INTERACTIVE COMPLEXITY CONTROL AND HIGH-SPEED STEREO MATCHING

Gruia-Catalin Roman, Andrew F. Laine, and Kenneth C. Cox

Department of Computer Science
WASHINGTON UNIVERSITY
Saint Louis, Missouri 63130

ABSTRACT

This paper is concerned with the development of a novel approach to edge based stereo matching. In the context of an incremental matching strategy we have replaced the traditional hierarchical (coarse-fine) matching by an approach called *complexity control based matching*. Our implementation of this method allows the user to select interactively features which (given the context provided by previous matches) are most likely to be matched successfully. The selection is done at the resolution of the original image and employs a rich set of feature properties (e.g., edge strength, orientation, length, texture, etc.) which may be used alone and in logical combinations. Both feature selection and feature matching algorithms execute at real-time rates and all interactions are via a stereo workstation.

Acknowledgements: This work was supported by Defense Mapping Agency under the contract DMA800-85-C-0010.

1. INTRODUCTION

Human interpretation of aerial images relies heavily on the availability of three dimensional views of the scene. These views are traditionally obtained with the aid of a stereoscope and pairs of photographs. Any attempt to automate the photo-interpretation process requires one to address the issue of depth extraction from multiple views of the scene. Lack of depth data deprives the photo-interpretation system of key information.

The typical stereo vision technique assumes the availability of only two images, by analogy to human binocular vision. The fundamental problem in stereo vision is the matching of corresponding points in the different views. Because of possible occlusions, not all points may have matches and context information is used to infer the depth of the unmatched points [1]. Techniques differ in the strategy they follow with regard to the generation of a unique and consistent set of matches.

Barnard and Fischler [2] surveyed current stereo vision techniques. Most approaches may be classified as area-based or feature-based. Area-based techniques rely on the surface continuity assumption and often involve correlation-based matching. Feature-based approaches focus on intensity variations that correspond to physical and geometric properties and intensity anomalies which may not have any physical relevance. Matching is often done at the symbolic level. Much effort has been devoted to the study of feature-based techniques because they provide better localization and exploit more contextual information [3,4].

Because, at present, no one knows how to build an automated system capable of replicating the human ability to eliminate false matches, reliability requires the participation of a human interpreter, henceforth called user. It is our belief that, in the short term, high reliability is achievable through the synergistic integration of high-speed advanced automation and human participation. The emphasis on real-time interactive stereo analysis distinguishes our approach from work pursued by others.

In the context of an incremental matching strategy we have replaced the traditional hierarchical (coarse-fine) matching by an approach called complexity control based matching. Our implementation of this method allows the user to select interactively features which (given the context provided by previous matches) are most likely to be matched successfully. Feature selection is done at the resolution of the original image and employs a rich set of feature properties (e.g., edge strength, orientation, length, texture, etc.) which may be used alone and in logical combinations. Both feature selection and feature matching algorithms execute at real-time rates and all interactions are via a stereo workstation of a design similar to that used by Ortony and his colleagues [7]. The complexity control is implemented by using the Gould/DeAnza IP-8500 hardware as a data-flow machine executing under the control of a VAX-11/750. The selected features together with features matched during previous iterations are supplied to a high-speed matching algorithm also implemented on the Gould/DeAnza hardware.

The remainder of this paper describes the mechanics of the real-time feature selection and matching. Section 2 contains an overview of the complexity control strategy and explains its role in stereo matching. Section 3 deals with the interactive feature selection algorithm. The description of the high-speed matcher and its performance characteristics are discussed in Section 4. Some concluding remarks complete the presentation.

2. METHODOLOGY OVERVIEW

The difficulty of the stereo matching problem, particularly in urban scenes, favors the use of incremental matching strategies. Hierarchical matching and the use of multiple images are two examples of incremental matching strategies. In the hierarchical (coarse-fine) strategy first proposed by Marr and Poggio [5], for instance, processing starts with a low resolution version of some image pair and proceeds by considering versions of increasing resolution. Matches obtained at one resolution level constrain processing at the next higher resolution level. The 3D model construction from multiple images being pursued by Herman and Kanade [6] is also incremental. Each new image of the

same scene is used to extract additional details about the 3D scene model.

This paper presents yet another incremental matching strategy. It uses only one pair of images and only one level of resolution. Each step involves the selection of a group of features from the left and the right images. The features are selected in such a way as to maximize the likelihood that correct matching will occur. As before, previous matches help the current step by providing useful constraints (ordering, figural continuity, etc.). Selection is based on feature properties that can be extracted directly from the original source images. Examples of frequently used feature properties are size, directionality, edge strength, texture, contrast, etc. We call this strategy *complexity control based matching (CCM)*.

