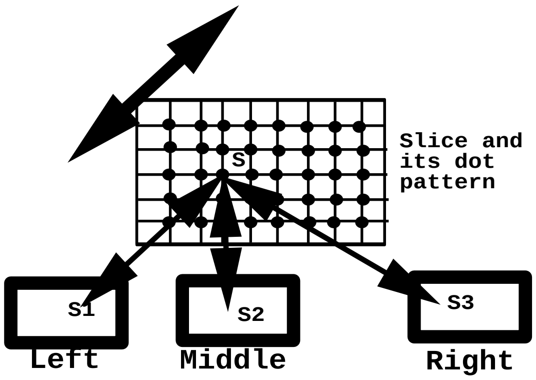
Spatial Filtering using the Active-Space Indexing Method

Sudhanshu K Semwal[1,2] Jun Ohya[1]
[1]ATR Media Integration & Communications Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan 619-02
[2]Department of Computer Science, University of Colorado, Colorado Springs
Colorado Springs, CO, U.S.A. 80933-7150
semwal@redcloud.uccs.edu|ohya@mic.atr.co.jp

Send all correspondences to: Dr. S.K. Semwal, Professor, Department of Computer Science, University of Colorado, Colorado Springs, CO, 80933-7150. Phone: (719)262-3545. Fax: (719)262-3369. E-mail: semwal@redcloud.uccs.edu



Active space creation

Figure 1: Imprint-set (S1,S2,S3) for point S. S1, S2, and S3 are 2D points on the respective camera-images.

```
Procedure DataCollection

 1. Mark a 12x12 grid pattern on a whiteboard.
    Physical grid-cells are 5cm by 5cm
 2. Set up three cameras so that the grid
    pattern is visible in all the three cameras
 3. Capture the three camera-images and store them
 4. Repeat step 3 seven more times, after moving
    the white-board with the grid pattern seven
    more times

end DataCollection
```

Figure 2: Preprocessing: Data Collection Algorithm for scanning a 55cm by 55cm by 70cm active-space.

```
Procedure CreateActiveSpaceIndexing

 1. For a set of left, center, and right camera
    images collected in step 3 of the DataCollection
    Algorithm

    do
     2. Identify all the 12x12 grid-intersection
        points
     3. Store the pixel-location of the grid-points
        along with their 3D-points on the white-board
    end do

 4. Repeat step 1-3, for all the eight sets of three
    camera-images.
end CreateActiveSpaceIndexing
```

Figure 3: (Preprocessing) Creation of the Active Space Indexing data structure.

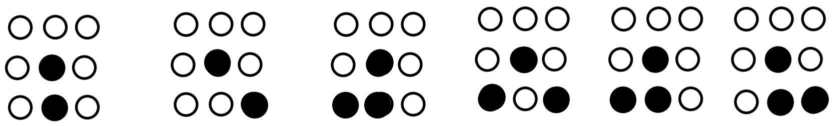
```
Procedure FindingA2DIndex(S)
 // returns the 2D Index (p,q) given a point S on
 //  the slice

 // Let x = Sx, y = Sy
 If (x,y) is within the Slice

    1. Find two consecutive vertical grid lines p and
       p+1 so that x is between p and p+1
    2. Find two consecutive horizontal grid-lines q
       and q+1 so that y is between q and q+1
    3. return(p,q)
  endif
  // otherwise S is not within the Slice
  return (not inside the Slice)
end FindingA2DIndex
```

Figure 4: Returns the 2DIndex or cell which contains the given point S

```
Procedure Find_S(S1, S2, S3)

// Given the imprint Set (S1,S2, S3)
// find the 3D location S

1. Use S1 for the left image, S2 for
   the center camera-image, and S3 for
   the right-camera image

   do

2.   Find indices I1, I2, and I3 for all
     the eight slices.  I1, I2, and I3
     are the grid-cells containing S1, S2, and S3.
     Note that I1, I2, and I3 refer to the projected
     cell locations for the white board locations.
     Therefore, there are eight such possibilities.
     For some white board placements, I1=I2=I3
     and the area would be zero.  Basically we
     are looking for two consecutive whiteboard
     slices between which we expect the point S
     to lie. If there are several white-board locations
     with zero area, then we select the first with
     zero area. If such a situation does not exist
     then conclude that (S1,S2,S3) do not correspond,
     otherwise perform steps 3-4 below.

3.   Using the two consecutive slices in the active
     space, find the 3D location of point S1 on these
     two slices.  This will define a line L1.  Similarly
     obtain lines L2 and L3 using S2 and S3,
     respectively.

4.   Find the minimum distances between the three pairs
     of lines (L1, L2), (L2, L3), (L1, L3).  Take the
     average of the three distances.  If this average
     is greater than the "closeness" criteria, conclude
     that (S1,S2,S3) do not correspod, else calculate the
     nearest points on these lines.  That would define
     the location of point S in 3D. Return point S.

end Find_S
```

Figure 5: Algorithm: Find S provides estimation of point S. If S1, S2, S3 are corresponding closely then precise estimates are obtained.

```
Procedure SpatialMarking

1. For every pixel in all the camera images
   do
2    Identify Significant points by considering the
     (r,g,b) of the 8 neigboring pixels, and
     using thresholding. Some cases when the center
     pixel is identified as significant point are given
     below:
```



```
Note: Mirror cases will also classify the pixel
      as significant

3. If a pixel is identified as significant then

    For all the 8 slices
     do

4.     Find the 2D Index, and add this pixel as
       candidate for being a significant pair for
       that 2D-cell indicated by the 2Dindex on a
       slice. Note that the 2D Index and
       the slice, in fact define a grid-voxel.  And
       we have a 12x12x8 3D-grid of voxels in our
       implementation.
     end for
   end if
  end for
```

Figure 6: Identifying significant points and Marking them in the 3D grid Voxels.

```
Procedure SpatialFiltering

1. For all the 12x12x8 3D grid voxels
   do
// Let p1, p2, p3 be pixels identified as significant
// (by the SpatialMarking process) from camera 1, 2, 3

2.  for i= 1 to p1

3.     S1 = ith pixel in this grid-voxel from camera 1
4.     for j = 1 to p2

5.        S2 = jth pixel in this grid-voxel from camera 2
6.        for k = 1 to p3

7.        S3 = kth pixel in this grid-voxel from camera 3
8.        thisPoint3D = Find_S(S1,S2,S3) // See Figure 3
9.        display(thisPoint3D)

          endfor k
       endfor j
    endfor i
   endfor
```

