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

Real-Time Implementation of a Differential Range Finder

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

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

ABSTRACT

Optical flow provides an effective measure of important visual information such as distance, shape, surface orientation, and boundaries. A simple and elegant method to determine optical flow is the gradient method, where the relationship between the spatial difference and the temporal difference is used to estimate optical flow.

As the number of equations that constrain the problem is one less than the number of unknowns, this problem is considered ill-posed and many assumptions have been made by previous authors to regularize the problem. Our method makes the assumption that the direction of camera motion, and hence the direction of optical flow, is known.

Many applications such as target acquisition, path planning, passive navigation, and landing on unknown terrain require range estimation. In this work, a scheme to estimate differential range using optical flow along a known direction is described. The factors affecting the accuracy of results, and various spatial and temporal smoothing algorithms used to increase the accuracy of the method are described. The effect of using edge detectors and apriori knowledge of the environment is considered next. While the former reduces the noise, the latter improves the range discriminability of the method.

We have implemented the method on a real-time, high-speed, Pipelined Image Processing Engine (PIPE), which processes sixty image frames per second. For the PIPE implementation, a horizontally moving camera is used, which produces optical flow along a scan line.

1. INTRODUCTION

Biological vision systems are so powerful that researchers, fascinated by it, are trying to develop artificial vision systems. Significant progress has been made since the advent of digital computers. Several approaches are being followed to obtain useful data from artificial vision systems. One such technique is to process a sequence of images to obtain optical flow, the motion of brightness patterns in the image. This has the potential to provide visual information such as distance, shape, surface orientation, and boundaries, which are useful in applications like target acquisition, path planning, passive navigation, etc.

The key to determining surface shape, orientation, etc. using optical flow is to determine the distance of the objects from the camera from optical flow. In this work, we determine the differential range using optical flow computed in real time. In section 2, we define differential range and optical flow and state some assumptions that can be used to mathematically model the optical flow. We show that the problem is ill-posed. We also show how the problem can be regularized by making an assumption about the direction of motion. In section 3, we give a brief description of Pipelined Image Processing Engine (PIPE), a parallel processing computer that processes images at video rate. Section 4 shows how the algorithm for optical flow can be implemented on PIPE. Results and discussion are given in section 5, where we describe several experiments. The conclusion summarizes the major ideas presented in this paper.

2. MOTION FIELD AND OPTICAL FLOW

When the image of an object is formed on the image plane of the camera, every point P_o on the object results in a corresponding point P_i in the image. If P_o moves with a velocity v_o in the real world, then P_i moves with a velocity v_i in the image plane. Thus, all points on the object have a corresponding velocity vector in the image. These velocity vectors constitute the motion field. As the object moves, the brightness pattern in the image plane also moves. The apparent motion of the brightness pattern is known as optical flow. In an ideal situation, this corresponds to the motion field.[6]

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 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.

$$d = \frac{-fV}{u} \tag{1}$$

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 (1):

$$\frac{d_1 - d_2}{d_1} = \frac{u_2 - u_1}{u_2} \tag{2}$$

We define Equation (2) as the differential range. The reasons why we calculate this function are

(i) Equation (2) 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.

The optical flow can be mathematically modelled and used as an estimate of the motion field, but this requires certain assumptions which are listed below.

1. The apparent velocity of brightness patterns can be directly identified with the movement of the surfaces in the scene.
2. The surface being imaged is flat, and thus there is no variation in brightness due to shading effects.
3. Incidental illumination is uniform across the surface.

Under these assumptions, the brightness at a point in the image is proportional to the reflectance of the surface at the corresponding point on the object. As the reflectance varies smoothly and has no discontinuities, the image brightness is also continuous, and hence differentiable. This property of the image intensity is used by gradient based methods to determine the optical flow. A well known gradient based method is the Horn and Schunk method,[6] where the image intensity function $I(x,y,t)$ in the image plane at a point (x,y) and at time t is expanded using Taylor's series and used to compute the optical flow field. The total derivative of the image intensity between two image frames separated by a very small time interval dt is zero.

$$\frac{dI(x,y,t)}{dt} = 0 \tag{3}$$

It should be noted that this is strictly satisfied only for translation of a rigid body in planes parallel to the image plane. Using the chain rule of differentiation, Equation (3) can be written as

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{4}$$

This can be written as

$$I_x u + I_y v + I_t = 0 \tag{5}$$

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. I_x, I_y and I_t can be measured from the image sequence, and the optical flow (u, v) can be computed. Since we have two unknowns u and v, and one relationship in Equation (5), the problem is considered ill-posed. Therefore, we need one more constraint to regularize the problem and to compute the flow field.

