# Parallel Dense Depth from Motion on the Image Understanding Architecture[1]

**R. Dutta** **C. C. Weems** **E. M. Riseman**
Computer Science Department
University of Massachusetts at Amherst
Amherst, Massachusetts 01003

## Abstract

This paper describes the design and implementation of a Single Instruction Multiple Data (SIMD) depth-from-motion algorithm on the Image Understanding Architecture Simulator. Correspondences are established in parallel for two temporally separated images through correlation. The correspondences are used to determine the translational and rotational motion parameters of the camera through a parallel motion algorithm. This is done by first determining the approximate translational parameters and then constraining the search for the exact translational and rotational parameters. Finally, the dense depth map is computed from the image correspondences and the computed motion parameters. Results are analyzed for three image sequences acquired from mobile vehicles (the Autonomous Land Vehicle, the Carnegie-Mellon NAVLAB, and the UMass Denning Robot). Depths are obtained at an average accuracy of about 8% in outdoor image sequences. The depth maps are processed to locate relatively small obstacles like cans and cones to a distance of about 60 feet. Larger obstacles like hills are located even when they are much further away. Issues related to the speedup and accuracy of the computationally intensive problem of motion analysis are explored in the context of the algorithm.

## 1 Introduction and Motivation

Motion analysis is one of the most computationally intensive tasks in computer vision. Usually motion algorithms have relied on some form of point or feature correspondences between two or more perspective views [1-9]. These correspondence–based approaches take advantage of the image displacements induced by egomotion. Most such methods match a few hundred points or features in two temporally separated images and quantitatively measure the image displacements. A consistent set of motion parameters is then determined to explain these displacements. Once the motion parameters have been determined, the depth of environmental points can be found by using their individual image displacements.

For autonomous navigation it is not enough to compute depth at a few hundred isolated points in the image. In order to detect and avoid obstacles it is necessary to find depth at a dense set of points. In addition, for practical scenarios it is desirable to process a large number of high resolution images within a small period of time. The example below calculates the number of pairs of frames that must be processed per second in a typical scenario for motion analysis (through the use of point-based or feature-based correspondence methods).

Let us assume that our camera has a resolution of $256 \times 256$ pixels and a field of view of $45°$ [2]. Let us further assume that the vehicle is moving with a speed of 50 km./hour and it is necessary to determine depth to a distance of about 10–30 metres at about 10% accuracy (except in the region immediately surrounding the focus of expansion (FOE)[3] where errors in depth are necessarily high). For correspondence–based methods it can be proved theoretically [16] that in order to achieve this accuracy the vehicle must move about 2 m. forward between the processing of successive pairs of frames. Since 50 km/hr is about 14 m/sec, the motion processing system should therefore be able to process a maximum of about 7 pairs of frames

[1]This work was supported in part by the Defense Advanced Research Projects Agency (via Harry Diamond Labs) by contract no. DAAL02-91-K-0047 and NSF grant CDA-8922572.

[2]For better recovery of rotational parameters it is best to have large field of view cameras with high image resolution. However, large field of view lenses give rise to various distortions and lower the effective resolution of the image. Our choice of camera parameters are typical of commonly available image processing systems.

[3]The FOE is the location in the image plane where the translational component of the image displacement is zero. If the camera moves straight ahead along its optical axis with no rotations, then the FOE is at the centre of the image.

per sec. and thus, the computation associated with each pair of frames should be approximately 140 milliseconds.

In spite of this severe requirement for speed, the problem of time complexity has not been adequately emphasized in the area of depth determination. Almost any reasonable motion algorithm needs to solve complicated non-linear equations related to the 5 independent parameters of motion. The sequential algorithms have concentrated on the the use of approximations and search space reduction for the solution of these equations. Furthermore, they have not attempted to compute depth at more than a few hundred locations in the image. However, when 7 complete pairs of frames need to be processed per second and the computation associated with each frame pair involves many floating point computations a parallel design and implementation is essential. If we could completely parallelize the problem, theoretically we could achieve a speedup by a factor of 65,536 times over the equivalent sequential version with a 256 $\times$ 256 array of processors.

None of the earlier work in parallel motion processing [10–15] attempt a comprehensive solution to the problem of dense depth determination from a sequence of images. In the last few years a variety of processor arrays have emerged for solving low level processing in computer vision [17]. The fine grained SIMD array computers have proved particularly versatile for solving such low level tasks. The AMT DAP series, the Connection Machine and the lowest level of the IUA are three machines which embody this computational paradigm. The SIMD class of machine makes it feasible to attempt real-time solutions to the dense depth-from-motion problem. The algorithm presented in this paper is implemented on the Image Understanding Architecture (IUA) simulator. The IUA is a three layered parallel machine specifically designed for image analysis. The lowest layer of the IUA is the CAAPP, a two dimensional grid of 1-bit serial processing elements, operating in the SIMD mode.