Figure 7: Generating Corresponding-pairs and 3D points

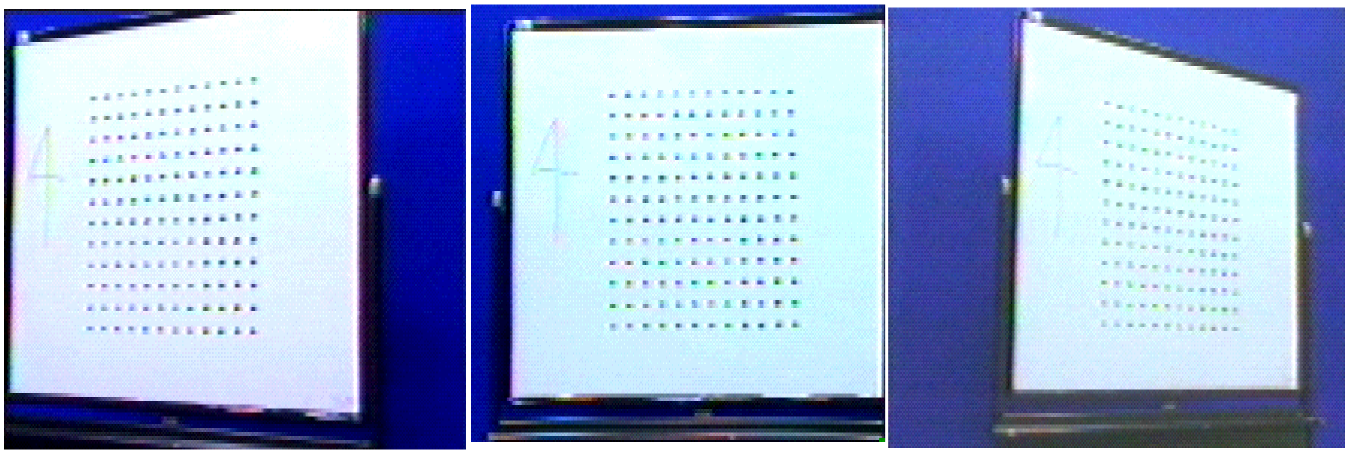


Figure 8: Whiteboard used for Data collection and three views

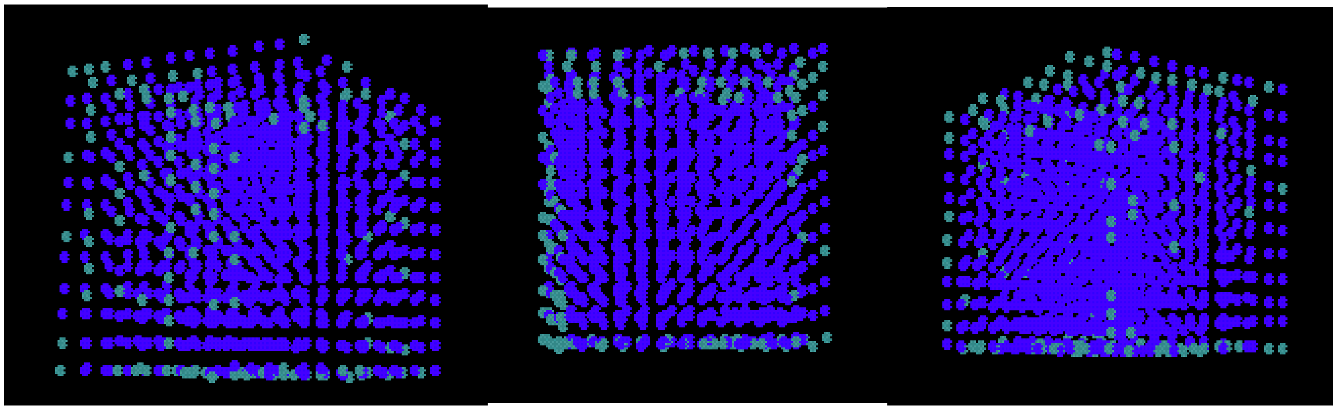


Figure 9: The projection points on all the three cameras for all whiteboard slice positions

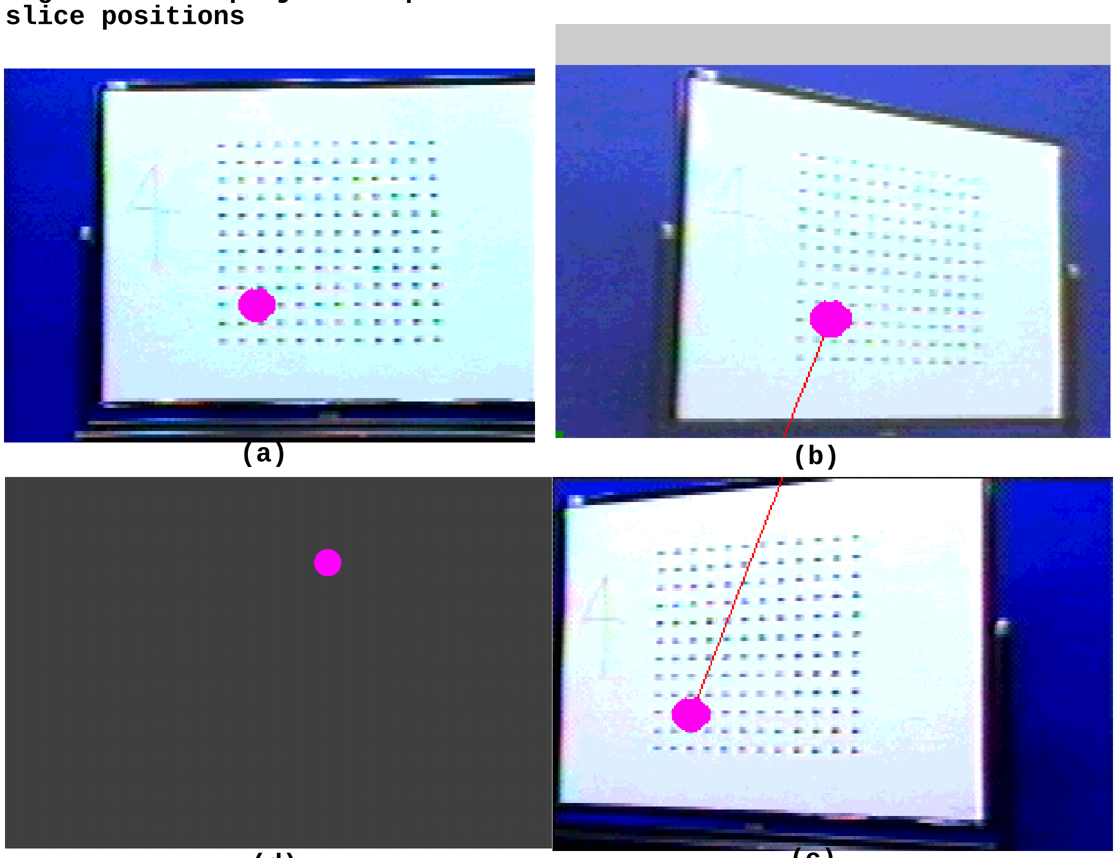


Figure 10: When the same 3Dpoint is specified on all the three camera images (a-c), its precise location (d) can be estimated

