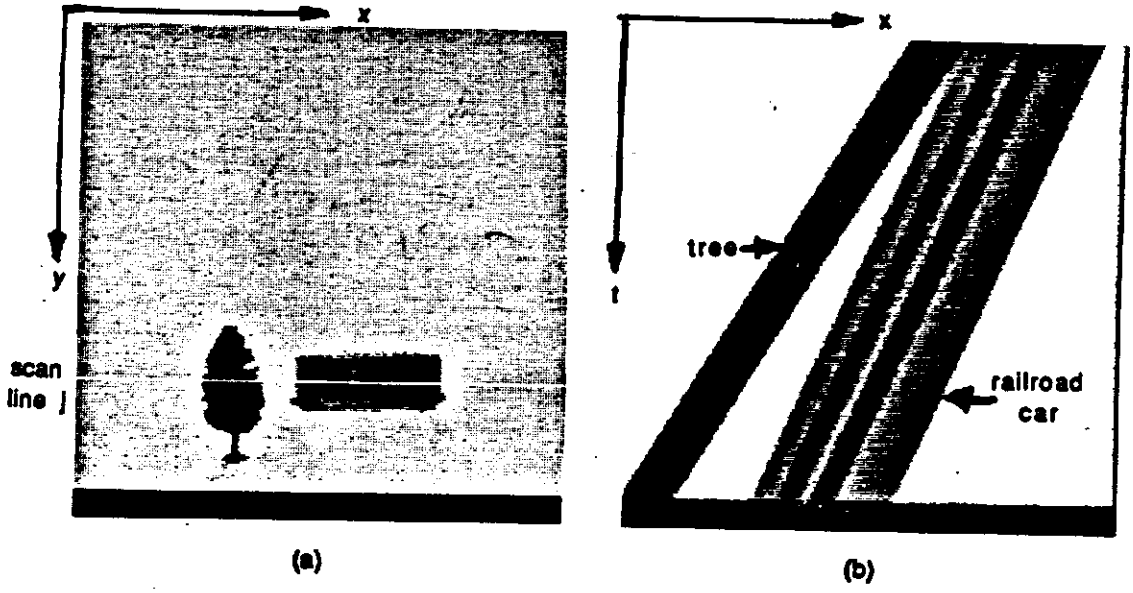


Figure 1: (a) Image at time k; (b) EPI for scan line j.

The use of time-space imagery for determining optical flow was first introduced by Bolles and Baker.[2] They determined slope by using a feature matching method. Here we take a different approach and use edge operators to determine the slope.

In section 2, we showed how optical flow can be obtained using the ratio of temporal difference over spatial difference (Equation 2). In an EPI, the horizontal axis represents space while the vertical axis represents time. Applying H_x to an EPI is similar to finding the spatial difference, while the application of H_y gives the temporal difference. If equation (6) is modified to determine the slope of a line in an EPI, and the variable y is changed to t, it can be written as follows:

$$\tan(\Phi) = \left[\frac{I_t}{I_x}\right] \tag{7}$$

Comparing equations (2) and (7), one can see that the two are similar and represent the same quantity. We use Equation (7) to determine optical flow in the EPI.

In section 3, it was stated that the gradient operators give high values at the edge points and zeroes for uniform regions. Therefore, optical flow will have high values at the edges and becomes indeterminate for uniform regions. The signal-to-noise of the flow is maximum at the edge and decreases as one moves away from it until it becomes

indeterminate. To find a high signal-to-noise value for flow, the flow value at the edge must be extracted, while the flow values around the edge must be eliminated. We do this by determining the zero crossings of the Laplacian-Gaussian of the image and returning flow only at these zero crossing points.

We confine ourselves to small values of optical flow (less than 2 pixels per frame-time) because flow values significantly greater than zero are known to induce large measurement errors due to under-sampling.[6] We have implemented our algorithm to compute the optical flow on PIPE, which processes images at video rate.

