# CONNECTION MACHINE STEREOMATCHING

Michael Drumheller

Thinking Machines Corporation
245 First Street
Cambridge, Massachusetts 02142

**Abstract:** This paper describes a parallel real-time stereomatching algorithm and its implementation on the Connection Machine™ computer, a new massively parallel computing system. The main features of the algorithm are 1) real-time performance, 2) the full exploitation of the *ordering constraint*, 3) a representation that easily maps onto a parallel computer architecture, and 4) the ability to efficiently use a variety of matching primitives. Some results, including timings, are shown for both real and synthetic data. Also discussed are the use of color information and some subtle variations of the basic algorithm.

## 1 Introduction

Research on stereomatching has been, until recently, a slow and painstaking process. Most stereo algorithms are computationally expensive, requiring many seconds, minutes, or even hours to run on conventional computers. Such a long feedback delay for changing the algorithm or the data makes it impractical to refine stereo algorithms by experimental methods.

In this paper we will show how a fine-grained massively parallel computer has been used to solve the speed problem and to develop a new stereomatching algorithm. The algorithm was outlined in a section of [2].

## 2 Description of the Connection Machine computer

The Connection Machine computer is a fine-grained massively parallel computing system designed and built by Thinking Machines Corporation for research in artificial intelligence. The prototype contains 65,536 1-bit serial processors which can communicate with each other by two distinct mechanisms. One of these mechanisms has the topology of a boolean 16-cube and is called the *router network*, or simply "the router." The other mechanism consists of a four-connected $x$-$y$ grid called the *north-east-west-south connections*, or "NEWS." NEWS is used for operations requiring *local* communication, such as convolutions and relaxation algorithms. The router is used for *global* operations such as permuting, sorting, merging, summing, histogramming, region-growing and image sampling. Each processor has 4K bits of memory. The machine is programmed in a *single instruction, multiple data* fashion from a host computer such as a Lisp Machine™or a VAX™. The computations performed by a particular processor depend on the data contained in that processor's memory. For example, the host machine may broadcast a request to each processor to add two numbers together, conditional upon whether the processor contains a particular piece of data.

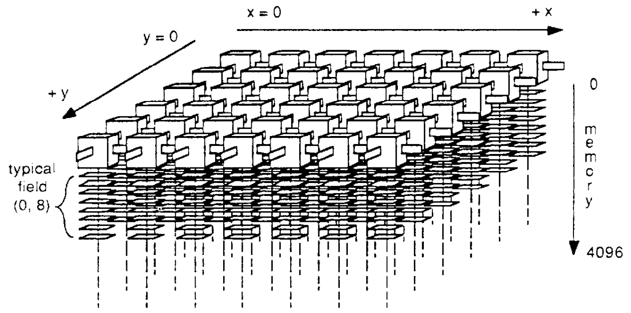

Figure 1. Schematic representation of the Connection Machine computer from the viewpoint of image processing. The processors may be viewed as $x$ and $y$ dimensions; the memory forms a third dimension. An 8-bit image could be stored in the field (0, 8).

In this way, any subset of the processors can "opt out" of a computation.

### 2.1 Virtual Processors

The Connection Machine computer can be space-shared by dividing it into small boolean $n$-cubes whose sizes depend on the number of users and the amount of processing power they need. For example, it was most convenient at the time of this writing to use a 16K-processor subnetwork of the standard 64K-processor configuration.

Image processing applications are programmed on the Connection Machine computer by assigning one processor to each pixel. If the number of pixels is greater than the number of processors, then *virtual processors* are used. Virtual processors are a low-level software facility through which each physical processor simulates several processors in separate blocks of its memory. Virtual processors are invisible to the user; they simply make the machine "seem larger" (and proportionally slower). We used 64K virtual processors (4 per physical processor) to process the 64K-pixel images shown in this paper.

See [4] for a more detailed description of the Connection Machine computer.