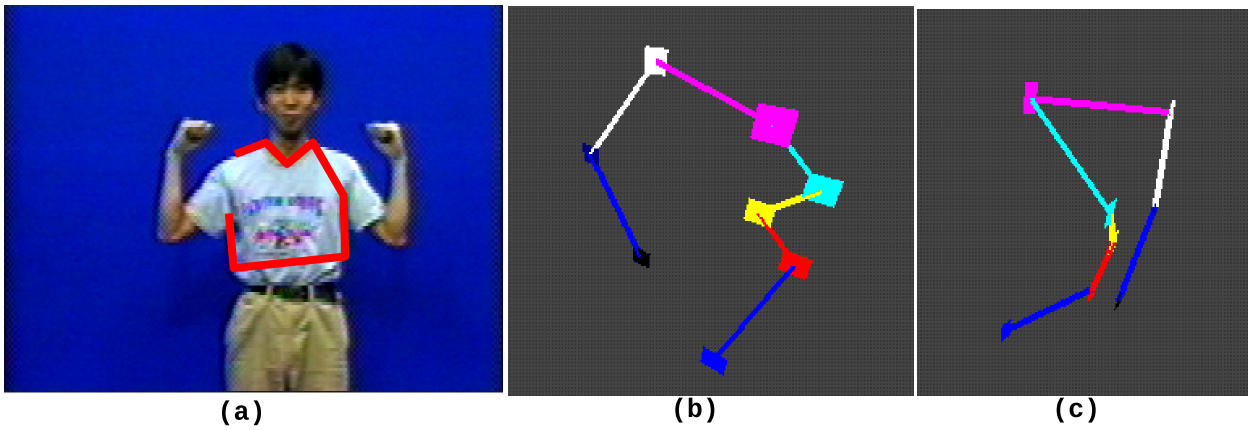


Figure 11: Corresponding pairs are specified using all the three camera images (only one of the three image is shown). The Find_S algorithm is used to obtain the 3D location for every corresponding pair, as shown in (b–c).

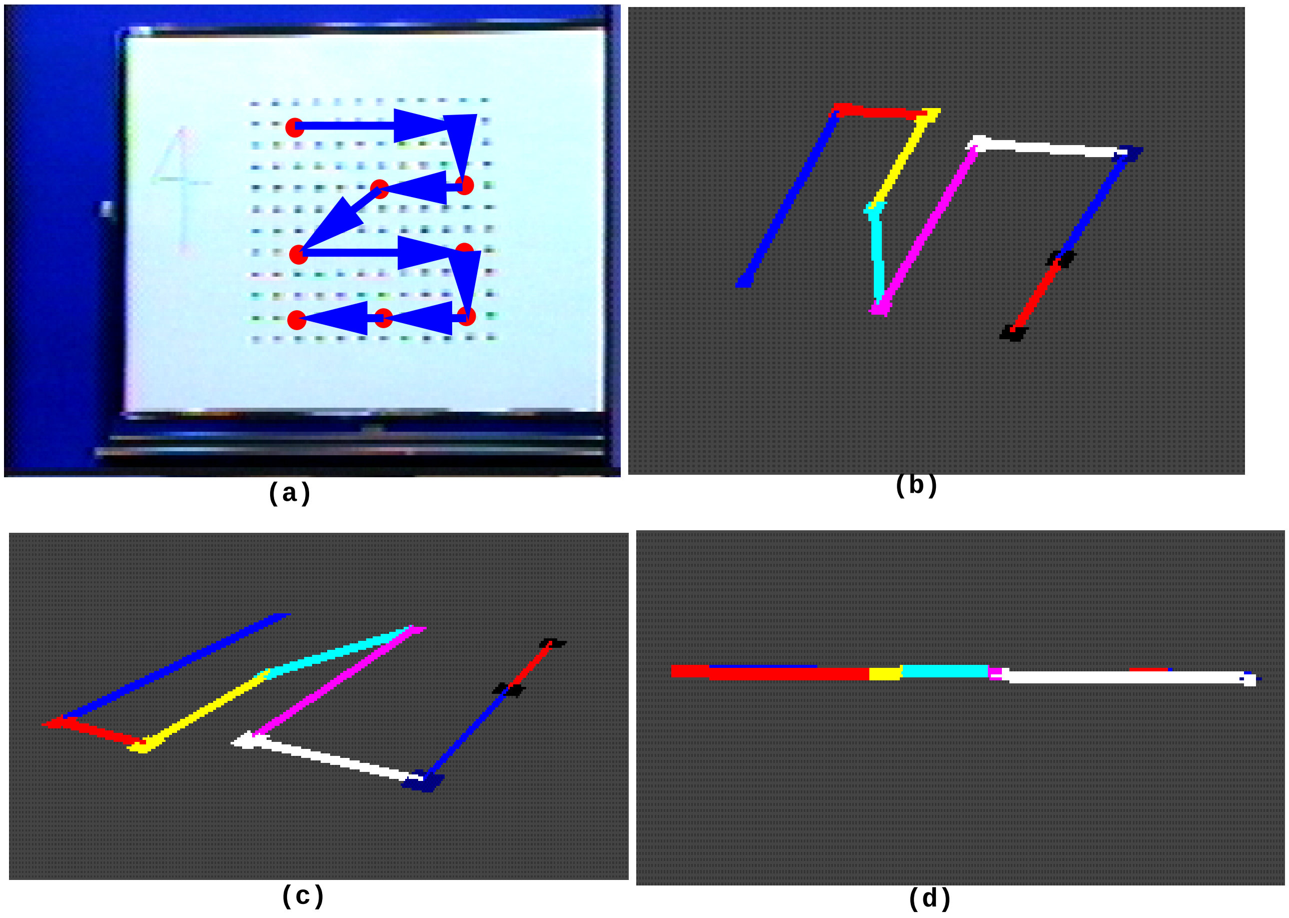


Figure 12: Corresponding points are specified in all the three camera images (only one of them is shown in Figure a). The 3D points estimated by the active space indexing algorithm are shown in Figures b–d.

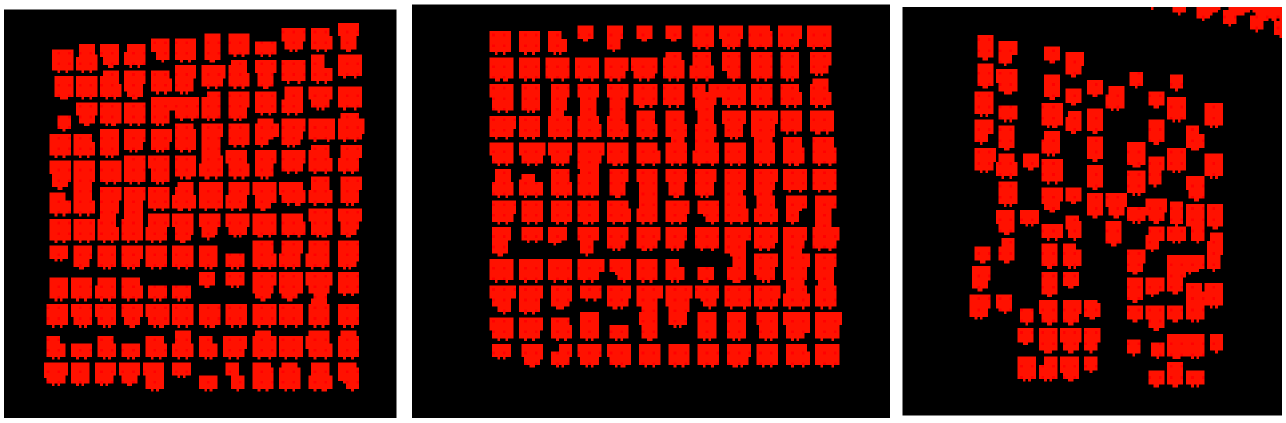


Figure 13: Displaying significant points on a slice using small rectangles. Thresholding is used.