5. PIPE IMPLEMENTATION

PIPE is a multi-stage, multi-pipelined image processing machine which may be used as the front-end of a real-time image understanding system. It was designed specifically for low level vision tasks at very high speed. It was conceived and designed at the National Institute of Standards and Technology (formerly National Bureau of Standards) by Kent, et al.[7] and is commercially available through Aspex Incorporated.[1] PIPE has a video stage to receive imagery, an input stage with four buffers, 1 to 8 identical Modular Processing Stages (MPSs) to process the imagery, and an output stage with four buffers to store images.

For further details on PIPE, refer to the references listed in this paper.[3,7-10,12,15]

The algorithm to determine differential range of a selected scanline of an image has the following steps.

1. Obtain the EPI.
2. Determine the best estimate of optical flow.
3. Segment the optical flow.

1. Obtain the EPI: We first create an EPI similar to Figure 1(*b*) in real time. Here we describe how the EPI for any selected scan line is obtained. For details on PIPE implementation, refer to the Appendix.

Figure 2 shows the steps to obtain the EPI for a selected scan line. The desired scan line is selected interactively using a program on the host computer. Figure 2(a) shows the image of an object and the selected scan line j at time t_1. Figure 2(b) shows only the scan line. This scan line is spread to the entire image as shown in Figure 2(c) using image warping capability of PIPE. An image with a mask line at the bottom is ANDed with the spread image to shift the scan line to the bottom as shown in Figure 2(d). The steps explained above are repeated for another scan line at the next instant of time t_2. Figures 2(e) through (h) show the results at time t_2, where the image is shifted slightly to the left due to the horizontal motion of the camera. To obtain the EPI, the image at time t_1 (Figure 2d) is shifted up and added to the image at time t_2 (Figure 2h). Figure 2(k) shows the result at time t_2. The process is repeated for subsequent scan lines. Since we are using 256×242 images, it takes 242 cycles or 8 seconds to obtain the first EPI, but subsequent EPIs are formed in real-time by positioning the most recent line at the bottom of the EPI (Y=242) and shifting the previous scan lines up by one row.

2. Obtaining the best estimate of optical flow: This step begins with the previously obtained EPI. Figure 3 shows the data flow graph for this process. As indicated in the left branch of Figure 3, the EPI passes through two Neighborhood Operators (NOPs), where it is convolved with horizontal and vertical edge operators GRAD_X and GRAD_Y respectively. The horizontal operator gives the spatial difference while the vertical operator gives the temporal difference. The outputs are then directed to a Two Valued Function (TVF) look up table called SLOPE which determines the slope, and hence the optical flow using Equation (7).

As mentioned earlier, in order to obtain the best estimate of optical flow, the zero crossings of the Laplacian-Gaussian of the image are used as a mask to determine where optical flow is returned. The right branch in Figure 3 shows how the zero crossings are determined. The EPI is passed through the difference of two Gaussians (which is equivalent to a Laplacian-Gaussian operator). The output is passed through a boolean neighborhood operator where the zero crossings are determined. The zero crossings are then ANDed with the optical flow image from the TVF to obtain the best estimate of the optical flow.

3. Segmentation of optical flow: It has been shown (Equation (3)) that optical flow is inversely proportional to the depth. Therefore, segmenting the optical flow is equivalent to segmenting depth. This is performed by using an interactive program from the host computer.