A common scenario in ground-based navigation occurs when the vehicle moves forward by undergoing primarily translational motion along with small rotations. In this case the FOE is within the field of view. The algorithm presented in this paper is designed to take advantage of this situation. It first determines the approximate translation and then constrains the search for the exact translational and rotational parameters. *In contrast to other methods which have not demonstrated their ability to recover dense depth maps and locate obstacles, our algorithm is fast, simple and robust.*

We start by providing a brief description of the IUA which emphasizes the features most pertinent to our application.

## 2 The Image Understanding Architecture (IUA)

We provide a brief description of the IUA [18] with particular emphasis on the lowermost level of the machine. The IUA is made up of three levels, each having a particular type of processor:

1. *Low Level* consisting of the Content Addressable Array Parallel Processor (CAAPP).
2. *Intermediate Level* consisting of the Intermediate Communications Associative Processor (ICAP).
3. *High Level* consisting of the Symbolic Processing Array (SPA).

The CAAPP and ICAP levels are controlled by a dedicated Array Control Unit which is directed from the SPA level. The low level processors are ideal for fine-grained SIMD computing, whereas the intermediate and high level processors are ideal for Multiple Instruction Multiple Data (MIMD) computing. Our algorithm uses only the low level of the IUA because of the nature of the task. The low level or CAAPP level is a 256 $\times$ 256 square grid array of custom 1-bit serial processors with local memory, one-bit registers, backing store, an ALU and data routing circuitry. The bit-serial processing elements are linked through a four way (North, South, East, West) communications grid. Intra-level communication within the CAAPP can take place in several ways [18].

## 3 Depth from Image Displacement

This section discusses the mathematical formulation for the algorithm. Figure 1 shows a right-handed coordinate system fixed with respect to the camera. Let us also assume the right hand rule for rotations and consider the case where the camera is undergoing motion. As can be seen from Figure 1 the environmental point $P$, with world coordinate $(X, Y, Z)$, is projected onto point $p$, in the image plane with image coordinates $(x, y)$. Let $f$ be the focal length of the camera, and denote by $\vec{T} = (T_1, T_2, T_3)$, $\vec{\Omega} = (\Omega_1, \Omega_2, \Omega_3)$ the translational and rotational rigid motion of the camera (This implies that $P' = -RP - T$ where $R$ is the rotation matrix and $P'$ is the new position of $P$ after undergoing rigid motion to the next frame).

We shall use the small rotation [4] motion equations, and for simplicity use the following abbrevi-

[4] This means that the magnitude of rotation $\mid \theta \mid \ll 1$. Also, $sin(\theta) \approx \theta$ and $cos(\theta) \approx 1$ to order $O(\theta^2)$. Using the approximation $\mid \theta \mid \ll 1$ we note that even if $\mid \theta \mid = 0.1\, radians$ (*i.e.* $\simeq 6°$), the relative error incurred is 0.2% for $sin(\theta)$ and 0.5%. for $cos(\theta)$. The small angle assumption is not a restrictive one in practical situations because large rotations induce such large image displacements that correspondence algorithms are unable to handle them reliably.

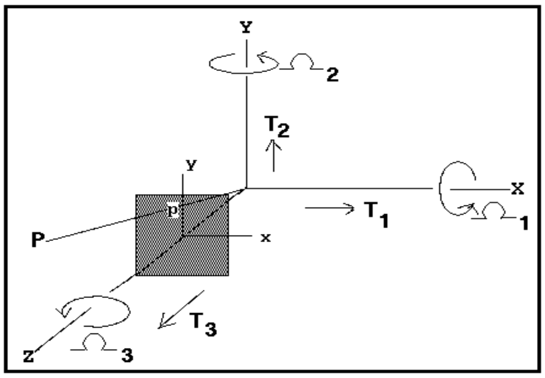



Figure 1: **Coordinate System**

ations:

$$\begin{aligned} I &= 1+\frac{\Omega_2 x}{f}-\frac{\Omega_1 y}{f} \\ J &= \left(\frac{\Omega_1 xy-\Omega_2 x^2}{f}\right)-\Omega_2 f+\Omega_3 y \\ K &= \left(\frac{\Omega_1 y^2-\Omega_2 xy}{f}\right)+\Omega_1 f-\Omega_3 x \\ \alpha &= -fT_1+xT_3 \\ \beta &= -fT_2+yT_3. \end{aligned}$$

With the above abbreviations the image displacement $\vec{l}$, induced by the motion of the camera, is given by

$$\vec{l} = u\,\hat{x} + v\,\hat{y} \equiv (u,v) \tag{1}$$

where $\hat{x}$ and $\hat{y}$ are unit vectors along the $x$-axis and $y$-axis respectively, and

$$u = \frac{J+(\alpha/Z)}{I-(T_3/Z)} \tag{2}$$

$$v = \frac{K+(\beta/Z)}{I-(T_3/Z)}. \tag{3}$$

The depth $Z$ of an environmental point $P$ can be determined from either equation (2) or equation (3). We denote by $Z_x$ the depth determined from equation (2) and by $Z_y$ the depth determined from equation (3). Hence,

$$Z_x = \frac{T_3 u+\alpha}{Iu-J} \tag{4}$$

$$Z_y = \frac{T_3 v+\beta}{Iv-K}. \tag{5}$$

Of course for perfect data, these would be equal; however, in the presence of noise, they will in general be unequal.