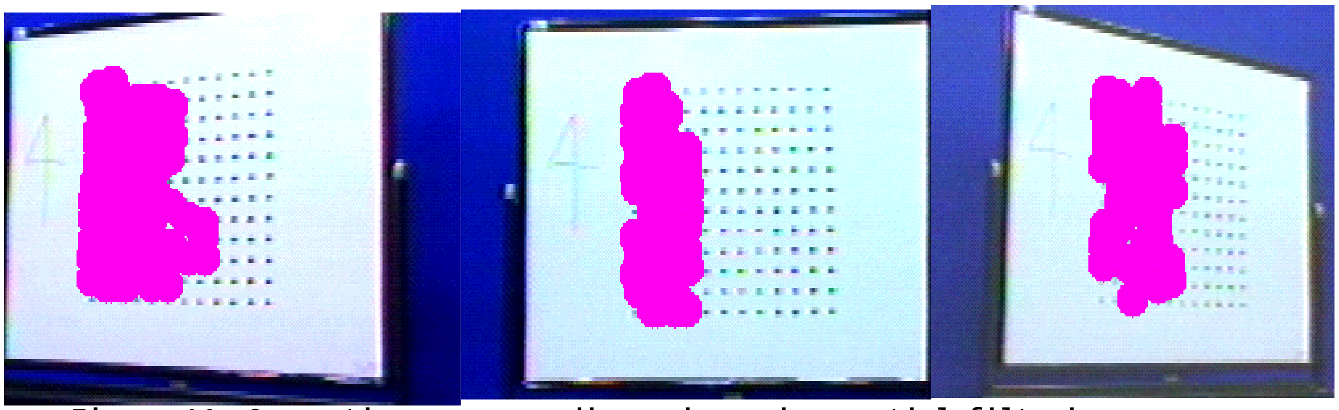


Figure 14: Generating corresponding pairs using spatial filtering

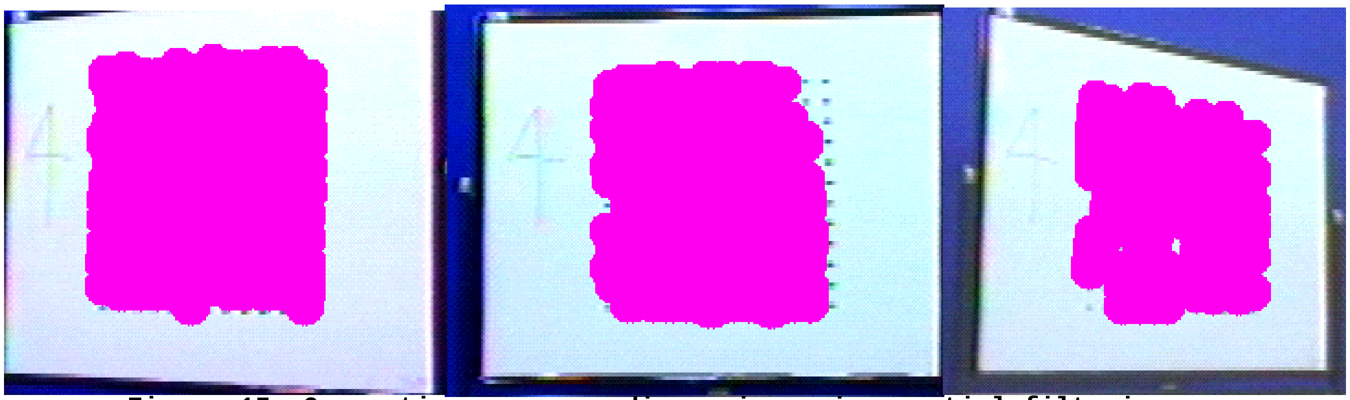


Figure 15: Generating corresponding pairs using spatial filtering

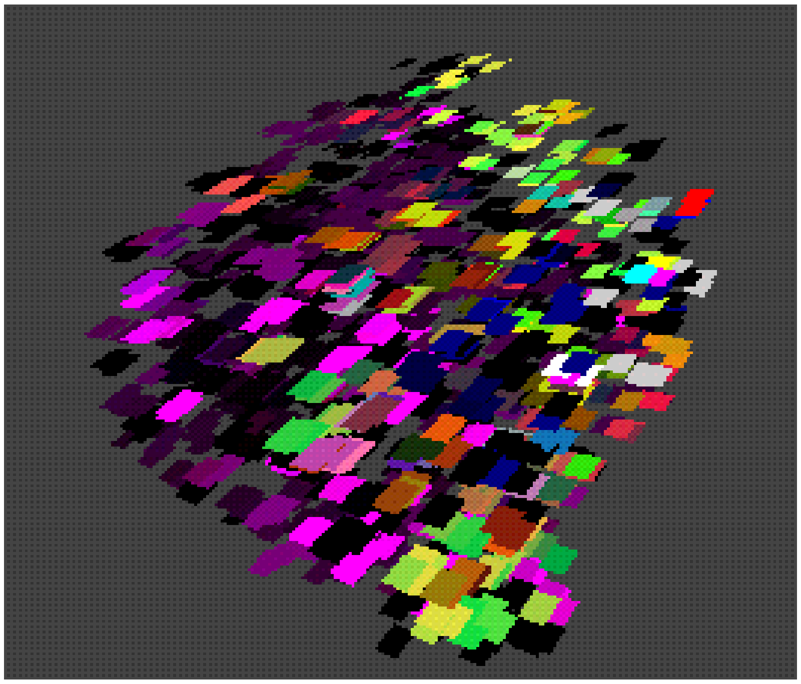


Figure 16: All the corresponding pairs from the eight slices are shown. Each rectangle represents the 3D location of a corresponding pair.

Abstract

Traditionally most camera based position estimation systems use only a few points to calibrate cameras. In this paper, we investigate a novel, and alternate approach for 3D position estimation by using a larger number of points arranged in a 3D grid. We present an implementation of the active space indexing mechanism which uses three cameras. Given the corresponding points in camera images, a precise estimation of the position can be obtained. The active space indexing method can be also used as a spatial filter to eliminate the large number of possible corresponding pairs from consideration. This capability, unique only to the active space indexing method, provides a tractable algorithm to the otherwise intractable situation.