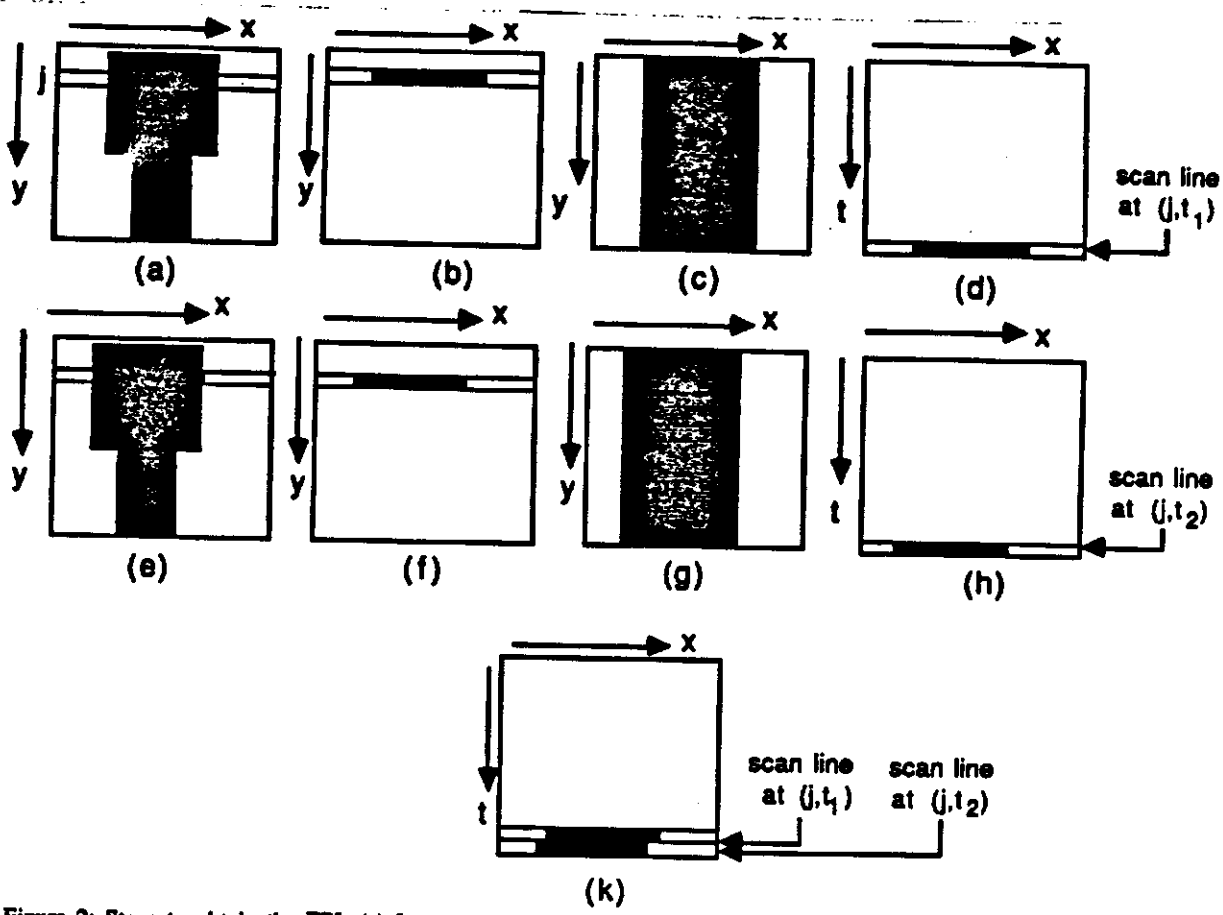


Figure 2: Steps to obtain the EPI. (a) Image of an object and the selected scan line at time t_1; (b) selected scan line at time t_1; (c) spread image of (b); (d) scan line at time t_1 shifted to bottom of EPI; (e), (f), (g) and (h) similar results at time t_2; (k) image (d) shifted up and added to (h).

FOR each pixel

IF (*flow* < *threshold*) *THEN intensity* = 0

ELSE intensity = 255

Consider two objects O_1 and O_2 at depths d_1 and d_2, respectively, with $(d_1 > d_2)$, from the camera. These give rise to optical flow values u_1 and u_2 $(u_1 < u_2)$, respectively. We interactively vary the threshold in the program shown above until the flow due to the object at depth d_1 disappears from the scene while the flow due to the object at distance d_2 remains. We thus say that we have discriminated between objects O_1 and O_2. By physically measuring the values d_1 and d_2, we can calculate the *range discriminability* r_d using the following equation obtained from Equation (4).

$$r_d(\%) = \frac{(d_1 - d_2)}{d_1} \times 100 \tag{8}$$

6. EXPERIMENTS, RESULTS AND DISCUSSION

A top view of the setup for our experiments is shown in Figure 4. We use an optical that rail has two platforms as shown in the figure: one translates horizontally and the other rotates about a vertical axis. The camera is mounted on the platforms as shown. The accuracy of the velocity of translation is within 0.02% of the selected velocity. Objects (O_1, O_2) were placed at different depths from the camera (d_1, d_2) and the optical flow was determined. Segmentation was performed on the optical flow and the range differential was calculated. The experiment was repeated by decreasing the difference $d_1 - d_2$ until further segmentation was not possible.