It is also possible to write the depth in terms of the displacement vector $\vec{l}$ and the motion parameters by using equations (1),(2), and (3). However, the resulting expressions are cumbersome to manipulate.

In this experimental scenario the vehicle is moving forward into the scene. The motion is mostly translational with small rotations. For this kind of "approximate translational motion" the approximate location of the focus of expansion [19, 20] can be found quite easily. This gives an initial approximate estimate for motion parameters. If the focal length of the camera is unknown, but the focus of expansion is known, then the time-to-adjacency [19] relationship is sometimes used to compute rough depth from the approximate estimate for motion parameters. The time-to-adjacency relationship gives

$$\frac{Z}{T_3} = \frac{D}{d} \tag{6}$$

where

- D = distance in pixels of point $p_i$, from the FOE.
- d = displacement of the point $p_i$.

To find better depths a more elaborate scheme is needed. From equations (4) and (5) we can see that there are two ways of determining depth for a particular set of motion parameters and image displacement. $Z_x$ is the depth that is determined from the $x$-component ($u$) of the image displacement and $Z_y$ is the depth that is determined from the $y$-component ($v$) of the image displacement. As $u$ becomes close to 0, $Z_x$ becomes hard to determine. Similarly, as $v$ becomes close to 0, $Z_y$ becomes hard to determine. When the movement of the vehicle is mostly forward a major portion of the image has significant magnitudes of both $u$ and $v$ and either $Z_x$ or $Z_y$ can give good depths.

For any point $i$ in the image the depth $Z_\mu$ is computed as the average of $Z_x$ and $Z_y$ (except in pathological cases where either $u$ or $v$ is zero). It is possible to arrive at an estimate of the *reliability* of $Z_\mu$ by noting the difference of $Z_x$ and $Z_y$ from each other. The reliability $\zeta$ is defined as

$$\zeta = \frac{\mho(Z_x, Z_y)}{\sqrt{Z_x^2+Z_y^2}}. \tag{7}$$

where

$$\begin{aligned} \mho(x,y) &= |\,x+y\,| \quad if\ x \le 0\ and\ y \le 0 \\ &= |\,x-y\,| \quad otherwise \end{aligned} \tag{8}$$

The reliability scale varies from 0 to $\sqrt{2}$ (no matter what the values of the two depths) with 0 being the best reliability. Qualitatively, we are testing how well the depths computed from the two components of the displacement vectors match. If both depths are positive and equal then $\zeta$ is zero (i.e. high reliability). If both depths are negative then then $\zeta$ is high (low reliability). In general, when one depth is positive and the other depth is negative then reliability is poor, although not as poor as the case when both are negative.

Let $\zeta_i$ be the reliability for depth obtained at point $i$ in the image. Let $n$ be the total number of displacement vectors. $Z_{x_i}$ and $Z_{y_i}$ are the depths computed by using the $x$ and $y$-components respectively of the $i^{th}$ displacement vector and the hypothesized values of the motion parameters. Then

$$\kappa = \frac{\sum_{i=1}^{n} \zeta_i}{n} \tag{9}$$

The motion parameters for which $\kappa$ is minimum is the best set of motion parameters. Unlike several other approaches this optimization criterion allows us to avoid the rather difficult problem of eliminating the depths from the error functions (which has to be done somehow when the deviation between the actual and predicted image displacements must be minimized). We can call the error function $\kappa$, given by equation (9) as the *normalized absolute deviation in directional depths.* The nice property of $\kappa$ is that it varies between 0 and $\sqrt{2}$. The lower the value of $\kappa$ the better the minimization.

We follow this section with the algorithm for depth determination.

## 4 Depth Determination Algorithm

The objective is to recover reliable depths of environmental points over as large a part of the image as possible. The parallel algorithm works in the following stages:

1. Determination of *image correspondence.*
2. Selection of the best image displacements between frames.
3. Determination of the *approximate translational motion parameters.*
4. Determination of the *exact translational and rotational motion parameters.*
5. Depth determination.

### 4.1 Image Correspondence

The algorithm for establishing image correspondence takes as input two temporally displaced images and the maximum possible displacement at any pixel. Restricting the maximum displacement does not significantly limit the efficacy of the implementation since it is usually possible to predict it in advance. The row and column displacements at every pixel are computed through correlation matching.