Key Words: Geometric Algorithms, 3D Spatial Data structure, 3D Position Estimation, Camera Calibration for Multiple camera images, Virtual Environments, Computer Vision Techniques for 3D Model building.

1. INTRODUCTION

There are many major benefits to camera-based techniques which make them desirable for virtual environments [1, 3]. One of the main benefit is that participants do not wear any encumbering tracking devices. However, camera based virtual environments must resolve the correspondence problem across multiple camera images. This may involve: (a) Identifying corresponding-pairs across two or more camera images, and then (b) estimating the 3D point for a corresponding-pair. There have been several compromises made to resolve the issue of correspondence over the last several decades. This has led to much interesting research performed in the areas of vision [5, 10], active-vision [6], object-recognition [7], and motion analysis [8, 11] , to name just a few.

Excellent surveys on methods for tracking human participants in virtual environments [25, 24] have appeared recently. So, to avoid unnecessary duplication, we simply ask interested readers to refer them. In this paper, we briefly mention the most recent efforts in this area. Narayana and Kanade [23] use depth (range) images to create visible surface models (VSMs) but has problems with holes and inconsistencies. Moezzi, Katkere, Kuramura, and Jain [12] used color based marching cube algorithm with only limited success; Iwasawa et. al. [19] used silhouette and obtained only approximate results; Haritaoglu, Harwood, and Davis [20] developed an intensity based discrimination to provide simple (called cardboard) human figures; Wren et. al. used blob models [13] in the Pfinder, but their system tracks essentially the 2D-information.

One difficulty which camera based systems face is the variation of color (intensity) values, as they may differ from one camera to another. So, it is very likely that even if we have identified corresponding points in multiple camera images so that they point to the same point in 3D, they may *not* have the same (r,g,b) values. Thus color based information poses severe reliability problems when it is used for determining the correspondence across multiple cameras. Ambiguities abound due to variations of shadows, noisy and blurred images, placement of light sources, occlusion, and the camera covering different areas of 3D-space. Thus, it is highly likely that the projection-pairs belonging to a 3D point may not have similar colors. For example, consider transparent objects, and their corresponding images may not have the color information as the background scenes may be completely different behind the transparent objects. Indeed, correspondence and occlusion still remain grand challenges.

Occlusion creates tremendous ambiguity and erroneous results in possibly every algorithm developed until today. Additional ambiguity also exists when multiple cameras are used as they almost always *never* cover the identical visible areas. As a result a pixel in one image may not have a corresponding pixel existing in another camera-image. The current direction of research has been to find structures from images, for example, identifying a few significant points in [19, 23, 13, 20], and many others, and resolve correspondences among them, once these limited number of points are obtained. Using calibrated cameras, 3D positions can be estimated, and in some cases mapped on to a synthetic human-like actor to represent an avatar of the participant. These approximations can add preciseness, and robustness problems, as estimation of correspondences may not be reliable. Recently, a second research direction has been emerging as mentioned in [24, 25]. Basically, combinations of several tracking methods is suggested for resolving the correspondence problem. Some methods have already been implemented, for example the Constellation(TM) system in [21], and the multi-sensor fusion approach in [22].

We are pursuing a third alternative towards resolving correspondences. It is hoped that the proposed alternative approach provides complimentary support to the other two approaches discussed above, and would lead to the solution of correspondence problem. Our belief is that, for accurate correspondence resolution, we must consider every pixel of related camera-images. As will be explained below, a large number of unique pixel combinations can be generated from just three camera images. Our approach is to simply check all the pixel combinations without extracting any structures. We believe that this method provides the best strategy for resolving correspondence. We also want to develop algorithms which are easily parallelize in future, similar to algorithms in [14]. Since we are using three cameras at this time, we should look at enormity of combinations which we wish to examine for correspondence. Since every possible pixel combination is to be now examined, let us analyze the worst case approach for such an attempt. Let us assume that every image is 640x480, giving us a total of 307200 pixels in one image. Since there are three images, we have $(307200)^3$, that is, $2.8x(10)^{16}$ possible combinations for a brute force method. This worst case is intractable. We have implemented a *tractable* solution using the active space indexing method. In this paper, we show that the major benefit of the active space indexing method is that it can be used as a spatial-filter so that many combinations can be quickly avoided or not be even generated. The new algorithm terminates in a few (4 to 6) minutes on an Onyx-SGI machine, thus providing a tractable solution. Further improvements are expected by implementing the proposed algorithms on specialized hardware.

The active-space indexing method was originally developed for the Scan&Track system [16]. In this paper, only a brief explanation of the active-space indexing method is in Section 2. For details of Scan&Track and other related details please refer [15, 16, 17]. Our motivation to develop the active-space indexing method was to recover the 3D position of corresponding pairs to be as accurate as allowed by any camera based system. We explain a highly accurate method for estimating the 3D position, called the active space indexing method in Section 3. We have obtained robust and accurate 3D position estimates for both the three camera and two camera implementation of the active-space indexing method. In this paper, we will only present the details of our three camera implementations in Section 3. Interested readers should consult [18] paper for the details of the two camera system. In Section 4, we discuss an important application of the active-space indexing method, that of providing a tractable solution for an intractable problem. In particular, we show that the active space indexing method can be used as a spatial filter as it eliminates a large number of corresponding pairs from consideration. Results in Section 5 show that our technique is capable of predicting the 3D position precisely, as well as act as a spatial filter.

2. THE ACTIVE-SPACE INDEXING METHOD

In the three camera implementation for the active-space indexing method, a camera is placed facing a white board. Two other cameras are placed slightly to the left and right of the participant. There is a high correspondence between the three cameras, and the amount of overlap among the three cameras is large. All the white-board placements are in clear focus from all the three cameras. Thus, camera images are minimally noisy and not blurred in our experiments. The camera placement is done before the preprocessing starts, and the active-space or work-volume covered by these camera can be decreased or increased, as necessary.

The active space indexing method uses a large number of 3D-points arranged in grid-fashion, and cover the whole 3D active-space. The method provides robustness against camera distortions as a large number of 3D-points and their projections are available for precise 3D position estimation. The active-space indexing method calculates the precise location of a 3D point given the 3D-point's projections on the multiple images. The significant point extraction algorithm is localized and is based upon the intensity of neighboring eight pixels, so a large number of significant points are identified as potential candidates for correspondence in every cameras. We explain both these algorithm in detail later. Although correspondence resolution still remains our goal, these results are certainly important and necessary steps toward realizing both visual realism and 3D interaction.