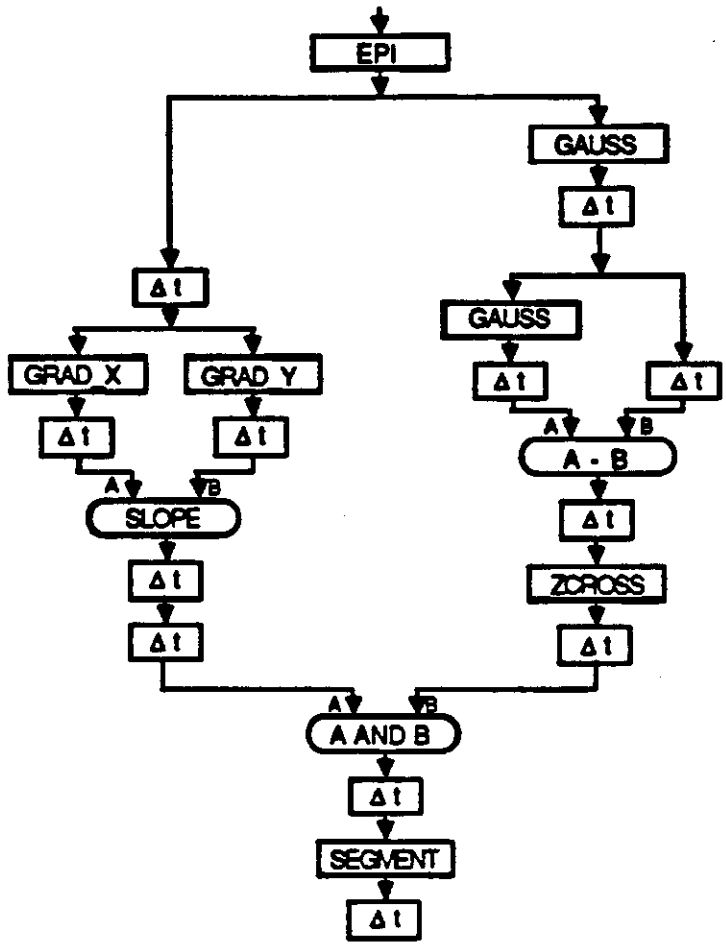


Figure 3: Data flow graph to determine the slope.

We experimented with 3×3, 5×5, and 7×7 edge operators. Since PIPE performs only 3×3 neighborhood operations, we first used 3×3 edge operators. The optical flow was thresholded to determine the discriminability of the method. The range discriminability calculated using Equation (8) was 18%. The experiment was repeated using 5×5 and 7×7 edge operators. 5×5 and 7×7 edge operators were approximated by products of 3×3 convolutions.[11]

To compare the different methods, we determined *the number of misclassified points* (N_{miss}) for each method. N_{miss} was determined as follows.

Two objects O_1 and O_2 were placed at depths d_1 and d_2, respectively, with $(d_1 > d_2)$, from the camera as shown in Figure 4. The threshold for making the optical flow due to object O_1 disappear was determined. Knowing this, a three-level segmentation was performed as follows.