There are many different ways of regularizing the problem, and numerous papers have been published on this subject. Horn and Schunk [6] have developed a robust algorithm by assuming that the apparent velocity of brightness pattern varies smoothly almost everywhere in the image. There are many choices for the smoothness constraint, like the first and second derivatives of the intensity or a linear combination of the two derivatives. They have claimed that their method is insensitive to quantization of brightness levels and additive noise. Several other algorithms have been investigated to avoid the smoothness assumption in the image. For example, Hildreth's scheme[5] smooths the flow along one dimensional curves corresponding to the zero crossings of the image. Gong [3] reports that Scott's four line method computes a dense optical flow similar to that of Horn and Schunk, and that the match procedure used in this method is a local mechanism instead of a global regularization. Konrad and Dubois[10] describe a Bayesian estimation method, where constraints on the estimated motion field are obtained through a random field model for the motion. The paper describes the formulation of a multigrid algorithm for Maximum a Posteriori Probability (MAP) motion estimation in order to handle large displacements efficiently, and to speed up the convergence rate of the method. Another Bayesian estimation model called Minimum Expected Cost (MEC) is also described in the same paper. Shahraray and Brown [14] use controlled camera motion to reduce the complexity of the problem, and use tensor product smoothing splines to construct a 3-D continuous approximation to the time varying intensity data. There are many papers which describe (i) the effect of camera rotation on the description of optical flow generated by a planar surface in motion,[7] (ii) the determination of 3-D motion and structure from optical flow generated by several moving objects,[2] (iii) motion and structure determination from point and line matches, and so on. There are many other papers which begin by assuming that optical flow has already been determined and describe how it can be used to perform segmentation of images and to recover relative motion and information about the shape of an object. However, research to obtain the optical flow and to reduce the noise in the system is still ongoing.

Consider a horizontally moving camera where the motion is considered to be along the x axis. Since the component of motion in the y direction is zero, Equation (5) becomes

$$I_x u + I_t = 0$$

The magnitude of velocity in the x direction is given by

$$u = \frac{|I_t|}{|I_x|} \tag{6}$$

If this assumption is made, there is no need for another equation. The use of this method is justified in such situations where the direction of motion of the camera is known. Using central differences, we can write Equation (6) as

$$u(x,t) = \frac{\dfrac{I(x,t+\Delta t - I(x,t-\Delta t)}{2\Delta t}}{\dfrac{I(x+\Delta x,t) - I(x-\Delta x,t)}{2\Delta x}} \tag{7}$$

By making both Δt and Δx equal to 1, the optical flow at any pixel location i can be obtained using the following equation.

$$du(i,t) = \frac{I(i,t+1) - I(i,t-1)}{I(i+1,t) - I(i-1,t)} = \frac{temporal\ difference}{spatial\ difference} \tag{8}$$

A literal implementation of Equation (8) does not yield satisfactory results because of the sensitivity of the gradient methods to motion and depth discontinuities. Also, there will be random errors in the imaging process itself, added to the systematic error during sampling and digitization. It has been observed that the sampling error in space is proportional to the spatial resolution of the camera and the second derivatives of the brightness function.[8] Therefore, the image must be smoothed to reduce both systematic and random errors. Since we use local operators, the spatial and temporal differences must be zero for homogeneous regions, but we find some noise points where both spatial and temporal differences exist. We do a gray scale thresholding to eliminate these noise points. As a result, both spatial and temporal differences give high values at the object boundaries and zeroes for uniform regions. The optical flow determined using Equation (8) will have high values near the object boundaries and becomes indeterminate for uniform regions. The signal-to-noise of the flow is maximum at the object boundaries 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 object boundaries must be extracted. We achieve this by masking the zero crossings of the Laplacian-Gaussian of the image on the optical flow.

In order to use the optical flow data efficiently, it should be computed in real time. We have implemented this algorithm on PIPE. A brief description of this machine is given in the next section.

3. AN OVERVIEW OF PIPE

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.[9] and is commercially available through Aspex Incorporated.[1] It has 8 bit gray scale resolution and operates on 256x256 images at video rate (60 fields per second). It can also operate on images of larger size at lower rates. A complete system can perform over one billion operations per second.

Figure 1 shows the block diagram of PIPE. PIPE consists of the following stages.