2.1. Definitions

Consider Figure 1 where S1, S2, and S3 are the projections of a point on S on the three, left, center and right cameras. We call (S1,S2,S3) a *precise-tuple* when S1, S2, and S3 are the actual projections of a 3D point S on the three cameras planes. This definition is of course generalizable to an n-camera system, but we only used three cameras. When a camera is digitized then approximations occur due to the digitization process itself. The camera-plane is now represented by a set of pixels. Since the camera image contains a limited number of pixels, a variety of projection related distortions are possible. This means that it is possible that the points S1, S2, S3 are a close approximation of a precise-set. So if we start three rays from the left, center, and right camera view points and pass them through S1, S2, S3 respectively, then the rays may not converge or intersect. If (S1,S2,S3) are a close approximation of the actual imprint-set, these rays could be close, but may not still intersect. Now we can define a criteria called the *closeness*. For the three camera case, closeness is defined as the average value of the shortest perpendicular distance between the three ray-pairs. These rays start from the respective view point, and pass through the three points S1, S2, S3 respectively. The *closeness* of zero indicated that the imprint-tuple (S1,S2,S3) is the precise-set of point S. Larger values of closeness indicate that perhaps S1, S2, and S3 are not corresponding pairs. Imprint-tuples (S1,S2,S3) satisfying the closeness criteria are called *corresponding pairs*. It should be mentioned here that in our active space indexing method, we do not use the camera view point to start the rays. Instead, we use the projections of several grid-points to generate the rays. This will be explained in more detail in Section 4. Given the imprint-tuple (S1,S2,S3), the *active-space indexing method* finds the 3D-cell or voxel containing the point S, as well as the precise 3D-location of point S (Section 3.3). These imprint-tuples can be provided manually (Section 4), or can be automatically generated from the camera-images (Section 5).

3. A NOVEL CALIBRATION TECHNIQUE

There have been many attempts to estimate the 3D motion of the participants with a minimal number of points used for camera-calibration [2]. Several examples exist where five or more points are used for creating stereo matching and novel views [10]. Most of the systems which use a minimal number of points are usually under-constrained, in the sense that any error in camera-calibration or camera orientation estimates may result in severe registration and accuracy errors [4, 23].

In implementing the active-space indexing method, we use 3D grid-points, during preprocessing, and create a spatial 3D-data structure using these points. Our motivation to use several 3D grid points is to subdivide the active-space, and use only a few 3D grid-points in the immediate vicinity and surrounding the point S The 3D grid data structure is created during preprocessing, and populates the active-space. This is a clear departure from all other camera-based virtual-environments where a minimal number of calibration points are used. The major advantage of using a large number of points is that the errors due to camera-calibration is minimized. In camera calibration techniques, slight error in calibration may result in erroneous results, especially when the estimated point is farther from the camera-plane. In our active-space indexing method the projection of all the 3D-grid points, as well as their location (indices) in the grid is known, so accurate measurements are available for nearer and farther 3D-voxles. Since these measurements and linear interpolation is used to precisely estimate the 3D location of a point S, the active space indexing method is robust and less erroneous.

3.1. Preprocessing

The algorithm for collecting grid-points from a set of planer-slices for all the three camera images is shown in Figure 2. A grid pattern is used on a white board. Multiple planar white board positions (slices) are obtained as the white-board is physically moved parallel to it's previous position. These slices create a 3D-grid of points covering the whole active-space.

In our implementation, we have used 12 rows and 12 columns grid and a total of eight white-board locations or slices for partitioning the active-space, and creating a 3D grid data structure. The grid-pattern occupies a 55cm by 55cm space on a white-board. Each square of the grid is 5cm by 5cm. The white-board is physically moved and recorded by three cameras at the same time. Eight such recordings result in eight slices for every camera. The inter-slice spacing is 10 cm. The white-board and the grid-pattern on it are visible from all the three cameras. The active-space volume is a cube of 55cm by 55cm by 70cm. The active-space is large enough to track the face and upper body of the participants. A larger active-space can be constructed by using a larger panel. Since 12 rows, 12 columns, and 8 slices are used for partitioning the active-space into 11 by 11 by 7 voxels, the voxel index (i,j,k) is such that $0 \leq i, j \leq 11$, and $0 \leq k \leq 7$. Here i, j, k are called the grid indices. There is one-to-one mapping of the 3D active-space and the 3D-grid. The 55 cm by 55 cm by 70 cm 3D-volume, representing the 3D active-space, is mapped to the 3D-grid, with indices varying from zero to eleven corresponding to 12 rows and columns on slices, and zero to seven corresponding to eight slices. Thus the mapping of active-space to the grid is well defined.

Camera-images for every slice are processed one by one for all the three cameras, and projections of all the 3D-grid points are collected. The algorithms are also provided in Figures 2 and 3. Since camera-images are assumed to be visibly clear and in focus (see Figure 8), errors due to noisy or blurred images are not considered in our experiments. If a grid-point of the slice is incorrectly identified during preprocessing, its effect would be localized, limited to the 3D space corresponding to only those few voxels bordering the incorrectly identified grid-point. All the grid-points for all the three cameras are shown in Figure 9. More details of creating active-space indexing data structure are not explained here as they can be found in [17].

3.2. Finding a 2D-index

For every camera-image, we also collect a set of horizontal and vertical lines, during preprocessing. These lines correspond to the rows and columns of white board planer slices. Now, given the pixel coordinate of a 2D point on a camera-image, we can quickly find the grid-index using these horizontal and vertical lines. First, we check if the point is outside the area defined by the four corner points of the 2D extent specified during preprocessing. If it is inside then we find the x and y grid-indices for the given pixel. Since the grid-lines are specified from left to right, we find two *consecutive* vertical lines p and p+1 such that the point's projection on an image, is on or to the right of line p and is on the left of line p+1. The x-index is then p. A similar algorithm is used to determine the y-index, q, by finding two *consecutive* horizontal lines such that the point is on or above line q, and below line q+1. This algorithm is in Figure 4. Since we only use twelve horizontal and vertical lines per slice, the *valid* grid index is between zero to eleven in both the horizontal and vertical directions. As there are only a fixed number of lines in our implementation, this operation is a fast, constant time O(1) operation.

3.3. 3D-Cell or Voxel Estimation given the Imprint-Set

To estimate the location of a 3D point given its imprint-set (S1,S2,S3), we have implemented the following algorithm. The active-space indexing mechanism finds the 3D location given the imprint-set (S1,S2,S3) triplet. The triplet in our implementation is provided by the user by using mouse-picks on the respective camera-images. Thus to test the correctness of our algorithm, we specify the corresponding pairs.

