Homography Based Parallel Volume Intersection: Toward Real-Time Volume Reconstruction Using Active Cameras

Toshikazu Wada, Xiaojun Wu, Shogo Tokai, Takashi Matsuyama
Department of Intelligence Science and Technology,
Graduate School of Informatics,
Kyoto University
Yoshida Hon-machi, Sakyo-ku, Kyoto, 606-8501, JAPAN
twada@i.kyoto-u.ac.jp, wxj@kuee.kyoto-u.ac.jp, tokai@kuee.kyoto-u.ac.jp, tm@i.kyoto-u.ac.jp

Abstract

Silhouette volume intersection is one of the most popular ideas for reconstructing the 3D volume of an object from multi-viewpoint silhouette images. This paper presents a novel parallel volume intersection method based on *plane-to-plane homography* for real-time 3D volume reconstruction using active cameras. This paper mainly focuses on the acceleration of back-projection from silhouette images to 3D space without using any sophisticated software technique, such as octree volume representation, or look-up table based projection acceleration. Also this paper presents a parallel intersection method of projected silhouette images. From the preliminary experimental results we estimate near frame-rate volume reconstruction for a life-sized mannequin can be achieved at 3cm spatial resolution on our PC cluster system.

1. Introduction

Motion data of the human body is often used to create realistic 3D CG animations. As well, human motion data can be utilized for telerobotics, ergonomics, biomechanics and sport performance analysis. Usually, motions are captured by commercial motion capture equipments, which are designed to track key points of the articulated object, such as joints. This tracking is realized by either magnetic or optical point measurements. Since magnetic point measurement uses bulky sensors on key points and the magnetic field can be affected by steel, optical tracking is widely used. Optical tracking uses radiational or reflective optical markers on key points. Their lights are observed by surrounding cameras and their 3-D positions are computed from these observation results. The limitations of current optical motion capture systems can be summarized as:

- **Optical markers may be occluded by the object (self occlusion)**: Most of the systems employ 3D joint model matching which compensates for the marker occlusions on some observed images. But, if a marker disappears from all observed images, the position of the point cannot be obtained.
- **3D shape information cannot be obtained**: Current motion capture systems only provide the positions of sparse points without 3D shape information which is essential to measure deformable objects.
- **The working space is limited to a small are**a. This is because each camera has limited spatial resolution and fixed camera parameters.

These limitations are removed by introducing the following techniques:

1. **3D volume reconstruction**: By matching a 3D object volume model with the obtained 3D volume, we can avoid the self-occlusion problem. As well, we can measure the object deformation from the volume information.
2. **Active camera control** : By changing the view direction of the camera, we can keep capturing the moving object without reducing the spatial resolution of the cameras. As well, by changing the zoom, spatial resolution can be maximized for a particular moving object.

In the case of using multiple cameras surrounding an object, *silhouette volume intersection* [1] [2] [6] [7] [9] [10] is the most popular idea for computing 3D volume of the object (Figure 1). This idea is based on *silhouette constraints* that each 2D silhouette of an object constrains the object inside the volume (frustum) produced by back-projecting the silhouette from the corresponding viewpoint. Therefore, by

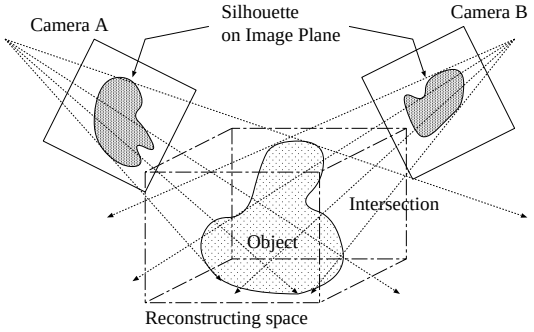


Figure 1. Silhouette Volume Intersection.

intersecting the frusta for all silhouettes, an approximation of the object volume is obtained. This is called *visual hull* [5] which constrains the object its inside. Recently, this idea is further extended using photometric information to reconstruct more accurate shapes [4].

From an algorithmic viewpoint, volume reconstruction methods based on this idea can be classified into two categories:

1. **Space carving method (SCM)** which cuts off those 3D points projected to the outside of at least one silhouette.