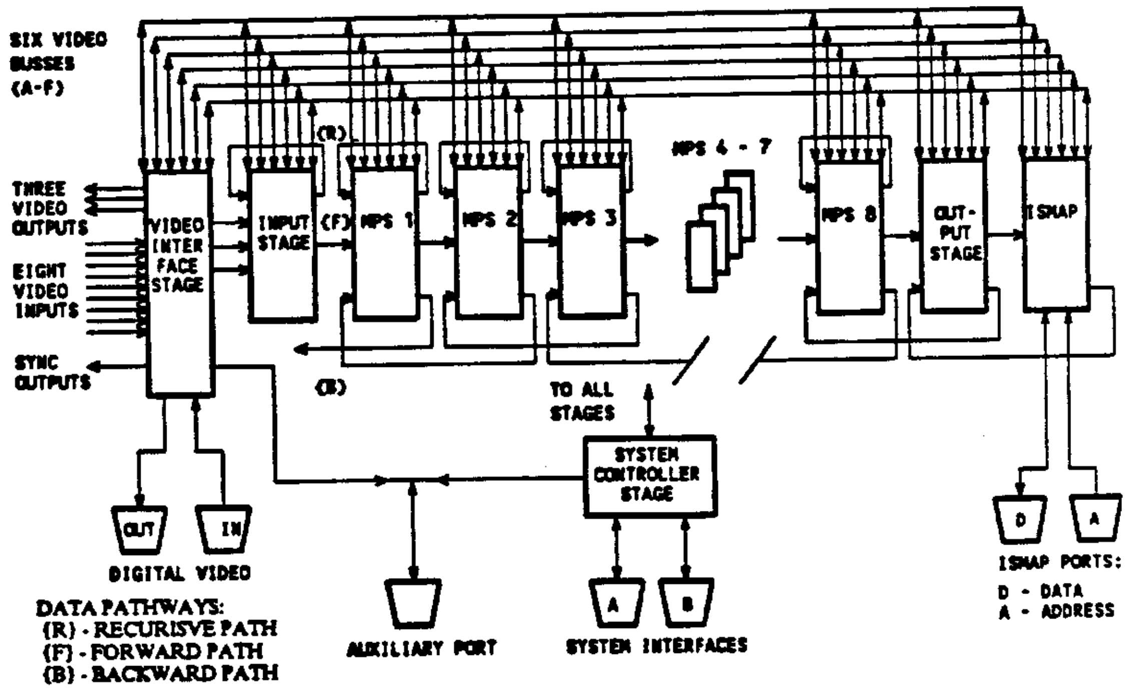

Figure 1: Block diagram of the PIPE system.
(Reprinted with permission from Aspex Incorporated.)

1. Video Interface Stage: This performs analog to digital conversions for receiving analog signals from sources like cameras, video tapes, etc. It also performs digital to analog conversions of signals for sending analog outputs to monitors or other output devices.
2. Input Stage: This contains an input arithmetic logic unit (ALU) and four frame buffers.
3. Modular Processing Stage (MPS): A simplified block diagram of an MPS is shown in Figure 2. All processing in PIPE is performed in one to eight identical MPSs. The number of MPS stages used depends upon the complexity of the algorithm. Each MPS has the following.
   a. A forward path connecting the output of each stage to the input of the next stage (or the output of the last stage to the output stage, described later).
   b. A backward path connecting the output of each stage to the input of the previous stage (this does not apply to the first stage).
   c. A recursive path connecting its output to its own input.
   d. Six video buses connecting the output of any stage to the input of any stage. Note that the previous three paths were local, while the present six paths are global.

[illegible]

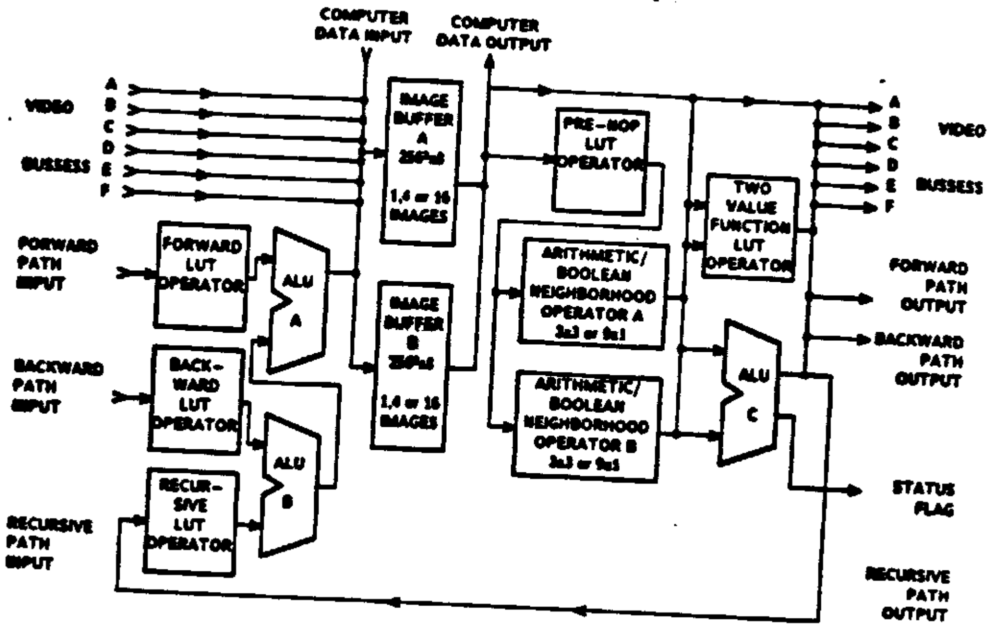

Figure 2: Simplified block diagram of a processing stage.
(Reprinted with permission from Aspex Incorporated.)

e. Two image buffers for storing images. A complete system can store 16 256×256 images in each buffer.

f. Two neighborhood operators (NOPs) to do any arbitrary 3×3 or 9×1 convolutions on the complete image. The NOPs can be arithmetic (ANP), boolean (BNP), or recursive boolean (RNP). Figure 3 shows how these are generated. The BNPs and RNPs are created for each bit plane using logical operators like AND, OR, NOT, and XOR on the variables shown in Figure 3a.