FOR each pixel

IF $(flow < noise\text{-}level)$ *THEN* $intensity = 0$

IF $(flow < threshold)$ *THEN* $intensity = 128$

ELSE $intensity = 255$

This program segments an image in such a way that the image will have an intensity value of 128 for the background and 255 for the foreground. To determine N_{miss}, a 5×5 window was used on the entire image. In each window, the number of points classified as foreground and the number of points classified as background were determined. The

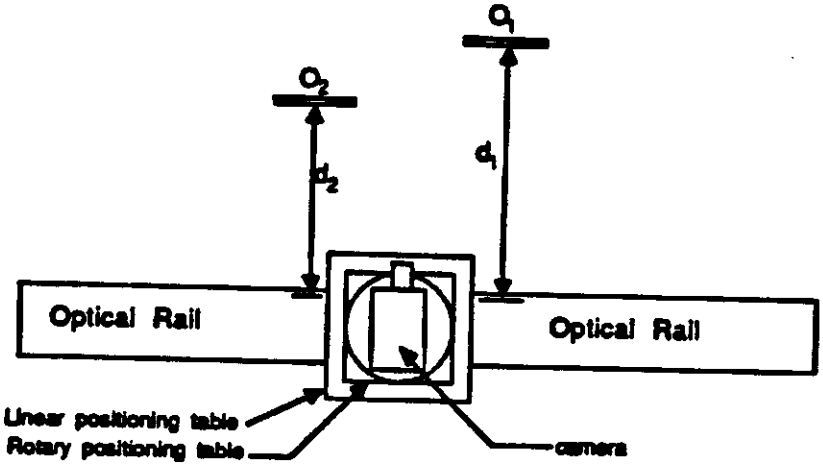
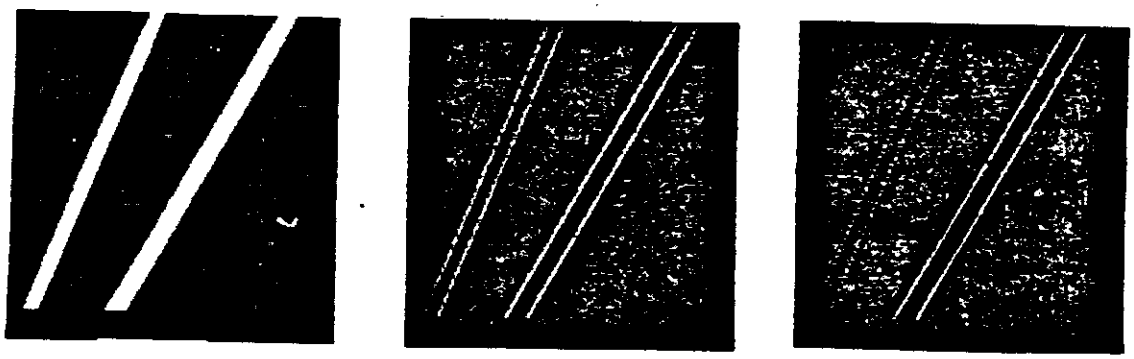


Figure 4: Experimental setup.

one with a lower value was considered misclassified. N_{misc} was calculated using the following equation.

$$N_{misc} = \frac{total\ number\ of\ misclassified\ points}{total\ number\ of\ points\ in\ the\ image} \tag{9}$$

N_{misc} was 7.1% when the 3×3 operator was used. When the 5×5 operator was used, N_{misc} reduced significantly to 3.6%. When the 7×7 operator was used, N_{misc} was 2.9%. The reduction in N_{misc} was not significant in the latter case, because the approximate convolution had a small error. However, the experiment proved that the results can be improved by using convolutions of higher order.



(a) (b) (c)

Figure 5: Experimental results showing segmentation. (a) EPI; (b) optical flow for (a); (c) segmented optical flow.

Figure 5(a) shows an EPI for two objects at different distances from the camera. The optical flow determined using the 3×3 edge operator for this EPI is shown in Figure 5(b). Here each pixel shows the magnitude of optical flow at the point. Figure 5(c) shows how the optical flow can be thresholded into three levels, 0, 128 and 255 to calculate N_{misc}. Since outputs for other edge operators are similar, they are not shown here.

If apriori knowledge of the environment is available, the PIPE implementation of the optical flow equation can be modified to improve range discriminability. Let a more distant object O_f be at a distance d_f from the camera, and let the optical flow corresponding to this object be u_f. Let d_n and u_n be the corresponding values for the nearer object O_n.