The parallel algorithm for computing image displacements works as follows

---

**Parallel Correlation**

*Store Frame-1 and Frame-2 of the image sequence in local processing element (PE) memory*

FOR hypothesized displacements within the maximum displacement range DO

BEGIN

*Shift each pixel in Frame-2 by the negative of the hypothesized displacement*

*Compute sum-of-absolute-differences correlation between the pixels of frame-1 and the shifted frame-2 in a spiral pattern [21].*

*If the correlation at a pixel is the minimum obtained until now, then store the hypothesized displacement and the current minimum correlation value in the local PE memory.*

END

After all hypothesized displacements have been considered the displacement corresponding to the minimum correlation at each pixel is the image displacement at that pixel. The value of the correlation gives a measure of the reliability of the match.

---

It should be noted that using sum-of-absolute differences rather than the sum-of-squared differences as the correlation measure saves time by avoiding multiplications. The repeated correlation computations are the most computationally intensive part of the algorithm and this is an important efficiency consideration. Integer representations are also preferable because floating point computations are costly in a bit-serial processor.

### 4.2 Selection of best image displacements

The coterie network of the CAAPP is used to partition the image into equal divisions [18]. For example, in one experiment the $256 \times 256$ image was divided into 64 sub-regions of $32 \times 32$ pixels each. In each sub-region the pixel which has the most reliable displacement vector is selected in parallel. Hence there are as many selected pixels as there are sub-regions. The displacement vectors at these selected pixels are called "selected" displacement vectors. The FOE and motion parameters are determined with the selected displacement vectors.

### 4.3 Approximate Translation

In dynamic imaging situations where the sensor is undergoing primarily translational motion with a relatively small rotational component, *approximate* translational motion algorithms may be effective in determining approximate depth [20]. By restricting the processing to the two dimensions of translational motion, there is a tremendous reduction in complexity from the five dimensions (excluding the scaling component of sensor velocity) of general motion.

The FOE recovery algorithm works as follows. First it draws a line through each selected displacement vector in the image. Let these lines be called "extended" displacement vectors. Then it finds all the possible intersections of the extended displacement vectors and votes in a Hough transform array corresponding to each intersection [22]. The parameter space for the Hough transform is given by the image coordinates where the extended displacement vectors can intersect each other. The number of votes for each intersection is an increasing function of the length and confidences of the displacement vectors forming the intersection. This ensures that longer displacement vectors are more heavily weighted and that more reliable vectors are also weighted more heavily. The point where the maximum number of intersections occur is the approximate FOE. A contiguous region which surrounds the approximate FOE and includes at least $\rho$ fraction of the votes is the region where the exact FOE is likely to be present.

The parameter space for the Hough transform is spread over all the processing elements of the CAAPP. The intersections have an $x$-coordinate and a $y$-coordinate. Since the CAAPP is a two dimensional array, it is easy to map each possible intersection into a distinct PE. The local memory of each PE contains the number of votes for an approximate FOE. The contiguous region where the exact FOE might be located is determined by summing up in parallel the votes gathered by neighbors [22].

### 4.4 Exact Translation and Rotation

Once the region in which the exact FOE can lie is determined the exact translational and rotational parameters can be computed by the optimization method stated in equation (9).

For example let the approximate FOE be at $(70, 55)$ with the exact FOE lying within 10 pixels of the approximate FOE. Then a square whose sides are 20 pixels is formed with $(70, 55)$ as the intersection point of its two diagonals. Then FOE's are hypothesized at each pixel bounded by the square (i.e. starting at $(60, 45)$ and ending at $(80, 65)$). At each hypothesized FOE the normalized absolute deviation in directional depths $\kappa$, is computed for the three dimensions of rotations. The rotations corresponding to the minimum $\kappa$ are the optimal rotational parameters corresponding to the hypothesized FOE. The minimum $\kappa$ is determined among all the hypothesized FOE's. The translation and rotation corresponding to minimum $\kappa$ are the exact motion parameters.

### 4.5 Depth Determination

Once the motion parameters are obtained the depth at each point in the image can be found by using the image displacements and the intrinsic camera parameters. Since each pixel of the image is represented by one processing element in the CAAPP, this stage is totally parallelized. The equations used for this purpose have been described in Section 3.