J

A	B	C
D	E	F
G	H	I

E = f(A,B,C,D,E,F,G,H,I,J)
J is the result of previous computation (Used only in RNPs)

(a)

1	2	1
2	4	2
1	2	1

GAUSS.ANP

(b)

Bit Plane	Equation
1	(A^E) | (B^D)
2	(A^E) | (B^D)
3	(A^E) | (B^D)
4	(A^E) | (B^D)
5	(A^E) | (B^D)
6	(A^E) | (B^D)
7	(A^E) | (B^D)

ZCROSS.BNP

(c)

Figure 3: (a) Equation to generate NOPs; (b) a sample ANP; (c) a sample BNP.

For RNPs, J is the result of previous computations, and J is not used for BNPs. Note that the output at each pixel is a function of intensity values at 9 pixels of the input image. Figure 3b shows a Gaussian operator (ANP) used to smooth images and Figure 3c shows the ZCROSS operator (BNP) used to determine the zero crossings of the Laplacian-Gaussian of an image.

g. Four look-up tables (LUTs), one in the forward path (FLUT), one in the recursive path (RLUT), one in the backward path (BLUT), and one just before the NOPs (PNLUT). These LUTs perform point transformations on the complete image, such as changing the number system, multiplying each pixel by some constant, thresholding the image, and so on. Note that the output at any pixel is independent of the values of

[illegible]

its neighboring pixels.

h. Two combining ALUs A and B and one output ALU C perform arithmetic and logical operations on two images. The operations include addition, subtraction, AND, OR, XOR. Note that the output is a function of operations performed on pixels at the same location in two different images.

k. One Two Valued Function (TVF) to do arbitrary point-by-point operations on two images such as multiplication, division, trigonometric operations, min/max operations, taking the square root, etc. It is also possible to do image warping operations such as rotating the image by an arbitrary angle.

4. Output Stage: This consists of 4 buffers. It is usually used as a display buffer to store images.

5. System Controller Stage: This consists of a master timing controller, a master program sequencer, and an interface between PIPE and host computers.

PIPE programs are developed using an interactive graphic development environment called ASPIPE, on a PC AT compatible computer, and downloaded into PIPE. The look-up tables and neighborhood operators required to program PIPE are created using a development environment called LUTGEN, hosted by an AT compatible computer. In addition, there is an Iconic to Symbolic Mapper (ISMAP), which is not used in the present work. For details about ISMAP and for further details about PIPE, the reader is referred to other references listed in this paper.[4,9,11-13]

4. PIPE IMPLEMENTATION OF THE ALGORITHM

The PIPE algorithm for determining the optical flow has the following steps.

1. Digitization and smoothing of the image
2. Determination of the spatial and temporal differences
3. Gray scale thresholding of spatial and temporal differences
4. Determination of the optical flow
5. Determination and masking of the zero crossings on optical flow
6. Segmentation of optical flow

Figure 4 shows the data flow graph for the algorithm implemented on PIPE. A brief description of the algorithm is given in the following steps. Each step is identified in Figure 4 using letters in parantheses.

1. Digitization and smoothing of the image: The image obtained from the input source is digitized (256×242) and represented in two's complement form. This introduces sampling and digitization error in addition to the random error in the measurement process. As we have already mentioned, smoothing reduces both the systematic error (by reducing higher order derivatives) and the random error (by averaging). We have used Gaussian operator to smooth the image both spatially and temporally. We have experimented with 3×3 and 5×5 Gaussian operators to smooth the image spatially. Since the PIPE can handle only 3×3 neighborhood operators, 5×5 operator was approximated using products of two 3×3 operators. A 3×3 Gaussian mask is shown in Figure 3b. Two consecutive 3×3 masks are used to obtain a 5×5 mask as shown in Figure 4, (a)-(b). The temporal smoothing was done using a 3×1 operator (not shown in Figure 4).

2. Determination of Spatial and Temporal Differences: The spatial and temporal differences were determined by using the denominator and the numerator of Equation (8), respectively. The temporal difference was determined by taking the difference between the first and the third images of a sequence (Figure 4, (c)). The spatial difference was determined at (d) in Figure 4 using the 3×3 operator SPATIAL shown as insert in Figure 4.

3. Gray scale thresholding: Both the spatial and the temporal differences will be noisy. Therefore, gray scale thresholding is performed on them as follows:

```
IF (value < lower_threshold) OR (value > upper_threshold) THEN value = 0
ELSE Value = Value
```

The *value* can be either the spatial or temporal difference. In Figure 4, (e) and (f) show the points where GTHRESH is used for gray scale thresholding.