Knowing u_f and u_n, the output optical flow can be scaled within the two-valued function look-up table in PIPE to obtain maximum discriminability. For example, suppose a table at a distance d_f from the camera has objects of various thicknesses lying on it. If the maximum thickness of the objects is known, then the system can be tuned properly to determine the thickness of objects with maximum possible efficiency. A range discriminability of 12% has been achieved using this technique.

The results can be improved further by carrying out temporal integration of the optical flow. The flow values are integrated over time to obtain an average value. It has been shown by many researchers that this technique has the potential to improve the accuracy of the method to 0.5%. The authors are working on a real-time implementation of this algorithm on PIPE.

In our experiments, the EPI was obtained using a single line from a sequence of 242 consecutive images. Since our analysis technique involves only local edge operators, the EPI analysis can be implemented on a full image if only the most recent few scan lines of each EPI are held in memory.

The EPI method is also useful for experiments in camera fixation. A fixation point is a point in 3-D space that projects to a single point in an image over some period of time while the camera is undergoing both translation and rotation. The optical flow around the fixated point changes as the camera moves, and the EPI can be used to analyze this.[13]

7. CONCLUSION

Depth from motion can be obtained in a variety of ways, using different techniques. EPI analysis can be used to determine optical flow using edge operators. The algorithm has been implemented on PIPE and results have been obtained in real time. The accuracy of the method is dependent on the size of the edge operator used, but real time implementation on PIPE becomes difficult as the size of the operator increases. The optical flow determined will have high values near the edge and becomes indeterminate for uniform regions, with the best estimate being at the center of the edge. Zero crossings of the Laplacian-Gaussian of the image have been used to localize these edge points. The discriminating efficiency of the method can be increased by using our knowledge about the environment. The EPI technique is an effective tool for research in optical flow, fixation, and other applications such as robot guidance.

ACKNOWLEDGEMENTS

This work was supported by the Defense Advanced Research Project Agency (DARPA), Tactical Technology Office. We thank Don Orser for his valuable comments and suggestions.

REFERENCES

1. *The PIPE User's Manual*, Aspex, 1987. 530 Broadway, New York, NY 10012
2. Bolles, R.C. and Baker, H.H., "Epipolar-Plane Image Analysis: A Technique for Analyzing Motion Sequences," *Proc. DARPA Image Understanding Workshop*, pp. 137-148, Miami Beach, Florida, Dec. 1985.
3. Herman, M., "Application of the PIPE Image Processing Machine to Scanning Microscopy," *Proc. SPIE Scanning Microscopy Technologies and Applications*, vol. 897, pp. 169-173, Los Angeles, CA., January 1988.
4. Horn, B.K.P. and Schunk, B.G., "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.
5. Jain, A. K., *Fundamentals of Digital Image Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersy, 1989.
6. Kearney, J.K., Thompson, W.B., and Boley, D.L., "Optical Flow Estimation: An Error Analysis of Gradient-Based Methods with Local Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 2, pp. 229-244, March 1987.
7. Kent, E.W., Shneier, M.O., and Lumia, R., "PIPE (Pipelined Image Processing Engine)," *J. Parallel and Distributed Computing*, pp. 50-78, 1985.
8. Luck, R.L., "An overview of the PIPE Systems," *Proc. Third Intl. Conference on Super Computing*, vol. 3, pp. 69-78, Boston, 1988.
9. Luck, R.L., "PIPE: a Parallel Processor for Dynamic Image Processing," in *Image Understanding and Man-Machine Interface*, ed. Pearson, J.J. Barett, E., pp. 109-115 , Proc. SPIE 758, 1987.

RANGEMAP

10. Nashman, M. and Chaconas, K., *Low Level Image Processing Techniques Using the Pipelined Image Processing Engine in the Flight Telerobotic Servicer*, Robot Systems Division, National Institute of Standards and Technology, 1989.

11. O'Leary, D.P., "Some Algorithms for Approximating Convolutions," *Computer Vision, Graphics, and Image Processing*, no. 41, pp. 333-345, 1988.