Given a imprint-set (S1,S2,S3) corresponding to a 3D-point S, we perform the following for the three camera-images in *every slice*: For the left camera-image, let the 2D index be denoted by I1 with x and y indices to be $I1_x$ and $I1_y$, respectively. Similarly the indices for the center and right camera-images would be denoted by I2 and I3, respectively. For every geometric-imprint (S1,S2,S3), we collect I1, I2, and I3 points for every slice. For discussion, we assume that point S is between slice k and k+1. The three grid indices I1, I2, and I3 define a triangle on every slice. Simple ray-optics suggests that the area would be decreasing as the rays converge at point S and then starts increasing as the rays diverge. We have implemented a linear search algorithm to determine the slice with the minimum area. Since we only have eight slices in our experiments, linear searching is a constant time operation[1] to find the two slices k and k+1. Let k be the left slice with the area A_L, and k+1 the right slice, with area A_R as the rays diverge. The 3D-cell index of point S would be (i, j, k) where i=$I1_x$ and j=$I1_y$ in our implementation. Here i could also be an average of x-indices of I1, I2, and I3 for slice k. Similarly j could also be an average of y-indices of I1, I2, and I3 for slice k. So (i,j,k) identifies the 3D-cell or voxel containing the point S. Since the *3D-cell index* of a point in active-space can be determined using this method, we call this method the *active-space indexing method.* This is step 2 of the algorithm presented in Figure 5. Results of this algorithm are also shown in Figure

[1]A binary search on the slices would be more appropriate if a large number of slices were used to scan the active-space.

10. Figure 10d shows the 3D voxel identified when the points S1, S2, and S3 are specified in three camera images as shown in Figure 10a-c.

4. FINDING THE EXACT 3D-POSITION

Once the voxel index (i,j,k) is identified from the given imprint-set, we use linear-interpolation to create three lines, one for each camera-image. This is the step 3 of the algorithm in Figure 5. We first explain how to find this line for the left camera. For finding the line, we consider two consecutive slices k and k+1. We know the pixel-coordinate of point S1 (it is given to us), we also know the pixel coordinates of grid indices (i, j), (i+1, j), (i, j+1), and (i+1, j+1) on slice k, as these pixel coordinates are collected during preprocessing. Note that the pixel coordinates of point S1 must be inside the polygon created by four pixel-points representing the grid indices (i, j), (i+1, j), (i, j+1), and (i+1, j+1) on slice k. A linear interpolation algorithm is now used to calculate the coordinates of point S1, in terms of grid indices on slice k. The four points surrounding the point S1, as well as their indices are collected during preprocessing. Linear interpolation provide the 3D coordinate of the projection on that slice. This 3D coordinate is of the form (iReal, jReal, k) where $i \leq iReal \leq (i+1)$ and $j \leq jReal \leq (j+1)$. Similarly we find another point on slice k+1 for the left camera. Joining these two points on slice k and k+1 provides a line for the left camera. Similar calculations are used for the center and right cameras to obtain a total of three lines, one each, for the three cameras. The intersection of these three lines provides the precise location of the point S. Since the calculations are based upon the correctness of the imprint set (S1,S2,S3), we have tested this algorithm by specifying a large number of imprint sets. Results are very precise and close to the second decimal point when compared for the known 3D point. Figure 11a-c shows the result of our implementation using 8 imprint-sets in all the three camera images. We have only shown the center camera image in Figure 11a. The 3D-points shown in Figure 11b-c correspond exactly to the shape in Figure 11a. Note that 3D-points in Figure 11b and 11c do not lie on one plane and correctly correspond to the specified body points of the participant in Figure 11a. Figure 12a shows a planer slice and the associated grids. We specified nine points on all the three camera-images. The location and shape resulting from these points is shown in Figure 12a in the center camera-image. We have correctly obtained a planar 3D-shape as shown in Figure 12b-d. Note the planar shape of the 3D form in Figure 12b-d. In the large number of experiments conducted we obtained a precise (correct up to two decimal points) 3D-position of point S given its imprint-set (S1,S2,S3). This results are also independently verified in our two camera system [18].

5. SPATIAL FILTERING

We have implemented a *closeness* criteria (see Section 2.1) to identify imprint-sets. The value of the *closeness* parameter can be set to any real number, e.g. .01 in the results presented in this paper. A value of .01 means that unless the lines are within .01 distance apart S1, S2, and S3 are not the projection-pairs of some 3D point S. This allows filtering of erroneous pairing of pixels S1, S2, and S3 which indeed are not the projections of the same point S, yet are identified by the filtering algorithm as possible candidates for checking correspondence.

In brute force approach, we generated all possible corresponding pairs using every pixel of the three camera-images as a possible candidate for correspondence. As explained earlier in Section 1, for a 640x480 image, the number of possible combinations are virtually intractable. This number is more than $2.8\text{x}(10)^{16}$. To check if these many tuples satisfy the closeness criteria was intractable, in the sense that, for even the multi-processor Onyx machine, the process did not finish even after several (five) hours of running. To make this brute force approach tractable, we used the active-space indexing method as a spatial filter. This is explained in the following sections.

5.1. Preprocessing for Spatial Filtering

The active-space indexing method partitions the active-space created by the 3D-slices into 3D voxels creating a convenient 3D-spatial separation. During preprocessing, we create a list of 2D-indices for every pixel in the left, right, and center images. This list is conveniently created using the 2D-index algorithm of Section 3.2. For every pixel, since we store the 2D-index for every slice, we actually have the list of 3D-voxels which projects to cover that pixel.

5.2. Generating corresponding pairs

To avoid generating all the corresponding pairs possible, we implemented a simple algorithm which looks at every pixel of an image to determine if that pixel could be a tip or an end-point of cylindrical human limbs. This determination is based upon the color of all the eight pixels surrounding that pixel, and thresholding, to determine if this pixel can be an end point. See also step 2, Figure 6.

This *localized* decision is less error-prone as errors due to color fluctuation across the image are minimized in comparison to our earlier efforts in [16]. The algorithm is better in comparison to our earlier efforts as now we do not have to extract the curve, and trace them as in [15]. The algorithm could also be easily implemented in hardware. Figure 13 shows the result of this algorithm of Figure 6 which is applied to slice number four. This process takes a small amount of time, typically, 3-4 seconds, on an Onyx machine.

5.3. Spatial Filtering Results

Procedure SpatialMarking (Figure 6) is used to collect the end-points from cameras 1,2, and 3 for every voxel as follows:

(a) For every pixel identified as a significant point (Step2, Figure 6), use the 2D-index for every slice (these 2D indices are generated during preprocessing) to mark all the active-space 3D-voxels which cover this pixel (steps 3 and 4, Figure 6) (b) Perform the above for all the significant (pixel) points on all the three camera-images, maintain a list in every 3D-voxel for marking by left, right, and center cameras (Step 1, Figure 6).

The algorithm for generating all combinations of corresponding pairs (S1,S2,S3) is given in Figure 7. Basically, if a 3D-voxel is marked by pixel S1, S2, and S3 in the left, center, and right cameras, respectively, then (S1,S2,S3) is used as an imprint set, and we calculate the 3D-point S given the imprint-set (S1,S2,S3) using the Find_S (step 8, Figure 7) algorithm. Once Find_S specifies a 3D point corresponding to (S1,S2,S3), the 3D point is displayed. All the valid corresponding pairs corresponding to all the eight slices, are generated, and displayed, as shown in Figure 16. Figure 16 shows the list of all the valid points which satisfied the criteria of closeness of point .01.