[illegible]

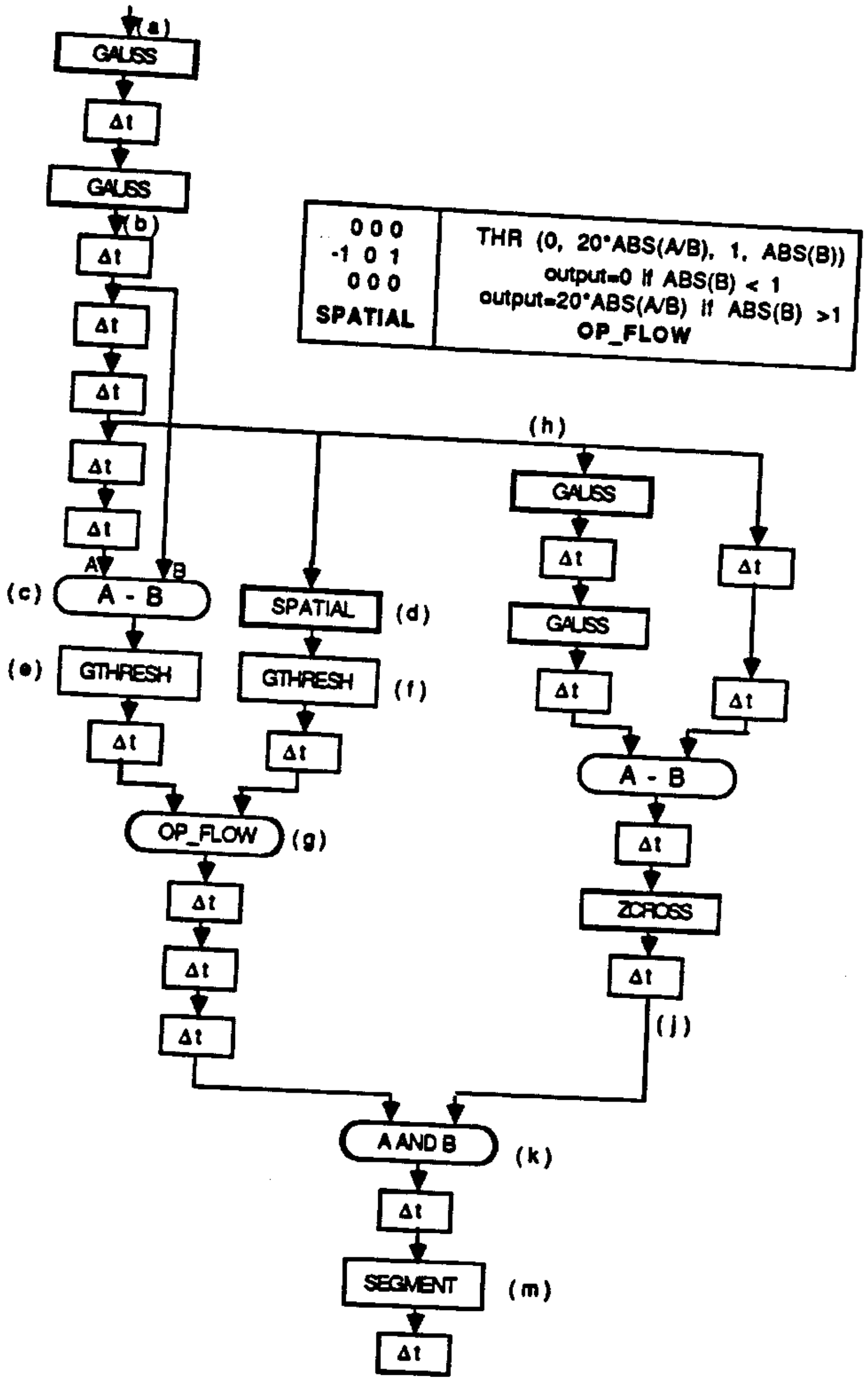

Figure 4 : Data flow graph for the algorithm. SPATIAL and OP_FLOW are also shown as inserts

[illegible]

Theoretically, it is not necessary to threshold the spatial and temporal differences. But in practice, it is observed that these images are noisy, and yield values even at points where there are no significant changes in intensity. These may give rise to false optical flow values and so should be eliminated. Figure 5 shows the difference between unthresholded and thresholded images.

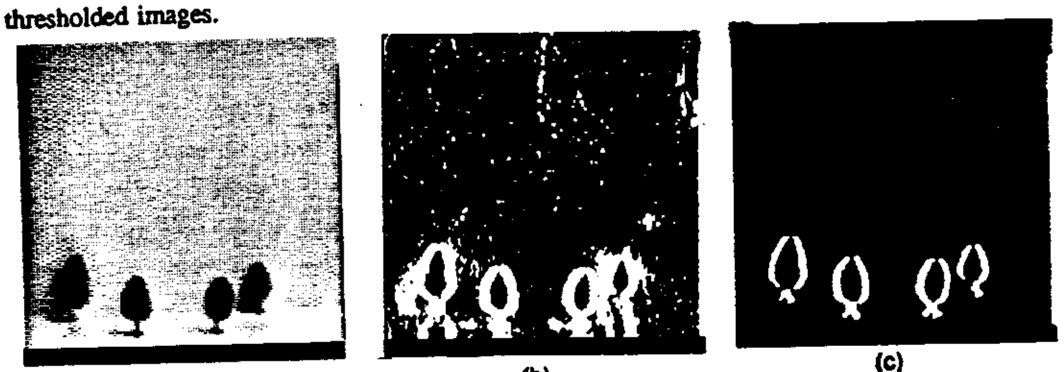
(a) (b) (c)