One advantage of our method is that feature position and identity are always preserved by contrast to hierarchical matching where low resolution images merge features (e.g., the sides of a highway) and introduce uncertainty with respect to their actual location. Moreover, in our approach, the feature selection criteria used in each step may be arbitrarily complex and are not predetermined by what is visible at a particular resolution. Flexibility is the key strength of CCM. To illustrate this point consider a hypothetical image where matching the weak edges is easy and could result in the immediate disambiguation of all remaining strong edges. Such a situation can be easily handled by CCM while hierarchical approaches could run into difficulty because the weak edges would disappear as the resolution of the image is reduced.

In our current implementation of the CCM paradigm we focused on matching chains of edgels. A *chain* is a sequence of (8-connected) edgels approximating a straight line segment. Chains are one pixel wide; non-horizontal chains have at most one pixel per scan line when the images are aligned along the epipolar axis; horizontal and nearly-horizontal chains have at most one pixel per column.

Using the CCM paradigm we have constructed a highly interactive visual environment for doing research on stereo matching in urban scenes. The user selects interactively subsets of the chains in the left and the right images and submits them for automatic matching. The appropriateness of the selection is visually checked via a stereo display. The results of the matching are also viewed in stereo and may be visually compared against the original images. Bad matches may be selectively eliminated and corrected. These steps are repeated until the geometry of the scene is extracted correctly and completely. The system's effectiveness is derived, on the one hand, from its rapid response and clear visualization of the progress and status of the computation and, on the other hand, from the judicious intervention of the user.

3. INTERACTIVE COMPLEXITY CONTROL

Complexity control is supported by a system component called *selector*. The selector is a high-speed interactive visually-oriented program. It allows the user to incrementally define and redefine the set of chains to be extracted from the left and right images for input to the high-speed matcher. The same set definition is used for selection of chains in both the left and right images. However, the definition includes parameters which may be adjusted independently for the left and right images.

Conceptually, the simplest form of chain selection may be represented by an expression such as

$$Right\_Selection = \{\, c \mid c \text{ is a chain in the right image and } min\_length_R \le length(c) \le max\_length_R \,\}$$

$$Left\_Selection = \{\, c \mid c \text{ is a chain in the left image and } min\_length_L \le length(c) \le max\_length_L \,\}$$

where *length* is a function returning the number of edgels in c, and *min_length/max_length* are user supplied parameters specific to the right and left images (as indicated by the subscripts). Sets of chains may be combined using using traditional set operations.

As shown in this section, chain properties such as *length* are precomputed and encoded as *characteristic images* while chain sets are encoded as *binary images*. A characteristic image associates with each edgel in a chain a value [1-255] corresponding to some chain property. The nature and range of available characteristic images provides flexibility to the selection process. In a binary image, all the edgels belonging to a selected chain are assigned a value of 1 while the rest of the image is set to zero. Using these representations the *min-max* comparison is reduced to a thresholding operation while the set operations are implemented as bit plane operations.

Interactive chain selection involves the definition and redefinition of the selection formulae and of the threshold parameters. The former is done from the keyboard while the latter via a trackball which is used to set lower and upper bounds; the results of changing the bounds are propagated to the appropriate expressions and immediate visual feedback (in stereo) is provided to the user. The high-speed required for such visual feedback is achieved by executing all calculations within the Gould/DeAnza IP-8500 image processor. Actually, the formulae used in the selection process are encoded as a directed acyclic data-flow graph with the image processor functioning as a data-flow machine. In the next section we explain the conceptual model supporting the selection process and provide an overview of the implementation strategy.

3.1. Model

Selector operations are modelled by a directed acyclic graph called a *logic graph*. The source nodes of the graph are *characteristic images*; all other nodes are *binary images*. Each binary image has an associated operation and operand(s); these define the edges of the graph. The operands of a binary image are its descendants in the graph. Figure 1 shows an example of such a graph; boxes represent characteristic images, circles represent binary images, and lines or pairs of lines labelled with an appropriate symbol represent operations. The implied direction of node descent is downwards, while the implied direction of data movement is upwards. Each node represents a complete image (in our system, 512 x 512 pixels).