### 2.2 Storing images in Connection Machine memory

A convenient way to visualize the Connection Machine computer for this implementation is shown in Figure 1. From the point of view of the NEWS communication mechanism, the processors act as "x" and "y" dimensions and each processor's memory is a "memory dimension." The term *field* will be used to refer to a rectangular block of memory underlying the entire NEWS grid and occupying a contiguous segment of the memory dimension. A field is denoted $(address, length)$. For example, an 8-bit video image might be contained in the field $(0, 8)$ which occupies memory locations 0-7 in every processor (see Figure 1). In this way, each processor represents a pixel. Since the amount of memory in each processor is large compared to the length of a typical intensity value (usually 8 bits), the machine can hold many different superposed images.

## 3 A simple Connection Machine stereomatching algorithm

A simple version of our algorithm, the complete version of which we will describe shortly, consists of the following steps:

*Compute primitives for matching.*

*Compute potential matches between primitives.*

*Determine the amount of local suport for each potential match.*

*Choose correct matches on the basis of local support and constraints on uniqueness and ordering.*

### 3.1 Matching features

This algorithm does not require a particular type of matching primitive. Many different types of primitives could be used; see [1] or [8] for examples. All results shown in this paper were obtained using *zero crossings* [5]. We have used other features; these will be discussed later.

In our implementation we first convolved the image with a small gaussian filter, then detected zero crossings in the output of a discrete laplacian operator.

#### 3.1.1 2-D convolution on the Connection Machine computer

Convolutions are performed by collecting intensity values from nearby processors via the NEWS mechanism, multiplying the values by a broadcast constant, i.e., the appropriate filter weight, and accumulating the product. These steps occur in parallel for all pixels. Some timings for gaussian convolutions are given in Table 1.

The time complexity of convolutions computed in this way is independent of the size of the image and proportional to the size of the filter.

| TOTAL DIAMETER OF GAUSSIAN MASK (PIXELS) | CENTRAL WIDTH "SIGMA" (PIXELS) | CONVOLUTION TIME ON THE CONNECTION MACHINE (SECONDS) |
|---|---|---|
| 7 | 1.2 | 0.022 |
| 13 | 2.2 | 0.042 |
| 31 | 5.2 | 0.107 |
| 49 | 8.2 | 0.180 |

Table 1. Actual timings for gaussian convolution on the Connection Machine computer. It should be noted that these figures are for a true gaussian convolution computed using a sampled gaussian kernel, not for the commonly-used approximation obtained by iterated local averaging.

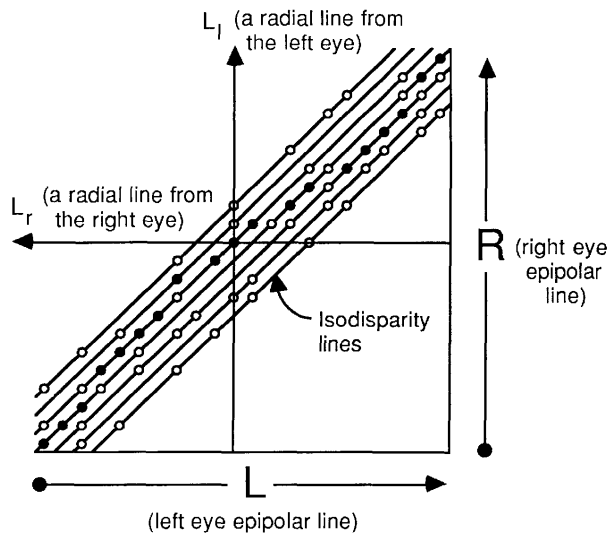
$L_l$ (a radial line from the left eye)
$L_r$ (a radial line from the right eye)
R (right eye epipolar line)
Isodisparity lines
L
(left eye epipolar line)

Figure 2. Schematic representation of stereomatching geometry (redrawn from [6]). This diagram represents the 1-D stereomatching problem at a particular $y$ coordinate, that is, for a pair of epipolar lines or scanlines. The black and white circles are potential matches, or points in $x$-$d$ space where the features from both images are compatible. The black circles are hypothetical correct matches. The uniqueness constraint states that there can be at most one match along any radial line from either eye (such as $L_l$ and $L_r$).

### 3.2 Computing potential matches

The set of potential matches in a one-dimensional stereomatching problem, i.e., for a pair of corresponding epipolar lines, can be represented by the diagram in Figure 2. These diagrams can be "stacked" perpendicular to the page to obtain a three-dimensional set of potential matches. This representation of the stereomatching problem is very convenient for mapping a stereo algorithm into the Connection Machine computer, since the resulting set of potential matches can be stored in a field in the machine (see Figure 3, compare with Figures 1 and 2).