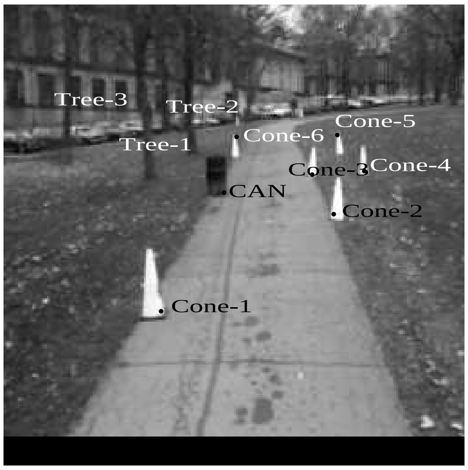



Figure 2: **First frame of the Sequence.**

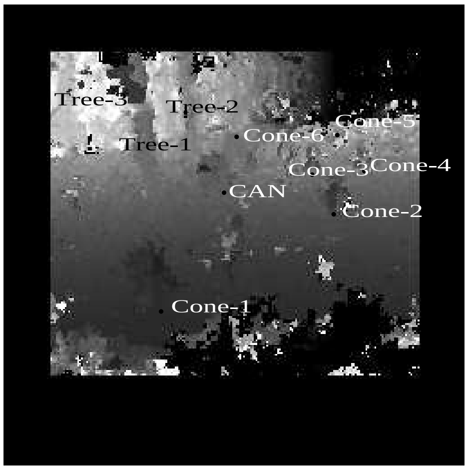



Figure 3: **Computed depth map.**

| Object | True depth (feet) | Computed Depth sample | | |
|---|---|---|---|---|
| | | #1 | #2 | #3 |
| cone1 | 21 | 22 | 23 | 25 |
| cone2 | 36 | 35 | 44 | 33 |
| cone3 | 56 | 45 | 53 | 54 |
| cone4 | 56 | 54 | 55 | 56 |
| can | 46 | 41 | 44 | 43 |
| cone5(*) | 76 | 96 | 66 | 70 |
| cone6 | 76 | 62 | 67 | 55 |
| Tree1 | - | 50 | 51 | 51 |
| Tree2 | - | 58 | 59 | 57 |
| Tree3 | - | 90 | 91 | 91 |

Table 1: **Depth Values of some environmental points.**(*) Cone5 is near the FOE and its depths are erroneous.

# 5 Experiments

The algorithm for parallel depth determination was coded on the IUA simulator. This section contains the results obtained on several image sequences.

**Carnegie Mellon NAVLAB sequence** - The first set of experiments used a sequence of twenty images. The images were collected on the Carnegie Mellon NAVLAB. The first image of the sequence is shown in Figure 2. The vehicle was made to move in an approximately straight line such that the distance between frames was 2 feet. The field of view of the camera was $45^{\circ}$ and $256 \times 256$ images images were collected. In order to determine the ground truth for environmental objects, traffic cones and a can were placed at measured distances (ranging from 21 to 76 feet). Obviously with a total movement of the vehicle of 40 feet, some of the cones disappeared from the scene in later frames. Figure 2 shows environmental objects whose depths had been hand measured (black dots), along with the rest of the scene with objects whose distances were not measured (e.g. trees). It should be noted that in general the scene is quite complex because of the presence of large homogeneously textured regions like road and grass, and the occlusion of the distant buildings through trees.

The quantitative results for the known environmental objects and some unknown objects are shown in Table 1. Three visually selected pixels were marked on each object. The value of depth returned by the algorithm at each of these pixels are recorded in Table 1. These recordings are referred to as "samples" in the table. From the table the average error in depth for the known objects is computed to be about 8% [5]. This corresponds quite well to the theoretical limits on depth determination presented in our previous work [16].

The depth map obtained by using the algorithm between the first and third frame of the sequence is shown in Figure 3 [6]. Complete separation of obstacles (e.g. cones, can) is not possible from the depth map because part of the surroundings of an obstacle in the depth map is at almost the same depth as the obstacle. This is very different from an intensity image where all regions of an obstacle usually have a different image intensity from its surrounding. The darker the color the closer the environmental point is to the camera (*Since the gray-level representation of depth on a printed page is poor , the results will be presented in various representations. On a high resolution display device the depth maps appear a lot better to the human eye.* ). It should be noted that white on the depth map indicates points at which depths are over 120 feet. Black indicates points where depth cannot be computed reliably (e.g. periphery of the image that is not visible in both images, points where displacement vectors are small or erroneous thereby giving rise to wrong depths etc.). Parts of cones, can and trees stand out in the depth map in subsequent representations.

Figure 4(a)-(l) shows the two frames, some intermediate results and several depth maps at different regions of the image. An explanation is necessary for the legends which illustrate the depth maps. The legends are histograms of depth maps. The x-axis shows the depth values and the y-axis indicates the number of pixels with a particular depth value. The shading of the histogram corresponds to the shading of the depth map. For example, in figure 4(e) (whose histogram is shown in figure 4(f)), the lightest region show depths of over 80 feet. The top of cones and can stand out in the depth maps. Observers unaccustomed to seeing depth images often attempt to compare them with intensity images and draw erroneous conclusions. In the case of a cone standing upright on the ground in 3-D, the 2-D image will have the following characteristics:

- The top of the cone is surrounded in the image by locations which are much further away than the cone.
- The bottom of the cone has almost the same depth as the ground in front of it and to its sides.