There are two categories of operations. *Threshold operations* map a characteristic image and a pair of integers (defining the threshold intervals) into a binary image. *Binary operations* map one or two binary images

into a binary image using the logical operations NOT (¬), AND (∩), OR (∪), SUBT (–), and XOR (⊕).

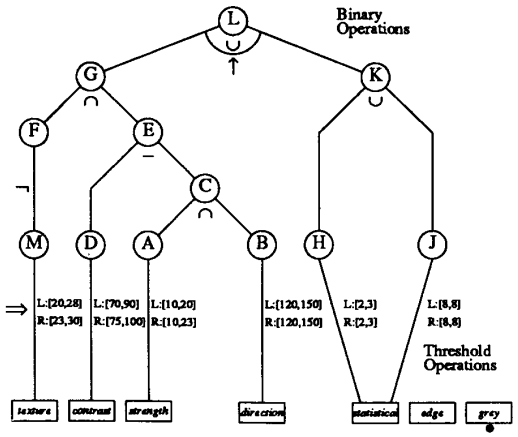


Figure 1: Sample Logic Graph and Pointers.

3.2. User Interface and Selector Algorithm

The user interface consists of a command interpreter. Two classes of commands are allowed, *definitions* and *actions*. Definitions create a logic graph. Actions inspect or operate on a logic graph. Actions may include setting the graph pointers (described below), thresholding, matching, and saving the current definitions or images in a file.

Definition commands are used to define an instance of a logic graph. Node C of Figure 1, for instance, represents the left (right) chains whose edge strength is between 10 and 20 (10 and 23) and whose direction is between 120 and 150 (120 and 150). This representation is evaluated as a dataflow graph during *thresholding*. The resulting changes are propagated upward through the graph.

Interpretation of the logic graph is controlled by three logical "pointers" (see Figure 1). The *current context* pointer refers to an image which serves as a background against which the user can visually evaluate the appropriateness of selected chains. Typically, the source image is used as context because it permits the user to see in stereo the relation between the selected chains and the scene being analyzed. However, any characteristic image may be used as the context as well. In either case, the context image is always displayed as an 8-bit grey scale image. The current context pointer in Figure 1 is identified by the symbol •.

The *current display*, represented by an arrow (↑), points to a binary image which is displayed within a color-coded graphics overlay channel. By moving the current display pointer to any non-source node throughout the logic graph, the user may visually examine the flow of data at that node as a function of the current graph structure and definition. The *current threshold* identifies active thresholding of a previously defined characteristic file, continuously creating a new binary image node. In Figure 1, the current threshold is indicated by ⇒.

Selection Step 1: Chains of length 60 or greater.

Matching Step 1: Correctly matched chains.

Selection Step 2: Chains with orientation between 30° and 60°.

Matching Step 2: Correctly matched chains.

Selection from the left and right image data sets of a stereo pair, occurs simultaneously using two IP-8500 image processors. However, only one logic graph and one set of pointers are needed. Visualization of data flowing through *any node* of the logic graph is accomplished via sophisticated graphics on the stereo workstation. Under user control, thresholding may be active for both the left and right images simultaneously or may affect each side individually. The visibility of *display* information and *context* may be controlled similarly. Thus the selector program provides the user with a powerful complexity control mechanism, couched within a simple conceptual model.

3.3. Performance Characteristics

The DeAnza IP-8500 hardware is used as a data-flow machine to accomplish high-speed parallel execution of all operations required by the logic graph. Individual bit-planes of image memory are used as bit-plane "registers" to store the values (binary images) of graph nodes; only those nodes which are descendants of the display pointer and ancestors of the threshold pointer need to be computed during thresholding. Other nodes needed for the thresholding computation (e.g., E and K in Figure 1) are computed once when the graph pointers are bound and stored in bit-plane registers.

Since all the necessary data for evaluating the logic graph is stored within the image processor's internal memory, evaluation of the logic graph can be performed entirely by the image processor with minimal I/O from the host. A fixed amount of overhead is involved in programming the IP-8500 for each operation, but we are able to keep the cost of each node operation to approximately 1/20 second. Thus, for a tree-shaped graph of depth five the maximum visual feedback is still only 1/4 second.

4. HIGH-SPEED MATCHING

Interactive complexity control simplifies the matching problem. Even relatively simple matchers perform well due to the reduced burden placed on each iterative step. The series of images shown on the previous page, for instance, illustrates the disambiguating power of simple properties such as chain length and orientation. Only the right image is reproduced here - the final matching results are shown at the end of the paper.