Figure 5: Effect of gray scale threshold on an image. (a) Original image from a sequence of images; (b) Spatial difference of (a) without gray scale threshold; (c) spatial difference of (a) with gray scale threshold

4. Determination of optical flow: It is observed that the error in temporal difference due to sampling is a function of the square of the optical flow.[8] Optical flow values significantly greater than one pixel per frame time will have large errors. Therefore we consider small optical flows only (less that 2 pixels per frame-time). Since PIPE does not handle floating point numbers, the optical flow is multiplied by a constant. Here, it is multiplied by 20. The TVF look up table OP_FLOW for PIPE implementation of Equation (8) is shown as an insert in Figure 4. Optical flow is determined at (g) in Figure 4.

5. Determination and masking of zero crossings: The zero crossings are determined by using the operator ZCROSS (see Figure 3c) on the Laplacian-Gaussian of the image (Figure 4 (h)–(j)). The edges (object boundaries) thus obtained are masked on the optical flow to obtain optical flow values only on the edges (Figure 4, (k)). The optical flow obtained is considered to be the best estimate of the optical flow.

6. Segmentation of optical flow: Equation (1) shows that the range is inversely proportional to optical flow. Therefore, range segmentation can be performed by segmenting the optical flow. Segmentation is performed at (m) in Figure 4 by using the following interactive program from the host computer.

```
FOR flow = 1 TO 255
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 depth 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 (2).

$$\textit{range discriminability} = r_d = \frac{d_1 - d_2}{d_1} \times 100 \qquad (9)$$

5. EXPERIMENTS, RESULTS AND DISCUSSION

A top view of the setup for our experiments is shown in Figure 6. 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

[illegible]

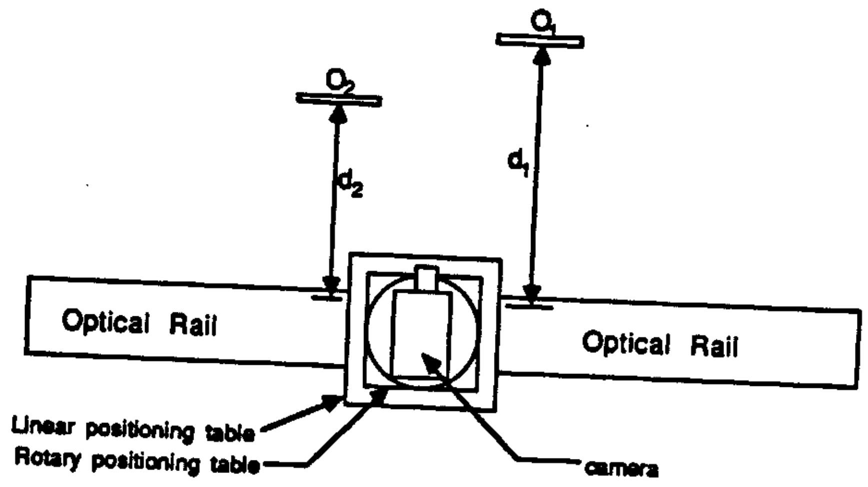

Figure 6: Experimental setup

(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 discriminability was calculated. The experiment was repeated by decreasing the difference $d_1 - d_2$ until further segmentation was not possible.

Several experiments were conducted using this algorithm. First, the input image was smoothed temporally using a 3×1 Gaussian operator, and spatially using 2 consecutive 3×3 Gaussian operators to approximate a 5×5 operator. The optical flow was thresholded to determine the range discriminability of the method. It was found that the results were quite satisfactory, with a range discriminability r_d of 20%. The experiment was repeated by dropping the temporal smoothing. It was found that r_d did not change significantly, although there were a few more noise points. The experiment was then repeated by dropping the spatial smoothing. This resulted in many more noise points. The difference in the results of these three cases was in *the number of misclassified points* (N_{mis}). The experiment to determine N_{mis} was conducted 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 6. 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 flow = 1 TO 255
IF (flow < noise-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_{mis}, 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 one with a lower value was considered misclassified. N_{mis} was calculated using the following equation.