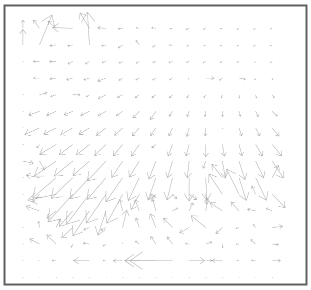
Hence, in the depth map the bottom of the cones and can merge with the surrounding ground whereas the top clearly stands out from its surroundings. Virtually all the major obstacles at least partially stand out in a magnified depth map. We have not magnified all the obstacles in Figure 4 because of space limitations. Nevertheless, it is quite clear from the figures that portions of the two trees, the can and the cones clearly stand out in the depth map. Thus in addition to the quantitative depths the obstacles are also detected. It can be seen that even small obstacles like cones and can are detected quite robustly by this algorithm to a distance of about 60 feet. Larger structure like trees can obviously be detected more easily even when they are quite far away.

**Autonomous Land Vehicle sequence**- A second experiment was done on a sequence collected via the Autonomous Land Vehicle. The data collection process is detailed in [23]. Preliminary results on this sequence are shown in Figure 5. It can be seen that the mountain which is rather far away is clearly identified.

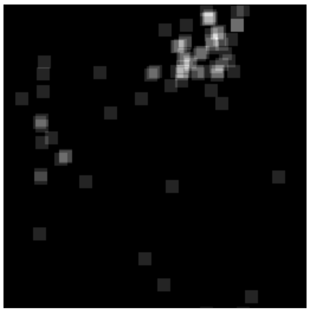
**Umass Denning Robot sequence**- A third experiment was done on a sequence collected via the Denning robot at the University of Massachusetts at Amherst. This image was taken indoor under poor lighting conditions. The robot moved 1.95 feet

[5] The mean depth of each known object is the average of the three smaples that have been shown in Table 1

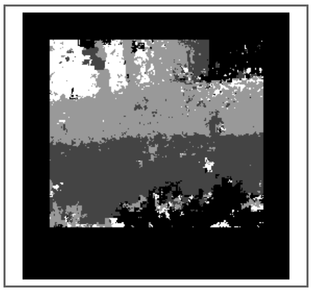
[6] The correspondences in the experimental section are from a hierarchical correlation based algorithm which has not been completely parallelized on the IUA. The completely parallel IUA implementation gives somewhat inferior results for depth.



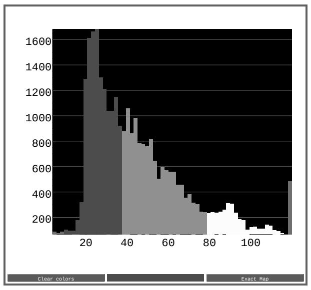
(a) First image. (b) Second image. (c) Image displacements.



(d) Hough Transform.

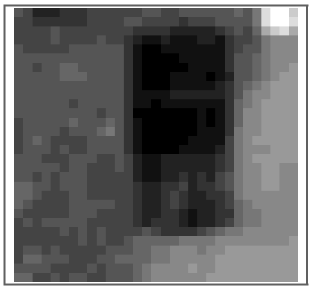


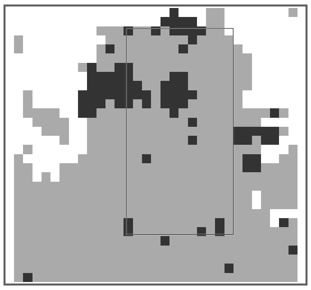
(e) Depth map of full image with legend in fig. f.

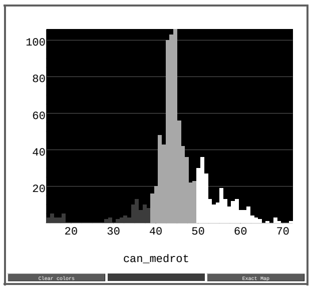



(f) Legend for fig. e.

*Note the gradual increase in depth with the trees standing out*

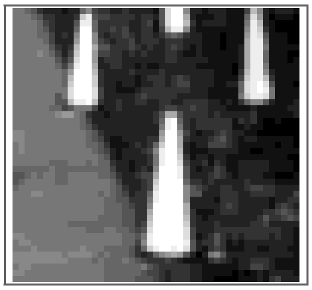


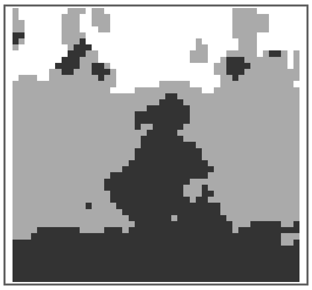


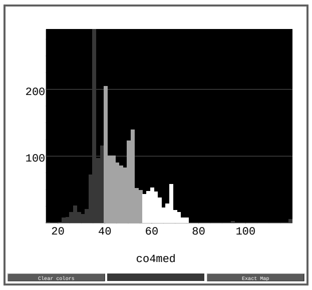



(g) Magnified can from fig. a. (h) Depth map of fig. g. (i) Legend for fig. h.