The key is to maximize the matcher's speed so as to provide the user with immediate visual feedback. We did this by keeping the matcher very simple and by exploiting the available hardware resources to the maximum. In the current matching algorithm, for instance, the matcher identifies edgel pairs that are unique in the context of the previous matches [8]. The matcher assumes an epipolar camera model; searching for candidates to match a particular edgel is limited to a maximum and minimum disparity range within the edgel's epipolar line. Our next versions of the high-speed matcher will include tests for similarity of contrast across the edge, direction, and texture in the vicinity of the edge.

4.1. Specification

Informally stated, the matcher seeks to identify, within the constraint of some given maximum disparity, pairs of unmatched edgels whose left and right neighbors (if they exist) have been pairwise matched.

Input to the matcher consists of: (1) the set of previous matches, (2) the set of selected edgels to be matched and (3) the disparity parameter δ. E_L (E_R) denotes the set of edgels selected from the left (right) edgel set. M_L (M_R) identifies the set of previously matched edgels on the left (right) side. ΔM_L (ΔM_R) denotes the set of left (right) edgels matched during the current iteration, i.e., $\Delta M_L \subseteq E_L - M_L$ ($\Delta M_R \subseteq E_R - M_R$). (It should be noted that subchains consisting of edgels all lying on the same epipolar line are treated, for matching purposes, as single points and are represented in E_L, E_R, M_L, and M_R by the left most edgel in the subchain.) The parameter δ specifies the minimum and maximum disparity ($\pm\ \delta$ pixels).

The criteria used in our matching algorithm include maximum disparity range, consistency with previous matches (an ordering constraint), and uniqueness.

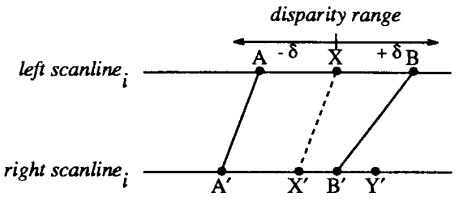


Figure 2. An intra-scanline ordering constraint.

Figure 2 shows that previously matched edgel pairs <A,A′> and <B,B′> provide a context that makes candidate pair <X,X′> unique within disparity range δ. Candidate pair <X,Y′> is eliminated by this ordering constraint although Y′ lies within δ of X.

4.2. Algorithm

In describing the algorithm we will use L and R to refer to the sets of edgels belonging to chains selected from the left and right images during the current iteration step. M_L and M_R will refer to edgels matched during the previous steps. All of these sets are encoded as two-dimensional arrays with indices ranging over the image space. $L(i,j)$, for instance, assumes the value "1" if the position (i,j) contains an edgel belonging to a chain selected from the left image and the value "0" otherwise.

High-speed matching is accomplished in five steps. Each step is symmetric with respect to the processing of left and right edgel sets. This symmetry is exploited by partitioning the pipelined processor into two independent pipelines. Each pipeline processes the left and right edgel sets concurrently, while a control program (host resident) maintains phase between the two. L, R, M_L, and M_R are stored in bit-planes of the high-speed memory arrays within our image processor. (On our Gould/DeAnza IP-8500, each memory array is 512 x 512 x 8 bits.) Most operations required by the matcher can be executed on the pipelined processor in a single frame cycle (1/30 second). A summary of the cost analysis of the algorithm is given in Section 4.3.

Eliminate previous matches.

$$L(i,j) = L(i,j) - M_L(i,j) \qquad [4.2.1a]$$

$$R(i,j) = R(i,j) - M_R(i,j) \qquad [4.2.1b]$$

Label intervals between matches. The set of previously matched edgel-pairs is a bijection. Therefore on each epipolar line, the intervals between matched edgels also match pairwise. In this step, we label the intervals of the previous match set; we assign identical labels to all pixels within a given interval. The labels are used to enforce an ordering constraint. The label sets are denoted B_L and B_R. Equation 4.2.2a shows that a label at B_L(i,j) is computed for each pixel by counting the number of previously matched edgels from $M_L(0,0)$ to $M_L(i,j)$. The result is that all pixels within matching intervals have the same label. Since our memory arrays are 8-bits deep, the labels are actually assigned values modulo 255.

$$B_L(i,j) = \left[\sum_{y=0}^{i} \sum_{x=0}^{j} M_L(x,y) \right] \bmod 255 \qquad [4.2.2a]$$

$$B_R(i,j) = \left[\sum_{y=0}^{i} \sum_{x=0}^{j} M_R(x,y) \right] \bmod 255 \qquad [4.2.2b]$$

Count candidate matches. This step is the heart of the matching algorithm. The purpose of this step is to count the number of edgels in L (R) that could possibly match each edgel in R (L). Candidates are counted only if they are within matching intervals, as defined by B_L, B_R, and δ. At the end of processing, each pixel in S_L and S_R will have the number of candidates found for each edgel in L and R respectively.