If a 3D-voxel is marked by $p1$ number of pixels from the left camera, $p2$ number of pixels from the center camera, and $p3$ number of pixels from the right camera, then the number of unique imprint-sets will be $p1 \times p2 \times p3$. All of those imprint sets satisfying the *closeness* criteria, as explained in Section 2.1 and also in Find_S, are collected as valid 3D-points. Figure 14 and 15 show the filtering algorithm working for all the three cameras in a variety of stages. Our results show that an extremely small percentage of corresponding pairs are being considered by using this filter in comparison to the worst case approach. This is because the active space has been divided into $11 \times 11 \times 7$ very small compartments.

Finally, we should point out that the filtering technique is effective in reducing the number of corresponding pairs which are to be generated. These corresponding pairs are spatially related. Multiplicity still exist, in the sense, that it is possible that many points on several slices satisfy the closeness criteria, and are displayed. Recovering the 3D surface from three cameras images, that is solving the correspondence problem, still remains a challenge.

6. CONCLUSIONS AND FURTHER RESEARCH

The active space indexing mechanism uses only those grid-points for position estimation which are spatially close to the actual 3D location of the point. This provides localization and could provide robustness against camera distortions. We are able to also provide an accurate and precise position estimation, once the corresponding pair is identified.

Our spatial filtering algorithm terminates quickly within a few minutes. This is a huge computational saving in comparison to the intractable worst case algorithm of generating corresponding pairs. To the best of our knowledge, there does not exist another camera-based system which employs such a spatial data structure to reduce the possible number of corresponding pairs.

Although we have drastically reduced the number of corresponding pairs by using spatial filtering, much work remains for recovering the precise geometry of the projected surface, especially under noisy and blurred conditions. We need to resolve the correspondence problem. In particular, *unique* corresponding pairs must be determined to extract the 3D surface. This problem, as we discussed in Section 1, is still a grand challenge. However, the spatial filtering provided by the active space method has drastically reduced the number of corresponding pairs in comparison to the worst case. This, we feel, is the major contribution of techniques presented in this paper.

ACKNOWLEDGMENTS

The first author would like to thank Dr. R. Nakatsu, President of ATR Media Integration & Communications (MIC) Research Laboratories, for inviting him to ATR. Thanks are also due to Mr. Fujimoto, I Fermin, and I Painuli, researchers in Department One for a variety of discussions and help.

References

[1] K Meyer, HL Applewhite, and FA Biocca. *A Survey of Position Trackers.* PRESENCE, 1(2), pp 173-200, MIT Press, Spring 1992.

[2] RF Rashid. *Towards a system for the Interpretation of Moving Light Displays.* IEEE Transaction on PAMI, 2(6) pp 574-581, 1980.

[3] MW Krueger. *Artificial Reality II.* Addison Wesley Publishing Company, Reading, MA, pp 1-277, 1991.

[4] A State, G Hirota, DT Chen, WF Garrett, MA Livingston. *Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking.* Proceedings of SIGGRAPH 1996, pp 429-438, 1996.

[5] O Faugeras. *Three-Dimensional Computer Vision: A geometric Viewpoint.* MIT Press, Cambridge, Massachussets, 1996.

[6] Blake A, Yuille A eds. Active Vision. The MIT Press Cambridge, MA, pp. 1-303, 1992.

[7] W. Eric and L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, Cambridge, MA, 1990.

[8] S Maybank. *Theory of Reconstruction from Image Motion.* Springer-Verlag, 1993.

[9] J Serra. *Image Analysis and Mathematical Morphology.* Academic Press, vols 1 and 2, 1988.

[10] Proceedings of the Third International Conference on Automatic Face and Gesture Recognition, pp 1-384, October 14-16 1998, Killington, Vermont, USA, IEEE Computer Society Press, 1996.

[11] Segan J and Pingali SG. A camera based system for tracking people in real time. Proceedings of ICPR96, IEEE Computer Society, pp 63-68, 1996.

[12] Moezzi S, Katakere A, Kuramura DY, and Jain R. *Immerssive Video.* Proceedings of IEEE VRAIS 96, IEEE Computer Society Press, Los Alamitos, pp 17-24, Santa Clara, CA, 1996.

[13] C Wren, A Azarbayejani, T Darrell, A Pentland. *Pfinder: Real-Time Tracking of the Human Body.* International Conference on Automatic Face and Gesture Recognition, 1996, pp 51-56, 1996.

[14] Kendall P Jr, Duff M. Modern Cellular Automata: Theory and Applications. Plenum Press, NY. Chapters 1, 4-9, 11, 1984.

[15] S Semwal and J Ohya. *Geometric-Imprints: A Significant Points Extraction Method for the Scan&Track Virtual Environment*, Proceedings of the Third International Conference on Automatic Face and Gesture Recognition Conference, April 14-16, 1998, Nara, Japan, pp 480-48, IEEE Computer Society, 1998.

[16] S Semwal and J Ohya. *The Scan&Track Virtual Environment.* Proceedings of the first International Conference on Virtual Worlds 1998 (VW98), Paris, France, July 1st-3rd, 1998, Lecture Notes in Computer Science, LNCS/AI1434, Springer Verlag, pp. 63-80, 1998.

[17] S Semwal. *Complexity issues in Virtual Environments.* Proceedings of the International conference on Artificial Tele-Existence, Tokyo, Japan, pp. 27-32 (invited paper), December 1998.

[18] I Fermin, S Semwal and J Ohya. *Indexing method for Three Dimensional Position Estimation.* Accepted for publication in the IEICE transactions and Information systems, pp. 1-8, 1999.

[19] I Shoichiro. J Ohya, K Takahashi, T Sakaguchi, K Ebihara, S Morishima. *Real-time, 3D Estimation of Human Body Postures from Trinocular images*, accepted for publication in IEEE International conference on Modeling People (mPeople), September 20, 1999, Greece.

[20] I Haritaoglu, D Harwood, LS Davis. *W4: Who? When? Where? What?A Real Time System for Detecting and Tracking People*, the Third IEEE International Conference on Automatic Face and Gesture Recognition April 14-16, 1998, Nara, Japan, IEEE Computer Society.

[21] Foxlin E, Harrington M, Pfeifer G. *Constellation (TM): A Wide-Range Wireless Motion Tracking for Augmented Reality and Virtual Set Applications.* Computer Graphics Proceedings, Annual Conference Series (Siggraph 1998), 371-378, 1998.

[22] G Welch and G Bishop. *SCAAT: Incremental Tracking with Incomplete Information.* Computer Graphics Proceedings, Annual Conference Series (Siggraph 1998) 371-378, 1997.

[23] PJ Narayanan, PW Rander, and T Kanade. *Virtualized Reality system*, ICCV98 Conference Proceedings Bombay, pp. 3-10, 1998.

[24] D Gavrila. *The Visual Analysis of Human Movement: A Survey.* Computer Vision and Image Understanding, 73(1):82-98, 1999.

[25] J Aggarwal and Q Cai. *Human Motion Analysis: A Review.* Computer Vision and Image Understanding, 73(3):428-440, 1999.