$$N_{mis} = \frac{\text{total number of misclassified points}}{\text{total number of points in the image}} \qquad (10)$$

The number of misclassified points N_{mis} was least when the image was smoothed both temporally and spatially. N_{mis} was not significantly affected when the temporal smoothing was dropped. N_{mis} was seriously affected when the spatial smoothings were dropped. Thus it was concluded that the noise decreases when smoothing increases. Also, the experiment proved that spatial smoothing has more influence on N_{mis} than does the temporal smoothing. The results are given in the following table. Also, Figure 7 shows some of the results of our experiment.

Effect of smoothing an image on N_{miss}

Test #	Type of smoothing	N_{miss}
1	Spatial only	3.9
2	Spatial and temporal	2.3
3	Temporal only	6.6
4	No smoothing	8.7

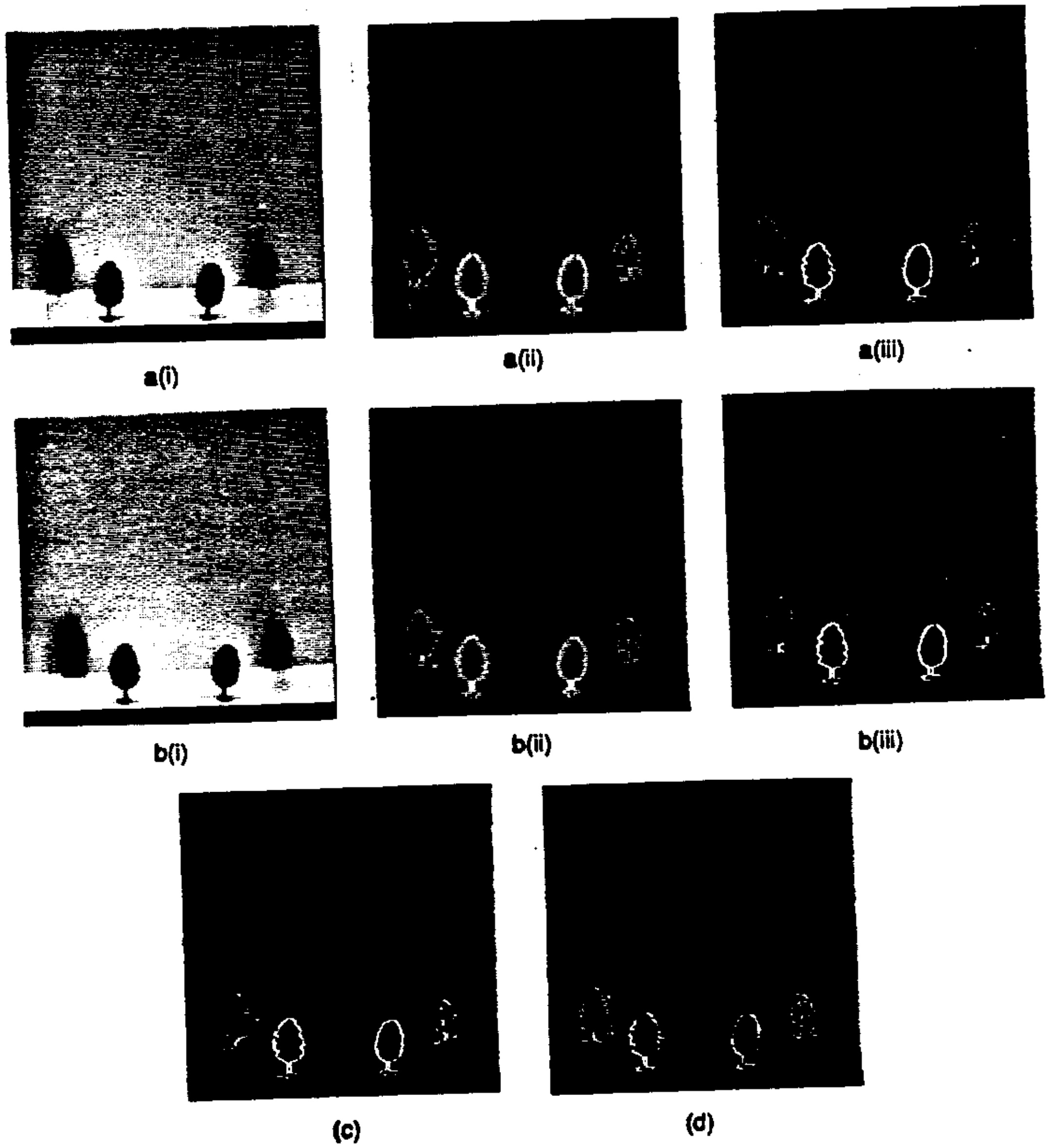

Figure 7: (a) Results using spatial smoothing only. (i) one of a sequence of 3 input images; (ii) optical flow for (i); (iii) optical flow masked with zero crossings. (b) Similar results using both spatial and temporal smoothings. (c) Final result as in a(iii) with temporal smoothing only. (d) Final result as in a(iii) without any smoothing.

If apriori knowledge of the environment is available, the PIPE implementation of Equation (8) can be changed to obtain maximum range discriminability. Let us assume that we have a situation where the farther object O_f is at a depth d_f from the camera, and let the optical flow corresponding to this depth 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 with a look-up table in PIPE to obtain maximum range discriminability. For example, suppose a table at a depth 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 15% has been achieved using this technique.

This technique was used by the authors on the image shown in Figure 8a. In this experiment, a tree in the foreground and a railroad car in the background were used. The optical flow was determined and passed through look-up tables to reduce the noise further. The output was thresholded to eliminate the background, and it was passed through expand and shrink operators. When the optical flow values corresponding to the edges of the foreground is passed through the expand operators, the optical flow spreads around the edges, and the optical flow at all these points becomes equal to the maximum optical flow in the neighborhood. If the thickness of the object is less than the distance of the object to the other objects in the scene, sufficient number of expansions can be performed to "fill in" the interior of the object. But at the same time, the optical flow expands to the exterior of the object also. When this output is passed through shrink operators, the optical flow shrinks from the exterior, and the number of shrinks can be selected suitably to retain the shape of the object. This results in a mask which can be used to mask out the foreground (Figure 8c).

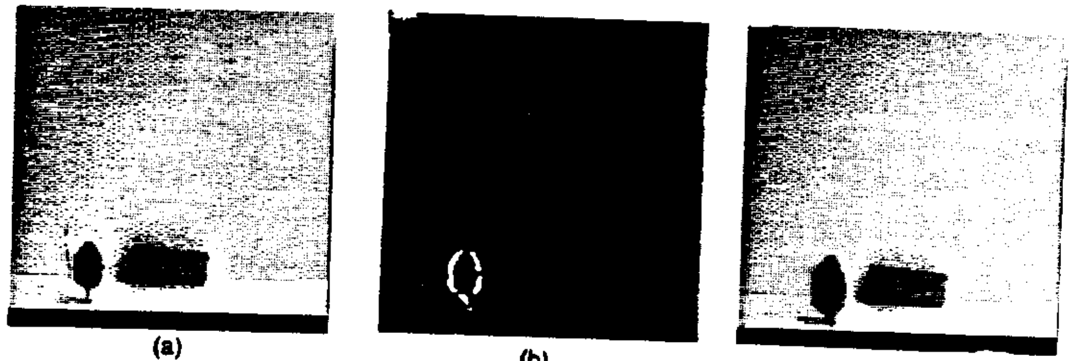
(a) (b) (c)

Figure 8: Image Segmentation using expand and mask technique. (a) Original image in an image sequence; (b) Thresholded optical flow of (a); (c) Expanded mask of (b) masked on (a).

All the experiments described above, except the one where expand and shrink operators were used, computed optical flow in real-time, with an update rate of 30 frames per second and a pipeline latency of $\frac{8}{60}$ of a second. The program with expand and shrink operators had an update rate of 3.75 frames per second.