For each pixel (i,j) in L, the number of candidate edgels found in R within the interval $i-\delta \le x \le i+\delta$ is computed by a sum of products; the product of $L(i,j)$ and $R(x,j)$ will be "1" only if both $L(i,j)$ and $R(x,j)$ are edgels. Otherwise the product is zero. The term *ordered_match* will have the value "1" only if the points (i,j) and (x,j) fall within matching intervals, (i.e., intervals having the same label number, see equation 4.2.3c). Similarly, equation 4.2.3b shows the corresponding computation for the right sum S_R.

$$S_L(i,j) = \sum_{x=i-\delta}^{i+\delta} L(i,j) \cdot R(x,j) \cdot ordered\_match\ (i,j,x) \qquad [4.2.3a]$$

$$S_R(i,j) = \sum_{x=i-\delta}^{i+\delta} L(x,j) \cdot R(i,j) \cdot ordered\_match\ (i,j,x) \qquad [4.2.3b]$$

$$ordered\_match\ (i,j,x) = \begin{cases} 1 & \text{if } B_L(i,j) = B_R(x,j) \\ 0 & otherwise \end{cases} \qquad [4.2.3c]$$

Eliminate multiple matches. L_0 and R_0, defined in equations 4.2.4a and 4.2.4b respectively, identify the set of matched edges that are uniquely matched in the context of previous matches. Each matched edgel (i,j) in L_0 has the value 1 if the number of candidate matches found for (i,j) is exactly 1. This computation is simply a threshold operation that is implemented through table lookup. It is folded into the next step and requires no cycles.

$$L_0(i,j) = \begin{cases} 1 & \text{if} S_L(i,j) = 1 \\ 0 & otherwise \end{cases} \qquad [4.2.4a]$$

$$R_0(i,j) = \begin{cases} 1 & \text{if} S_R(i,j) = 1 \\ 0 & otherwise \end{cases} \qquad [4.2.4b]$$

Compute final matches. At this point the unique matches of L_0 and R_0 have been computed by *L*eft-to-right and *R*ight-to-left driven matching. However, one-sided uniqueness will result in matching errors due to occlusions. Our matcher requires matches to be uniquely matched in both directions, thereby enforcing a symmetry constraint on all matched pairs. This means that *each edgel* of a candidate pair must satisfy the matching criteria *independently* before the matcher will accept them as a matched pair.

$$\Delta M_L(i,j) = \bigcup_{x=i-\delta}^{i+\delta} L_0(i,j) \cdot R_0(x,j) \qquad [4.2.5a]$$

$$\Delta M_R(i,j) = \bigcup_{x=i-\delta}^{i+\delta} L_0(x,j) \cdot R_0(i,j) \qquad [4.2.5b]$$

After one complete iteration of matching, the sets ΔM_R and ΔM_L contain symmetrically unique matches that are consistent with respect to the ordering of previous matches. The set of previous matches (M_L, M_R) is updated with the new matches (ΔM_L and ΔM_R) and both are displayed in color-coded graphics on the stereo workstation. Control is automatically returned to the selector program.

4.3. Performance Characteristics

A summary of the run-time analysis of our current matching algorithm is shown in Figure 3. Between each step, there is an additional hidden cost of configuration setup. Configuration setup costs typically include programming for I/O control, internal pipeline routing, and loading memory control registers and look up tables. These fixed operations take constant time and are usually done between vertical blanking periods of the display. The total run-time cost shown in Figure 3 is linearly related to the parameter δ, the maximum disparity range. For a disparity range of 24 (±12 pixels), the theoretical time for matching is only 1.75 seconds; empirically, we obtained values in the range 2.6 to 3.0 seconds, reflecting an additional fixed cost of pipeline processor configuration.

High-Speed Matcher Performance Analysis
(for disparity ±δ and 1/30 second per cycle)

Step Description	*Cost (cycles)*
Eliminate previous matches	1
Label intervals between matches	1
Count candidate matches	2δ + 1
Eliminate multiple matches	0
Compute final matches	2δ + 1
Total Cost	4δ + 4

Figure 3: High-speed matcher performance analysis.