For simplicity, we have assumed that the images are perfectly registered and that all epipolar lines are horizontal. Potential matches are allowed to occur between zero crossings of the same sign on corresponding epipolar lines. Note that this implements the *compatibility constraint* [6]. For $D$ disparity values ranging from $d_i$ to $d_f$, we compute the set of potential matches in the following way:

*Allocate a field $P = (paddr, D)$ to contain the set of potential matches. Initialize $P$ to contain zero everywhere.*

*Allocate two 2-bit fields, $L$ and $R$, initialized to contain the zero crossings of the left and right images.*

*While holding $L$ stationary, "slide" $R$ horizontally one pixel at a time along the x-axis of the NEWS grid, from $x = d_i$ to $x = d_f$. After the $i^{th}$ shift, write a 1 into the field $((paddr + i), 1)$ at each $(x, y)$ where $L$ and $R$ contain identical zero crossings.*

For example, in the third step, after the first shift of $R$, the field $((paddr + 0), 1)$ would be 1 at every point $(x, y)$ where the disparity for $L(x, y)$ could be $d_i$.

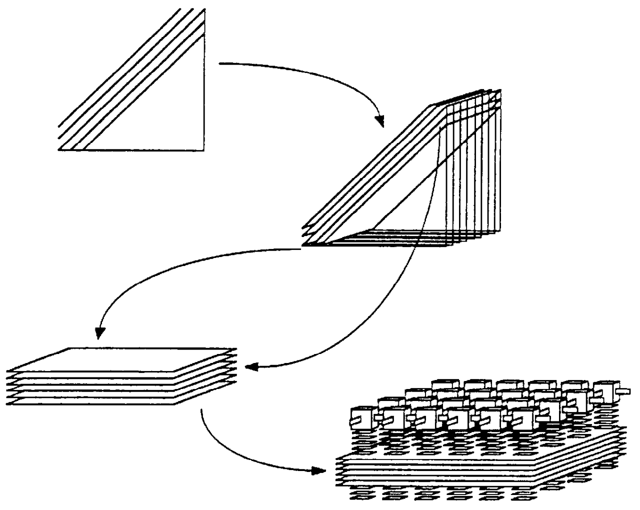

Figure 3. Diagram describing how the geometry of stereomatching is mapped into the Connection Machine computer. The 1-D situations (as in Figure 2) are "stacked" in the $y$ direction to form a complete 2-D stereomatching problem.

After this operation, the field $P$ represents a rectangular block of $x$-$y$-$d$ space with some configuration of matches (1's) embedded in it.

#### 3.2.1 Image registration

As stated so far, this algorithm assumes that the stereo images are perfectly registered with respect to the epipolar lines. In practice, of course, this is rarely the case. In our implementation, we first "undistort" the images to compensate for perspective distortion and camera misalignment. The positional error of each pixel is measured using a semi-automatic system based on a test pattern which, in effect, produces a stereo pair with no false matches. Each pixel is then "sent" to its corrected position via the hypercube communication mechanism.

Due to rounding-off of pixel displacements, it is inevitable that more than one pixel will collide at the same destination pixel. We take the average of the colliding pixels as the value of the destination. It is very convenient to use the router for this computation, since it automatically combines colliding messages with a user-specified "combining function," such as *MAX, MIN, LOGIOR,* etc. In our case, the combining function is *ADD.* The destination pixel is normalized according to the number of pixels that collide there.

### 3.3 Gathering local support

Our first step at distinguishing the correct matches from the false ones is to apply a *continuity* or *smoothness* constraint. Many algorithms based on such constraints have been developed [6] [9] [10] [11] [12].

#### 3.3.1 Three-dimensional convolution

A straightforward way to measure how well each disparity satisfies the smoothness condition is to convolve the three-dimensional region of $x$-$y$-$d$ space contained by the field $P$ with a three-dimensional kernel that gathers support from locally smooth disparity configurations. There are many different kernels, or *support functions*, that will do a good job on this task. Reference [6] uses a very simple support function (or "excitatory region") that is circular, uniformly-weighted, and flat, i.e., it occupies only one level in the disparity dimension. More elaborate, 3-D support functions are described in [9] and [12]. It should be noted that these algorithms do not perform a simple linear convolution. Prazdny's algorithm, for example, includes a non-linear step designed to ignore irrelevant matches and cut down on computational costs [12].