*The top of the marked can stands out whereas the bottom merges with the ground.*








(j) Magnified cones from fig. a. (k) Depth map of fig. j. (l) Legend for fig. k.

*The tops of the three cones stand out whereas the bottoms merge with the ground.*

Figure 4: Results for the CMU sequence.

between frames. The results for the depth maps are shown in Figure 5.

### Timings

Before presenting the timings let us caution the reader that these are results obtained by running our algorithm on the simulator of an experimental parallel machine under development [7]. For the Carnegie Mellon NAVLAB sequence the PE cycles taken for the various stages are as follows:

| | *CYCLES* |
|---|---|
| *Correspondence stage* | 5315983 |
| *Trans. stage after vector selection* | 78513 |
| *Trans. and Rot. stage with depth* | 35318 |
| *Sum of the above three stages* | 5429814 |

This gives us an estimate of about 0.54 sec. for running the algorithm on an IUA running at 10 MHz. The majority of the time in our algorithm is consumed by the computation of correspondences in searching over a $41 \times 41$ window for possible displacements. Reducing the size of the search window, for example by assuming contraints on the translational magnitude and direction, along with some rudimentary knowledge of depth, can make the algorithm much faster.

It should be noted that with sophisticated mechanical devices like land navigation systems, gyroscopes, inertial navigation systems etc. it is is often possible to constrain the estimates of the motion parameters. If such knowledge is available then this algorithm can be speeded up considerably because the range for hypothesized FOE's and the range of rotational parameters become smaller.

## 6 Conclusions

This research demonstrates the feasibility of fast depth determination through the use of motion algorithms under the SIMD mode of processing. The algorithm is relatively simple because it has been designed specifically for the case of a vehicle moving mostly ahead with small rotations. It is a common scenario in navigation. Even though it is algorithmically simple, the depth computations are robust. Furthermore, there is usually no manual selection among multiple minima for computing motion parameters. Some widely used algorithms (like that proposed by Horn [4]) frequently require manual selection of the optimal motion parameters because of the presence of multiple minima. In this algorithm only the region around the approximately determined FOE is searched for computing optimal motion parameters. This does not normally give rise to multiple minima for small rotations.

The system can compute approximate depth even when the focal length of the camera is not known. This is done by using the time-to-adjacency relationship after determining the approximate FOE. The drawback of the system is that the large number of floating point operations are expensive for a SIMD machine. However, the large number of processors more than compensate for this. The algorithm is being refined for getting faster and better correspondences. Methods for taking care of larger rotations are are also being investigated.

*To summarize, the key contribution of this system is that it is able to recover dense depth maps and locate obstacles quickly, simply and robustly. With reasonable assumptions, the algorithm can run in real time on the IUA.*

[7] The timings given are estimates and are subject to change. The estimates include only the time taken by the CAAPP. The time required for front-end processing is not available. Since, the rotation computation part is still in the process of development, there are some sequential aspects involving front-end processing. This will be reduced in subsequent implementations.

## References


[1] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. In *Image Understanding 1984*, pages 179–193. Ablex Publishing Corporation, 1984.

[2] R. Tsai and T. S. Huang. Uniqueness and estimation of 3-d motion parameters of rigid bodies with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 13–27, January 1984.

[3] O. Faugeras, F. Lustman, and G. Toscani. Motion and structure from motion from point and line matches. *International Conference on Computer Vision*, pages 25–34, 1987.

[4] B. K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4(1):59–78, 1990.

[5] J. W. Roach and J. K. Aggarwal. Determining the movement of objects from a sequence of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 554–562, November 1980.

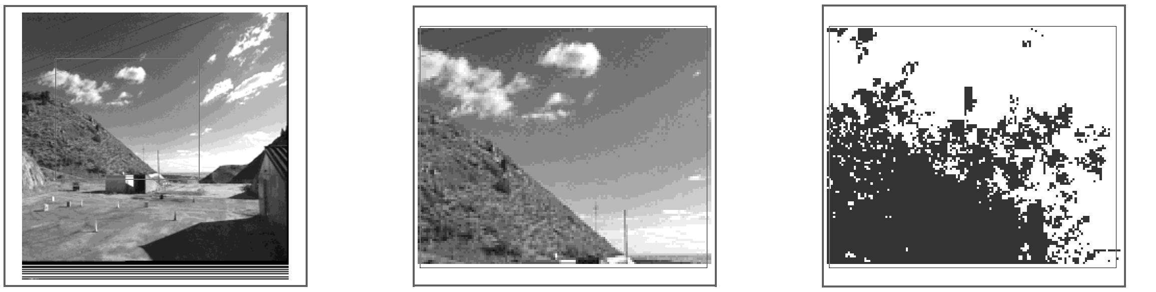
[6] Gilad Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 384–401, July 1985.