5. CONCLUSIONS

In this paper we have presented a new incremental stereo matching approach. Major components of a high-speed interactive system that implement this idea have been built and tested. The current system provides an environment for experimentation with and development of new matching strategies which at a later point in time may be automatically invoked. The interactive nature of the system forced us to exploit to the greatest possible extent the capabilities of the available technology and has helped researchers achieve a better understanding of complex algorithms. The emphasis on interactive visual environments is a trademark of our research group.

6. REFERENCES

[1] Arnold D., and Binford, T. O., "Geometric constraints in stereo vision: Image processing for missile guidance", *Proc. Soc. Photo-Opt. Instrum. Engr.* 238, 1980, pp. 281-292.

[2] Barnard, S. T. and Fischler, M. A., "Computational Stereo", *Comp. Surveys* 14 , No. 4, 1982, pp. 553-572.

[3] Lem, H. S., and Binford, T. O., "Stereo Correspondence: Features and Constraints", *Proc. Computer Vision and Pattern Recognition Conference*, June 22, 1986, Miami Beach, Florida., pp. 373-379.

[4] Baker H., and Binford, T. O., "A System for Automated Stereo Mapping", *Proc. Image Understanding Workshop*, Palo Alto, Calif., 1982, pp. 215-222.

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

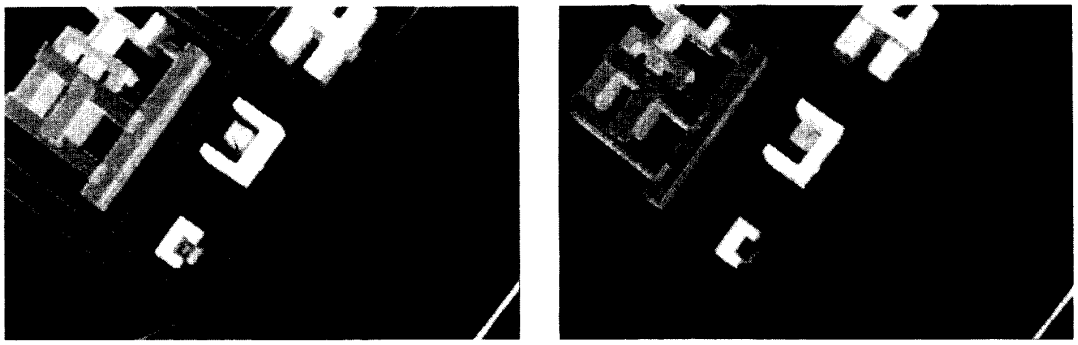
[6] Herman, M. and Kanade, T., "Incremental Reconstruction of 3-D Scenes from Multiple, Complex Images," *Artificial Intelligence* 30, 1986, pp. 289-341.

[7] Ortony, A., "A System for Stereo Viewing," *The Computer Journal* 14, No.2, 1971, pp. 140-144.

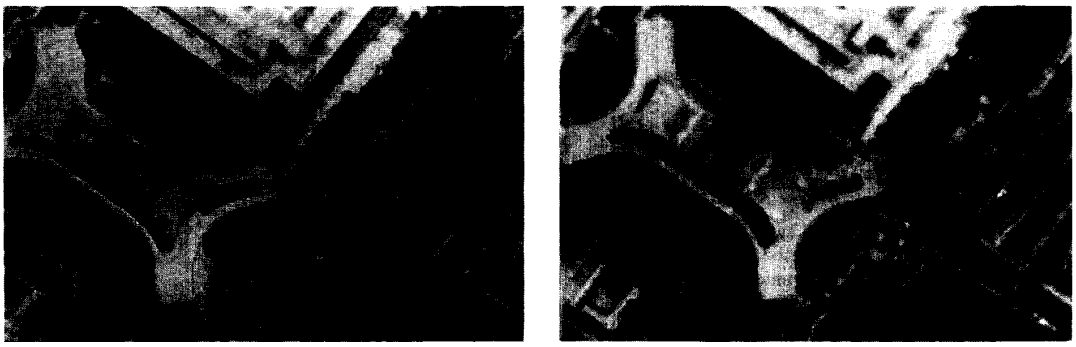
[8] Yullie, A.L. and T. Poggio, "A Generalized Ordering Constraint for Stereo Correspondence", MIT AI Lab Report AIM-777, Massachusetts Institute of Technology, Cambridge, MA, May 1984.

SAMPLE RESULTS

Right image *Left image*



Synthetic Images



Aerial photographs of Washington D.C.