Three-dimensional convolutions are computed in a manner similar to two-dimensional convolutions. In the 3-D case, however, a memory location in each processor might accumulate values not only from neighbors in the $x$ and $y$ dimensions, but also in the *memory* dimension.

### .4 Enforcing uniqueness

The *uniqueness constraint* [6] allows a left image feature to be matched with only one right image feature and *vice versa.* Every working stereo algorithm uses the uniqueness constraint or something similar to it. It expresses the fact that under normal circumstances a single physical object does not simultaneously give rise to a single feature in one image and many features in another image.

Figure 2 and its caption illustrate the uniqueness constraint. In the simple algorithm which we are now describing, potential matches are retained if they have the maximum local support score along the radial lines projecting from each eye. This process of *non-maximum suppression along lines of sight* has been called the *winner-take-all* approach; it is analyzed in detail in [14]. It is used, with slight variations, by the algorithms in [9], [12] and [14], and it is similar to the *inhibition* employed by the cooperative algorithm in [6].

## 4 A new algorithm combining uniqueness and ordering constraints

### 4.1 The forbidden zone

Every potential match is surrounded by an hourglass-shaped region extending through the $d$ and $x$ dimensions, as shown in Figure 4. This region is called the *forbidden zone*, see [13]. Any straight line lying in the forbidden zone must intersect *no more than one match*, unless the scene contains transparent or narrow occluding objects. Examples of such a scene include a pane of glass with markings on both surfaces or a vertical wire suspended in front of a textured wall. These situations can give rise to violations of the *ordering constraint.*

Assuming that the scene contains none of the situations just described, any surviving match must be unique not only along the left and right eye radii ($L_l$ and $L_r$ in Figures 2 and 4), but along any line situated between them, such as $L$ in Figure 4. The set of all such lines fills the forbidden zone completely. Therefore, we implemented an algorithm in which *a potential match is eliminated unless its local support score is greater than that of any other match in its entire forbidden zone.*

If there are $D$ disparity levels, the forbidden zone surrounding each potential match contains approximately $D^2/2$ other potential matches. Therefore, the *a priori* time complexity of an algorithm that considers every score in ev-

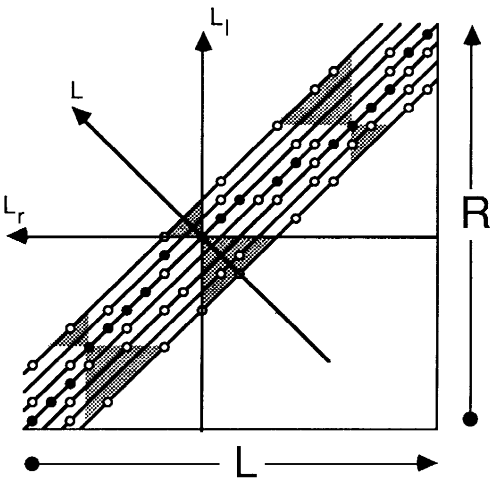

Figure 4. The forbidden zone. Enforcing uniqueness in the directions of $L_l$ and $L_r$ in this diagram corresponds to allowing only one match per image feature (recall Figure 2). If we assume that the scene contains opaque objects and no narrow occluding objects, then a straight line lying *inside* the forbidden zone, such as $L$ above, must also contain at most one match. Consideration of all such lines implies that every correct match must be the only match in its own forbidden zone. The forbidden zones of a few matches are shown above (shaded). Note that none of the black dots (hypothetical correct matches) has another black dot inside its forbidden zone.