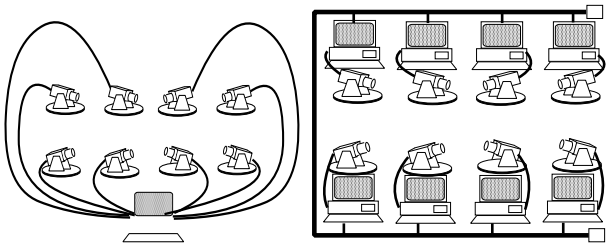
2. **Volume intersection method (VIM)** which computes intersections of all back-projected frusta of silhouettes.

SCM can be regarded as a verification based method using silhouette constraints, and VIM is the straightforward realization of the basic idea. Since SCM requires less memory than VIM, most of the previous methods are classified into SCM.

As for the camera control suitable for the volume reconstruction, it requires 3D object information. This is because the moving object must be covered by the common observing area of all cameras. That is, like a moving cage, the common observing area must follow and always cover the entire body of the moving object. For this coordinated camera control, the 3D location and shape of the object are required. Therefore, the volume reconstruction speed is a crucial problem for the camera control. Ideally, it should be done in real-time, i.e., short latency, and near frame-rate volume reconstruction.

The reconstruction speed is ruled by the computation scheme as follows.

As shown in Figure2 (a), if a high-speed computer achieves frame-rate volume reconstruction for a certain number of cameras, this speed will slow down as the number of cameras increases. That is, single process 3D volume reconstruction does not have scalability. On the other hand,



(a) single processing (b) parallel processing

Figure 2. Computation Schemes for volume Reconstruction

```
foreach v ∈ {voxel} begin
  v :=' occupied'
  foreach c ∈ {camera} begin
      project v to image plane of c
      if projected point isn't in the silhouette then
          v :=' empty'
          goto ENDINNERLOOP
      endif
  end
ENDINNERLOOP:
end
```

Figure 3. Basic Algorithm for SCM. $\{voxel\}$: set of all voxels, $\{camera\}$: set of all cameras, $'occupied'$: value of occupied voxel, $'empty'$: value of non-occupied voxel.

network connected computers, each of which has an active camera(Figure2 (b)), can sustain its volume reconstruction speed for numbers of cameras, because both of the image capture and silhouette image generation are done by individual computers in parallel, and the volume reconstruction can also be parallelized[1]. That is, parallel processing is the most promising approach to achieve a scalable real-time 3D volume reconstruction.

In this paper, we propose a parallel volume reconstruction method based on silhouette volume intersection. In the following sections, we describe basic method for parallel volume reconstruction, acceleration of back-projection, parallelization of intersection, and experimental results.

2. Basic algorithm

To realize high-speed parallel volume reconstruction, we first have to select the basic algorithm, i.e., SCM or VIM.

[1] Although the latency grows up as the number of images increases.

```
{voxel} := {'occupied'}
foreach c ∈ {camera} begin
   {F(c)} := {'empty'}
   foreach p ∈ {silhouette(c)} begin
      project p to 3D space and generate a ray b(p, c)
      {F(c)} := {F(c)} ⋃ r(p, c)
   end
   {voxel} := {voxel} ⋂ {F(c)}
end
```

Figure 4. Basic Algorithm for VIM:
$\{voxel\}$: set of voxels, $\{silhouette(c)\}$: set of silhouette pixels observed by camera c, p: silhouette pixel, $\{camera\}$: set of all cameras, $\{F(c)\}$: frustum generated by back projection from silhouette observed by camera c, $b(p, c)$: a 3D ray passing both p and the optical center of c.

The basic algorithms of SCM and VIM are shown in Figure3 and 4, respectively. In SCM, the outermost loop parameter is *voxel*, but in VIM, it is *camera*, i.e., computer in our computation scheme. This means VIM is suitable for parallel processing, because the back-projection process can be divided into small
processes back-projecting each silhouette to 3D space which run on each computer without referring to other silhouettes. Also in VIM, an intersection process can be decomposed into pairwise intersections. These properties are suitable for parallel processing.

The drawback of basic VIM algorithm is the large memory required to represent the frustum $\{F(c)\}$. There are two types of solutions for this problem:

1. Employ octree volume representation of frusta.
2. Divide a frustum into small subsets and compute VIM for these subsets sequentially.