The results can be improved further by carrying out temporal integration of the optical flow. The optical flow values are integrated over time to obtain an average value of the optical flow. 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 real-time implementation of this algorithm on PIPE.

6. CONCLUSION

The determination of optical flow using gradient based methods is an ill-posed problem. The assumption that the direction of camera motion is known regularizes the problem so that gradient methods can be used successfully. The algorithm can be easily implemented on PIPE, and the results can be obtained in real time. Since the gradient methods assume that the imagery is continuous and differentiable, smoothing the image is essential. Both spatial and temporal smoothings can be performed, but spatial smoothing has more influence on the output than does the temporal smoothing. The results have shown that without both spatial and temporal smoothings the number of misclassified points (N_{mis}) is 8.7% whereas N_{mis} reduces to 2.3% with both spatial and temporal smoothings. The truncation errors in PIPE affect the range discriminability. If prior knowledge of the environment can be obtained, the look-up table for computing the optical flow can be modified to yield greater accuracy. Results have shown that this technique can improve the range discriminability from 20% to 15%. Optical flow is zero for homogeneous regions in the image. By using expand and shrink operations, the optical flow can be expanded and masked on the original image, thus segmenting the whole image. Although determining the optical flow at every instant of time gives good results, the accuracy of the method can be increased significantly by using temporal integration techniques.

7. ACKNOWLEDGEMENT

This work was supported by the Defense Advanced Research Project Agency (DARPA), Tactical Technology Office.

REFERENCES

1. *The PIPE User's Manual*, Aspex, 1987. 530 Broadway, New York, NY 10012
2. Adiv, G., "Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects," *IEEE Transactions on PAMI*, vol. PAMI-7, no. 4, pp. 384-401, July 1985.
3. Gong, S., "Curve Motion Constraint Equation and its Application," *Proc. Workshop on Visual Motion*, pp. 73-80, Irvine, CA., March 20-22, 1989.
4. 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.
5. Hildreth, E.C., *Measurement of Visual Motion*, MIT Press, Cambridge, MASS., 1984.
6. Horn, B.K.P. and Schunk, B.G., "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.
7. Kanatani, K.-C., "Transformation of Optical Flow by Camera Rotation," *IEEE Transactions on PAMI*, pp. 131-143, 1988.
8. 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.
9. Kent, E.W., Shneier, M.O., and Lumia, R., "PIPE (Pipelined Image Processing Engine)," *J. Parallel and Distributed Computing*, pp. 50-78, 1985.
10. Konrad, J. and Dubois, E., "Multigrid Bayesian Estimation of Image Motion Fields using Stochastic Relaxation," *IEEE Transactions on PAMI*, pp. 354-362, 1988.
11. 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.
12. Luck, R.L., "An overview of the PIPE Systems," *Proc. Third Intl. Conference on Super Computing*, vol. 3, pp. 69-78, Boston, 1988.
13. 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.
14. Shahraray, B. and Brown, M.K., "Robust Depth Estimation from Optical Flow," *IEEE Transactions on PAMI*, 1988.

[illegible] R