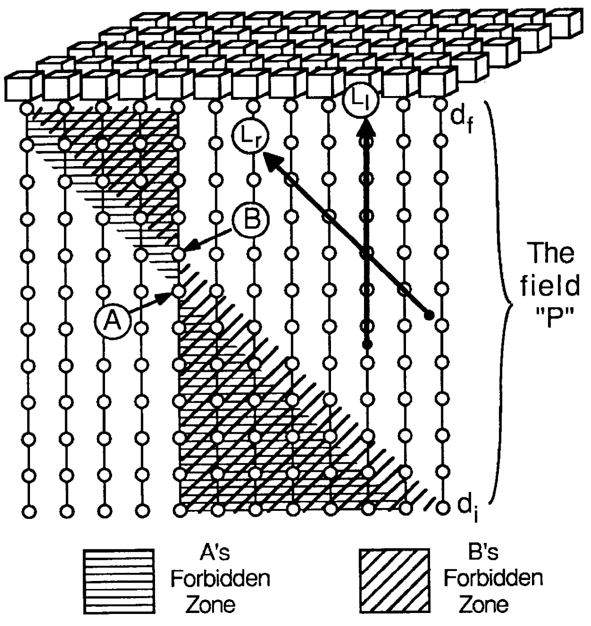

Figure 5. Algorithm for non-maximum suppression over the entire forbidden zone (see text). $L_l$ and $L_r$ (from Figures 2 and 4) are redrawn here to make the relationship between this figure and Figure 4 explicit. Note, however, that the line $L_r$ is tilted 45 degrees. This is a consequence of the way the field $P$ was constructed in section 3.2. In particular, lines parallel to $L_l$ lie inside a single processor's memory, but lines parallel to $L_r$ are oriented as shown.

ery forbidden zone is $O(D^3)$. However, an $O(D^2)$ algorithm exists, which we now describe.

### 4.1.1 An $O(D^2)$ algorithm for non-maximum suppression over the entire forbidden zone.

Figure 5 shows a slice of the field $P$, through the *x-memory* plane, in the neighborhood of a single processor. Figure 5 is equivalent to Figure 4, but it makes the relationship between Figure 4 and the Connection Machine architecture more explicit. Notice that here the forbidden zone is "skewed." This is a consequence of the way the field $P$ was constructed in section 3.2. In particular, a vertical line in Figure 4 maps into a vertical line in Figure 5, but a horizontal line in Figure 4 maps into a diagonal line in Figure 5.

In our algorithm, a potential match must examine the local-support scores of every potential match in its forbidden zone. The Connection Machine implementation steps through each disparity level $i$, first up ($i = d_i, ..., d_f$) then down ($i = d_f, ..., d_i$). On the upward pass the lower "lobes" of the forbidden zones are considered. For example, in Figure 5, the match $A$ checks whether its score is greater than all scores in the horizontally cross-hatched lobe below it. If this is not the case, then $A$ is disqualified. Next, the match $B$ checks whether its score is greater than all scores in the diagonally cross-hatched lobe below it. If this is not the case, then $B$ is disqualified. Note, however, that the maximum value found in the horizontally-shaded lobe can be cached and used to help compute the maximum in the diagonally-shaded lobe. In particular, in order to compute the maximum value in the entire diagonally-shaded lobe, $B$ needs to examine only the scores covered by *diagonal shading alone*, because the scores covered by *both diagonal and horizontal shading* have already been examined (by $A$). In this way, the time complexity of the foribdden zone computation can be kept down to $O(D^2)$. (Note that a similar "downward pass" is necessary to cover the upper lobes.)

## 5 Results and discussion

Some results of running our algorithm on natural and synthetic images are shown in Figures 6 through 10.

### 5.1 Time complexity of the algorithm

Given the efficient implementation of the forbidden-zone computation described in the previous section, the most time-consuming component of the entire algorithm is the local support step, which involves an expensive 3-D convolution. This operation requires time proportional to the number of disparity levels being investigated. Therefore, for a particular local-support function, the time complexity of the entire algorithm is roughly $O(D)$.

### 5.2 Using 3-D support functions

References [9] and [12] describe algorithms that are based on the winner-take-all approach and which use elaborate support functions. These functions are circularly symmetric in the $x$-$y$ plane, with a butterfly-shaped cross-section. The values of the function decrease gradually from a nonzero weight at the center to zero at the extreme perimeter. Such a function is designed to respond to a configuration of matches that is locally smooth and roughly planar, and which may have a nonzero *disparity gradient* [9].

The Connection Machine implementation allows the use of such 3-D support functions, but the best results we have obtained so far have been with simple 2-D kernels. We intend to perform further research on this topic.