The former is one of the most effective approach, because it reduces the size of 3D volume representation, as well as accelerates intersection operation for ordinary objects. Furthermore, there is an accelerated octree generation algorithm [10]. However, since it is optimized for binary silhouettes, there is no room left to extend the binary silhouette images to gray scale or color images[4]. Also, for some sparse objects such as a skeleton, the octree representation may not be compact. Because of these reasons, we employ the latter approach in this paper.

By decomposing the 3D space $\{voxel\}$ into smaller disjoint sub-spaces $\{\{v\}\}$, e.g., planes, boxes, etc., we obtain an algorithm as shown in Figure 5. In this modified algorithm of VIM, the outermost loop parameter becomes the subset $\{v\}$ of voxels. However, back projection can still be parallelized.

```
{voxel} := {'empty'}
foreach {v} ∈ {{v}} begin
   {v} := {'occupied'}
   foreach c ∈ {camera} begin
      {F(c)}_v := {'empty'}
      foreach p ∈ {silhouette(c)} begin
         compute projection b_v(p, c) from p to {v}
         {F(c)}_v := {F(c)}_v ⋃ b_v(p, c)
      end
      {v} := {v} ⋂ {F_v(c)}
   end
   {voxel} := {voxel} ⋃ {v}
end
```

Figure 5. Modified Algorithm for VIM: $\{\{v\}\}$: disjoint decomposition of $\{voxel\}$, $\{v\}$: subset of voxels, $\{F(c)\}_v = \{F(c)\} \bigcap \{v\}$, $b_v(p, c) = b(p, c) \bigcap \{v\}$.

From this algorithm, the basic operations for VIM can be classified into two categories:

1. Back-projection
2. Intersection of back-projected silhouettes

In the following sections, detailed acceleration methods for these operations are discussed.

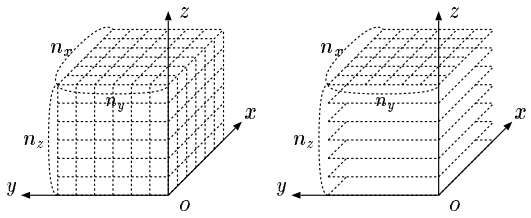
3. Back-projection accelerations

Back-projection is the most expensive computation in VIM, because it requires a considerable arithmetic operations. There may be three approaches for the acceleration:

1. Use a lookup table embedded in voxel space representing the projection points on 2D image plane.
2. Use special hardware for projection.
3. Use accelerated algorithms for perspective projection.

In our problem, the first approach cannot be employed, because the camera action changes the relationship between voxels and pixels. The second is a promising way to achieve a certain acceleration, but to design the hardware we first fix the algorithm. Then, we take the last approach in this paper.

As for the disjoint decomposition of 3D space $\{\{v\}\}$ in the modified VIM algorithm, we employ parallel plane decomposition as shown in Figure6. This is because plane-to-plane perspective projection (*homography*) [8] is computationally less expensive than general perspective projection:



(a) (b)

Figure 6. 3D space representations: (a) 3D voxel space, (b) Disjoint decomposition into parallel planes.

- General perspective projection from 3D point (X, Y, Z) to 2D point $(x_1/x_3,\ x_2/x_3)$ can be represented by the following equation:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (1)$$

  This transformation requires **9** additions, **9** multiplications, and **2** divisions.

- In the case of homography, i.e. the source 3D point is constrained on a 2D plane, the projection equation is simplified to the following equation:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \quad (2)$$

  This requires **6** additions, **6** multiplications, and **2** divisions per point.

As described above, homography is faster than general perspective projection. In the following sections, we show further acceleration methods for this homography based VIM.