[7] A. Bruss and B. K. P. Horn. Passive navigation. *Computer Vision Graphics and Image Processing*, pages 3–20, January 1983.

[8] H. C. Longuet Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, pages 133–135, September 1981.

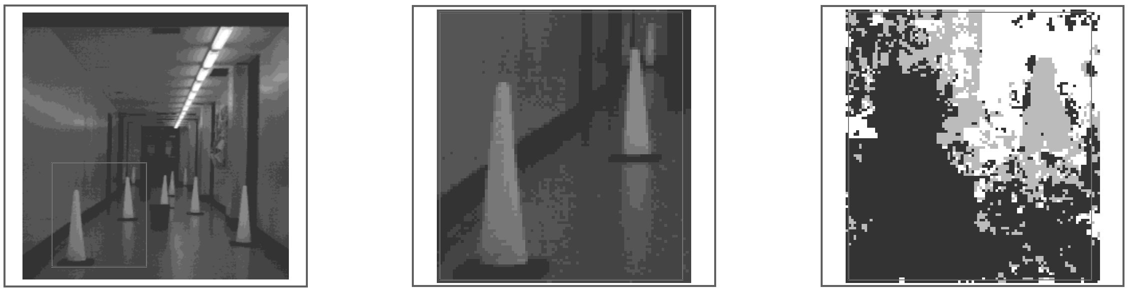
[9] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 451–476, May 1989.

[10] C. Koch, H. T. Wang, B. Mathur, A. Hsu, and H. Suarez. Computing optical flow in resistive networks and in the primate visual system. *Proceedings IEEE Workshop on Visual Motion, Irvine, Calif.*, pages 62–72, March 1989.



(a) First image of ALV seq. (b) Magnifed region of fig a. (c) Depth map of fig. b.

*The far away hill is shown to clearly stand out with the foot merging with the ground .*



(d) First image of UMass seq. (e) Magnified region of fig. d. (f) Depth map of fig. e.

*The two cones stand out very clearly with the botoms merging with the ground.*

Figure 5: Results for the ALV and the Umass Hallway sequence


[11] S. Gong and M. Brady. Parallel computation of optic flow. *European Conference on Computer Vision*, pages 124–133, 1990.

[12] F. Heitz, P. Perez, and P. Bouthamy. Parallel visual motion analysis using multiscale markov random fields. *IEEE Workshop on Visual Motion*, October 1991.

[13] A.N.Choudhary, M. K. Leung, T.S. Huang, and J.H. Patel. Parallel implementation and evaluation of motion estimation system algorithms on a distributed memory multiprocessor using knowledge based mappings. *International Conference on Pattern Recognition*, 1990.

[14] H. H. Bulthoff, J. J. Little, and T. Poggio. A parallel motion algorithm consistent with psychophysics and physiology. *Workshop on Visual Motion*, 1989.

[15] D. T. Lawton M. Streenstrup and C. Weems. Determination of the rotational and translational components of a flow field using a content addressable parallel processor. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 401–404, 1983.

[16] R. Dutta and M. A. Snyder. Robustness of correspondence-based structure from motion. *International Conference on Computer Vision*, December 1990.

[17] P. M. Dew, R. A. Earnshaw, and T. R. Heywood. *Parallel Processing for Computer Vision and Display*. Addison-Wesley, 1989.

[18] C. Weems, S. Levitan, A. Hanson, E. Riseman, J. Nash, and D. Shu. The image understanding architecture. *International Journal of Computer Vision*, Volume 2(3):251–282, 1989.

[19] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall Inc., 1982.

[20] R. Dutta, R. Manmatha, E. Riseman, and M. Snyder. Issues in extracting motion parameters and depth from approximate translational motion. *Proceedings DARPA Image Understanding Workshop*, Volume 2:pages 945–960, April 1988.

[21] Charles C. Weems. An algorithm for a simple image convolution on the titanic content addressable parallel array processor. Technical report, COINS Dept., Univ. of Massachusetts at Amherst, 1983. COINS Technical report 83-07.

[22] R. Dutta. *Depth from Motion and Stereo: Parallel and Sequential Algorithms, Robustness and Lower bounds*. PhD thesis, Computer Science Dept., University of Massachusetts at Amherst, 1993. (Forthcoming).

[23] R. Dutta, R. Manmatha, L. R. Williams, and E. M. Riseman. A data set for quantitative motion analysis. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 159–164, June 1989.