### 5.2.1 Color vectors and other matching features

It is possible to combine information from various color channels to improve stereomatching results. One method that we have tried is to use the *sign of convolution* (SOC) as a matching primitive. This was used by Nishihara in the PRISM stereomatching system [8].

Our approach was to compute the SOC for each of three color channels, then load them into separate positions in a 3-bit field of "SOC color vectors." These vectors were used instead of the 2-bit zero crossings to compute the potential matches. This was found to give a noticeable improvement for very contrived scenes. For example, we ran such an algorithm on a scene containing a matte white hammer on a matte white background, with a *randomly colored random dot stereogram* projected onto the objects to provide a rich texture for matching. This technique, called *unstructured light*, was invented by Nishihara [8]; he used it for a single broad color channel. We noticed a severe drop in the number of matches and a roughly proportional drop in the number of errors. The method offered little or no improvement for typical natural scenes. We do not regard these results as conclusive; rather, we mention them in order to stimulate interest in the use of color in stereomatching.

## 6 Conclusions

Stereomatching can be performed extremely fast on a massively parallel processor such as the Connection Machine computer. The use of this machine also helps us represent the problem easily, since the geometry of the stereomatching situation naturally maps into such a parallel architecture.

The Connection Machine computer has been used to implement an efficient new algorithm, *winner-take-all using the entire forbidden zone*. The new algorithm exploits uniqueness and ordering constraints more fully than previous similar algorithms.

The tremendous speed afforded by the Connection Machine computer makes it possible to experiment with sophisticated computer vision algorithms, such as the stereomatching algorithm described here, interactively.

## 7 Acknowledgments

The author is indebted to T. Poggio for his guidance in this work. Many of the ideas in this paper originated in discussions with him.

Connection Machine is a trademark of Thinking Machines Corporation.

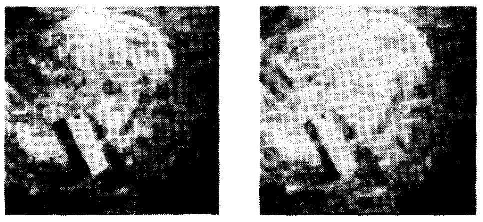
Figure 6. A natural stereo pair. The scene consisted of a terrain model approximately 1 meter wide, with a simulated building added.

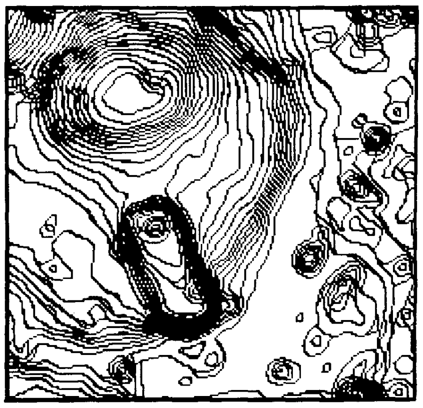
Figure 7. Contour map for the natural stereo pair. The disparities were computed using the simple version of algorithm, i.e., with non-maximum suppression only in the directions of $L_l$ and $L_r$ (from Figure 2). The algorithm was run over a disparity range of 23 pixels; the support region was a flat square 23 pixels on a side. Computation time for the stereomatching algorithm (not including edge detection) was 0.95 seconds. The contour map was computed by drawing isodisparity lines after interpolating the disparity field. (The interpolation algorithm used a "rubber sheet" model. This is basically an iterative solution to the heat equation, performing approximately 1000 iterations of nearest-neighbor averaging at 20 bits of precision and with known disparities held fixed, followed by a slight gaussian smoothing, all computed in 1.5 seconds).

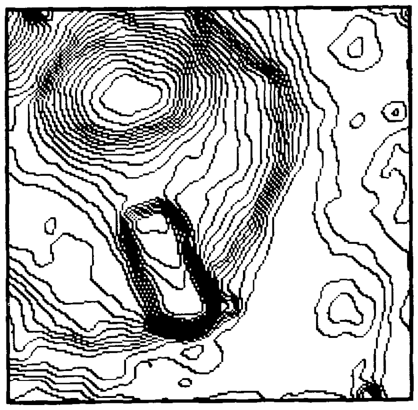
Figure 8. Another contour map for the natural stereo pair, *using the full-forbidden-zone algorithm*. All other parameters are the same as in Figure 7. Note that drastic errors (such as the peaks and erratic elevations appearing in Figure 7) are almost non-existent in this contour map.