3.1. Linear-wise homography

To reduce the number of arithmetic operations for homography, we can exploit the following property (Figure 7):

Property 1 *For any combination of two planes A and B, there exists at least one set of planes $\{C\}$ whose every member C satisfies that two intersection lines $A \bigcap C$ and $B \bigcap C$ are parallel and C involves a fixed line P passing a point o.*

Proof

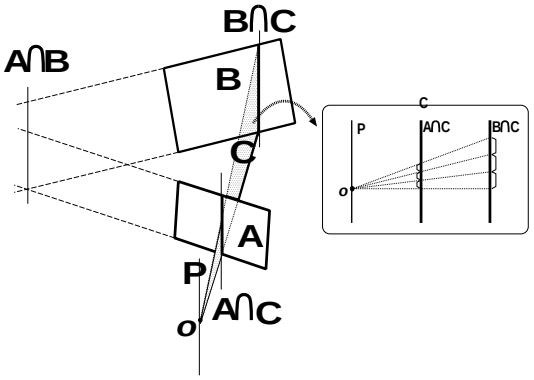


Figure 7. Basic idea of linear-wise homography.

- **Planes A and B are not parallel**, there exists a intersection line $A \bigcap B$. The planes A and B can be decomposed into two disjoint sets of parallel lines $\{L_A\}$ and $\{L_B\}$, both of them are parallel to $A \bigcap B$. Let a line P passing the point o be parallel to $A \bigcap B$. A plane C involving line P makes two intersection lines $A \bigcap C$ and $B \bigcap C$.
  Since, plane C involves P, and P is parallel to $A \bigcap B$, the plane C is also parallel to $A \bigcap B$. Then, the plane C can also be decomposed into disjoint set of parallel lines $\{L_C\}$, whose members are parallel to $A \bigcap B$. Hence, all parallel line decompositions $\{L_A\}$, $\{L_B\}$, and $\{L_C\}$ consist of lines parallel to $A \bigcap B$. Any two different lines picked up from $\{L_A\} \bigcup \{L_B\} \bigcup \{L_C\}$ never intersect, because all these lines are parallel. That is, $A \bigcap C = \{L_A\} \bigcap \{L_C\} = L_{AC}$,and $B \bigcap C = \{L_B\} \bigcap \{L_C\} = L_{BC}$. Since $L_{AC}, L_{BC} \in \{L_C\}$, $A \bigcap C$ and $B \bigcap C$ are parallel.

- **Planes A and B are parallel**, any plane C which is not parallel to these planes makes two intersection lines $A \bigcap C$ and $B \bigcap C$, both of them are obviously parallel. Hence, $\{C\}$ can be defined for any line P passing the point o.

Q.E.D.

The homography between two planes A and B can be decomposed into homographies between 1D lines $A \bigcap C$ and $B \bigcap C$ ($C \in \{C\}$) by regarding the point o as the center of perspective projection. Property 1 guarantees that these two lines can be parallel, if we select the line P properly. Homography between two parallel lines can be represented as 1D scaling. This can be done by linear scanning: starting from two corresponding points $\overrightarrow{x_0}$ and $\overrightarrow{X_0}$, subsequent corresponding points on these lines are computed by adding constant 2D vectors $\overrightarrow{\delta}$ and $\overrightarrow{\Delta}$ to these points:

$$\overrightarrow{x_{i+1}} = \overrightarrow{x_i} + \overrightarrow{\delta}, \quad \overrightarrow{X_{i+1}} = \overrightarrow{X_i} + \overrightarrow{\Delta}. \quad (3)$$

This means **4** additions are enough to compute the homography of a point. But, there are initialization overheads to compute (1) starting point pair $\overrightarrow{x_0}$ and $\overrightarrow{X_0}$, and (2) scaling coefficients. These overheads are equivalent to computing two homographies, two vector subtractions, two scalar divisions per each line pair. In the case of

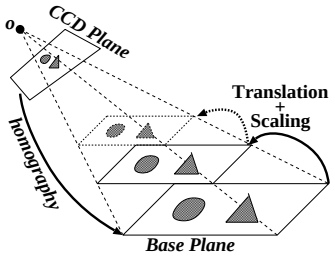


Figure 8. Plane-wise homography: Homography between parallel planes can be done by scaling and translation of a 2D image on source plane.

homography to a plane consists of $n_x \times n_y$ elements, the overheads per point are estimated as: $16/\sqrt{n_x n_y}$ additions, $12/\sqrt{n_x n_y}$ multiplications, and $6/\sqrt{n_x n_y}$ divisions, where $\sqrt{n_x n_y}$ represents the estimated number of lines on the plane.