12. Rangachar, R., Hong, T.-H., Herman, M., and Lupo, J., "Real-Time Implementation of a Differential Range Finder," *Proc. SPIE Real-Time Image Processing II: Algorithms, Architectures, and Applications*, Orlando, Florida, April 1990.

13. Raviv, D. and Herman, M., "Towards an Understanding of Camera Fixation," NISTIR 89-4217, Robot Systems Division, Sensory Intelligence Group, National Institute of Standards and Technology, Gaithersburg, MD 20899, December 1989.

14. Rosenfeld, A. and Kak, A.C., *Digital Picture Processing, Vol 2.*, Academic Press, Inc., Orlando, Florida , 1982.

15. Singh, A., "Image Processing on PIPE," Technical Report TN-87-093, Philips Laboratories, Briarcliff Manor, New York, 1987.

APPENDIX

Here we give the details on PIPE implementation of our algorithm to obtain the EPI. In the following PIPE implementation, we consider a 256×242 image. An image called *Y pixel coordinate* (YPC) is defined as an image where the intensity value of each pixel in some particular row is equal to the row number. Thus, each pixel in the first row has a value of 1 while each pixel in the 242nd row has a value of 242. Similarly, the *X pixel coordinate* (XPC) image has an intensity value of 1 in the first column and a value of 256 in the 256th column. Since PIPE is an 8 bit machine, we consider 8 bit planes, numbered 0 to 7. A value of 1 in all bit planes is equivalent to an intensity of 255.

The first step in obtaining the EPI is to select the desired scan line for which the EPI is to be created. To obtain the scan line, we generate a horizontal mask line at the desired horizontal row and mask it on the input image. Figure 6(a) shows the data flow graph to obtain the mask line.

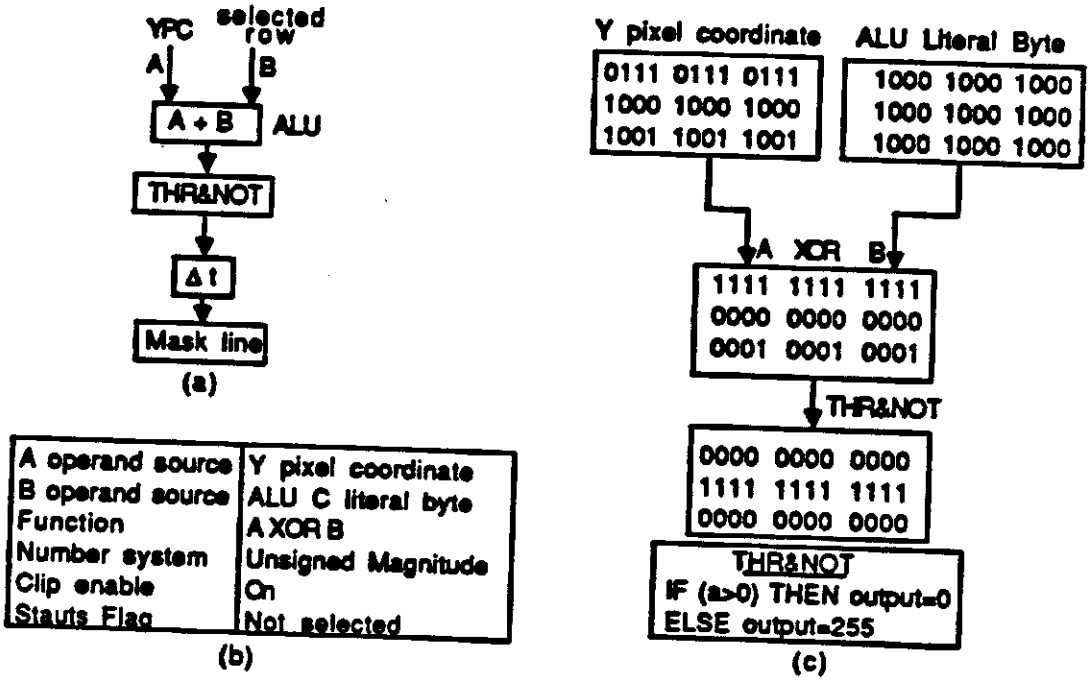


Figure 6: (a) Data flow graph to generate a mask line; (b) inputs to the ALU shown in (a); (c) computations near the selected row (row 80).