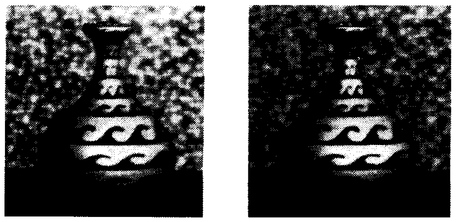

Figure 9. A computer-generated stereo pair. The scene consisted of a vase in front of a textured backdrop.

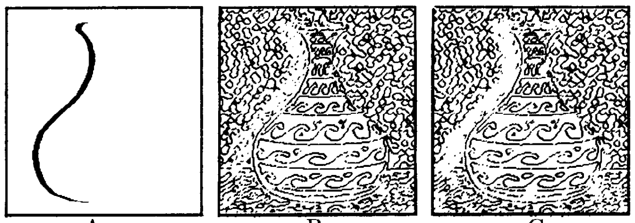

Figure 10. (A) shows the occluded region for the synthetic stereo pair, where no matches should be found. (B) shows the locations where matches were found using the simple algorithm. (C) shows the locations of matches found using the full-forbidden-zone algorithm. Note that the full-forbidden-zone algorithm was more successful at eliminating matches in the occluded region, where the ordering constraint is violated.

## REFERENCES

[1] **Canny**, John F. "Finding Lines and Edges in Images," *M.I.T. A.I. Memo 720*, 1983.

[2] **Drumheller**, Michael and **Poggio**, Tomaso. "On Parallel Stereo," *1986 IEEE International Conference on Robotics and Automation*, April 1986, pp. 1439 1448.

[3] **Grimson**, W. Eric L. "From Images to Surfaces," M.I.T. Press, Cambridge, MA, 1981.

[4] **Hillis**, W. Daniel. *The Connection Machine*, M.I.T. Press, Cambridge, MA 1985.

[5] **Marr**, D. and **Hildreth**, E. "Theory of Edge Detection," *Proc. Roy. Soc. London*, vol. B 207, pp.187-217, 1980.

[6] **Marr**, D. and **Poggio**, T. "Cooperative computation of stereo disparity," *Science* 194, pp.283-287, 1976.

[7] **Mayhew**, J.E.W. and **Frisby**, J.P. "Psychophysical and computational studies towards a theory of human stereopsis," *Artificial Intelligence* 17 (1981), pp.349-385.

[8] **Nishihara**, Keith. "PRISM: A practical real-time imaging stereo matcher," *M.I.T. A.I. Memo 780*, Cambridge, MA, May 1984.

[9] **Pollard**, S. B., **Porrill**, J., **Mayhew**, J. E. W. and **Frisby**, J. P. "Disparity Gradient, Lipschitz Continutity, and Computing Binocular Correspondences," *Artificial Intelligence Vision Research Unit AIVRU ref. no. 010*, University of Sheffield, England, 1985.

[10] **Ohta**, Yuichi and **Kanade**, Takeo. "Stereo by Intra- and Inter-scanline Search Using Dynamic Programming," *Carnegie-Mellon University Technical Report* CMU-CS-83-162, 1983.

[11] **Baker**, H. H. and **Binford** T. 0. "Depth from Edge and Intensity Based Stereo," *Seventh International Joint Conference on Artificial Intelligence*, August 1981, pp. 631-636.

[12] **Prazdny**, K. "Detection of Binocular Disparities," *Biological Cybernetics*, 52, pp.93-99, 1985.

[13] **Yuille**, Alan L., and **Poggio**, Tomaso. "A Generalized Ordering Constraint for Stereo Correspondence," *M.I.T. A.I. Memo 777*, Cambridge, MA, May 1984.

[14] **Marroquin**, José L. "Design of Cooperative Networks," *M.I.T. A.I. Lab Working Paper 253*, Cambridge, MA July 1983.

[15] **Marr**, D. *Vision*, Freeman, San Franscisco, pp.111-125, 1982.