3.2. Plane-wise homography

As discussed above, linear-wise homography accelerates general homography between non-parallel planes. Here we show the homography can further be accelerated by using the special property between *parallel planes* [3] .

As shown in Figure 8, homography between two parallel planes is simplified to 2D isotropic scaling and translation:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = s \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{4}$$

where s represents the scaling and (t_x, t_y) the translation vector. Equation(4) shows that this transformation requires **2** additions and **2** multiplications per point without any overhead. By using linear scanning employed in linear-wise homography, the number of arithmetic operations can be further reduced. In this case, since every parallel line decomposition of silhouettes can be used for homography, the raster scanning can also be used as:

$$\overrightarrow{x_{i+1}} = \overrightarrow{x_i} + (s_x, 0), \quad \overrightarrow{X_{i+1}} = \overrightarrow{X_i} + (S_X, 0). \tag{5}$$

This requires only **2** additions with overheads: $2/\sqrt{n_x \times n_y}$ additions and $2/\sqrt{n_x \times n_y}$ multiplications per point.

This transformation is a pure 2D geometric transformation. This means that 2D image processing hardware can be employed to accelerate this process.

Since this acceleration method can be applied only to parallel planes, our strategy is to 1) perform linear-wise homography from the image plane to the *base-plane* placed at the bottom of the voxel space, 2) then apply plane-wise homographies from the base-plane silhouette to parallel planes.

3.3. Comparison

If the 3D voxel space consists of $n_v = n_x \times n_y \times n_z$ voxels, the total arithmetic operations can be estimated as follows[2]:

[2]Because of the shape of frustum, the number of arithmetic operations may be reduced for higher plane. But, it is ignored in this estimation, because it depends on the camera location.

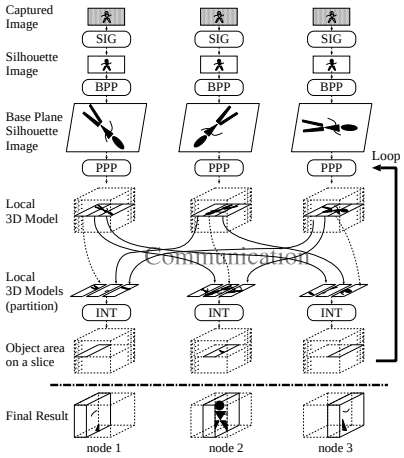


Figure 9. A process flow of the local silhouette division method: SIG: silhouette image generation, BPP: base plane projection, PPP:parallel plane projection, INT: intersection.

- **General Perspective Projection:** $9n_v$ additions, $9n_v$ multiplications, and $2n_v$ divisions.
- **Normal homography:** $6n_v$ additions, $6n_v$ multiplications, and $2n_v$ divisions.
- **Linear-wise and Plane-wise homography:**
  Linear-wise homography: $4n_x n_y + 16\sqrt{n_x n_y}$ additions, $12\sqrt{n_x n_y}$ multiplications, and $6\sqrt{n_x n_y}$ divisions.
  Plane-wise homography: $(2n_x n_y + 2\sqrt{n_x n_y}) \times (n_z - 1)$ additions, and $2\sqrt{n_x n_y}(n_z - 1)$ multiplications.
  In all total,
  $2n_v + 2n_x n_y + \sqrt{n_x n_y}(2(n_z - 1) + 16)$ additions,
  $\sqrt{n_x n_y}(2(n_z - 1) + 12)$ multiplications,
  and $6\sqrt{n_x n_y}$ divisions.

Some examples of arithmetic operations are listed in Table 1. From these results, it is clear that the combination of linear-wise and plane-wise homographies is the most effective method for back-projection.

4. Parallel intersection

Another issue in parallelizing VIM is the intersection of binary silhouette images projected on each plane, which requires communication among computers.

The improved VIM algorithm shown in Figure 5 has the following two problems:

1. If the image is passed to all computers sequentially while intersecting local silhouette images, it will produce idle computers waiting for the data arrival.

Table 1. Number of arithmetic operations in the case of $n_v = n^3$.