First a 256×242 image called *selected_row* is created such that the intensity value of every pixel in this image is equal to the row number of the scan line to be selected (say for example, row 80). A *YPC* image and the image *selected_row* form the *A* and *B* operands for the Arithmetic Logic Unit (ALU)of PIPE. These two are XORed *bitwise* in the ALU. The inputs to the ALU are shown in Figure 6(b). In the output image, only the row with $Y = 80$ will have zero in all bit planes. The result is passed through a Look Up Table (LUT) which negates the values, with the result that only the row with $Y=80$ will have 1 in all bit planes, or a total value of 255 since PIPE is an 8 bit machine. In the same LUT, the values less than 255 are set to zero, so that only the desired mask line has 1 in all bit planes. When this is masked on an input image, only the line corresponding to $Y = 80$ is selected. Figure 6(c) shows how the pixel values change near $Y = 80$ (only the first 4 bits are shown).

Figure 7 shows the data flow graph to obtain the EPI. The mask line is ANDed with the input image to obtain the scan line.

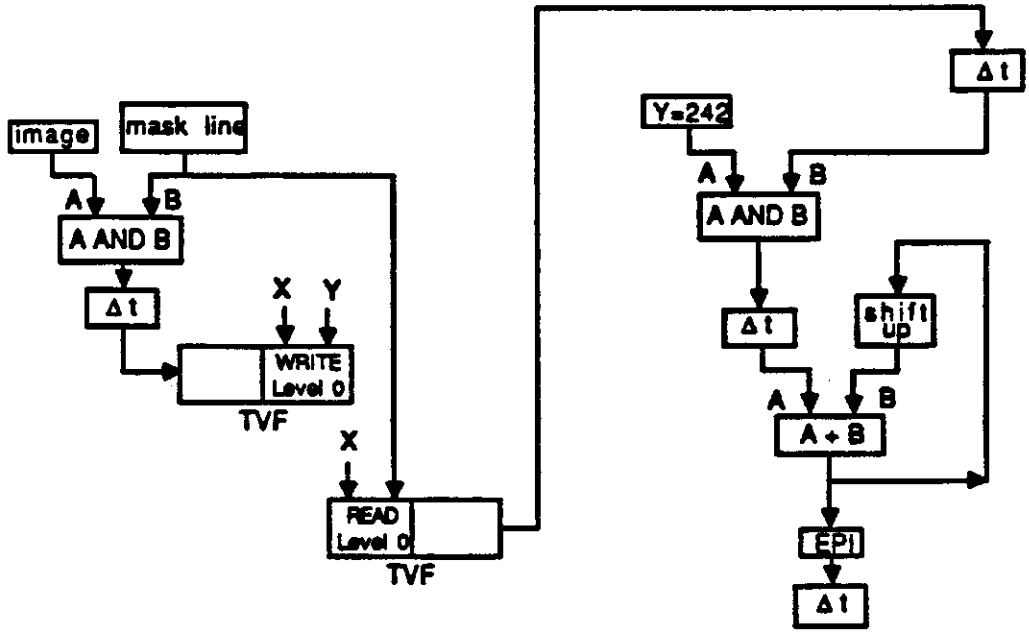


Figure 7: Data flow graph to obtain the EPI

The selected scan line is input to a *Two Value Function (TVF)* of PIPE. This is done by *writing into* the TVF the intensity values of the scan line image. The address of each pixel, i.e., its *X* and *Y* pixel coordinates, are used to write the intensity values to the TVF. While reading the output from the TVF, the Y pixel coordinates are replaced by the mask line as shown in Figure 7. As a result, all the rows in the image will have the scan line. Another horizontal mask line is created at $Y=242$ and masked on the previous result. The scanline is shifted to the bottom of the EPI being created. At the next instant of time, the present scan line is shifted up, and the new scan line stored at $Y=242$. The procedure is repeated until the EPI is obtained.