	Perspective Projection			Homography			Linear-wise & Plane-wise Homography		
n	add	mul	div	add	mul	div	add	mul	div
100	9.0E6	9.0E6	2.0E6	6.0E6	6.0E6	2.0E6	2.0E6	2.1E4	600
1000	9.0E9	9.0E9	2.0E9	6.0E9	6.0E9	2.0E9	2.0E9	2.0E6	6000

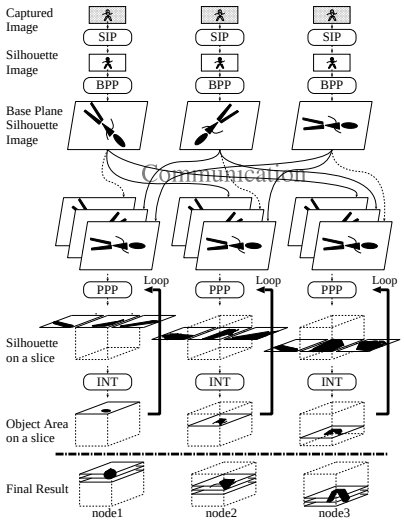


Figure 10. A process flow of the base-plane duplication method.

2. In this algorithm, the volume reconstruction is performed for each plane. This requires the synchronization of back-projection processes running on different computers. Since the elapsed time for homography may vary depending on the camera location, the synchronization will also cause idling time.

As for the first problem, by dividing the projected silhouette into small fragments and exchanging these fragments among the computers, the intersection process of divided planes can be done by all computers in parallel. We call this method *local silhouette division*. The process flow of this method is illustrated in Figure 9.

However, local silhouette division cannot solve the second problem. To solve this problem, we employ another parallelization method to share the result of base-plane projection computed by all computers. If all base-plane silhouettes are copied once by all computers, volume slices at any height can be computed on independent computers without any communication. This can be done by plane-wise homography and intersection processes. By assigning volume slice reconstructions at different heights to different computers, the synchronization problem can almost be solved. We call this parallelization *base-plane duplication*. The process flow is illustrated in Figure 10.

Figure 11. External view of the PC-cluster system.

5. Experiments

In this section, the prototype system for homography based VIM and some experimental results are shown.

5.1. PC cluster system

We are implementing the homography based VIM on the following PC cluster (Figure 11):

- The cluster consists of 9 node PCs and 1 host PC. Each PC has two Pentium III 600MHz CPUs and 256MB memories with the Linux operating system.
- These PCs are mutually connected by a high speed network (Myrinet by myricom) which provides full-duplex 1.28G bps

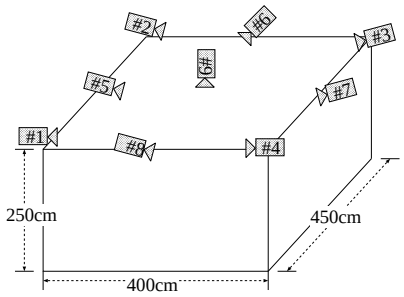


Figure 12. The camera settings.

bandwidth. The effective communication speed between 2 PCs is measured over 100M Bytes/second.

Each node PC has one pan-tilt-zoom camera (SONY EVI-G20). The camera parameters are controlled via RS-232C and its video output is captured by corresponding node PC. This camera can be regarded as a Fixed Viewpoint Camera (FVC): the optical center does not move while rotating the camera. In this case, by back projecting the observed images of many view directions onto a virtual plane using correct internal camera parameters, a seamless wide image can be generated. By using this property, internal camera parameters, such as radial-distortion coefficient, focal length, etc., can be obtained.

Camera settings and camera calibration

9 node PCs have 9 active cameras (FVC); these are mounted near the ceiling as shown in Figure 12. In this setting, the plane floor is commonly observed from every camera. This means the projection between image plane and the floor can also be represented by a homography. From the homography matrices H_1 and H_2 from the floor to image planes of two cameras, we can obtain a collineation matrix $M = H_2 H_1^{-1}$. Matrix M can be estimated by giving four corresponding points of the floor on two image planes. By decomposing the matrix M, we can obtain relative rotation R and translation T components between two cameras, as well as the surface normal vector of the floor $\overrightarrow{n}$. Based on this property, external camera parameters are obtained. That is, the external camera calibration is also done based on homography.

5.2. System implementation

The implemented VIM is based on base-plane duplication. The system has some additional functions to the original algorithm:

1. Each 2D silhouette image is cropped by its circumscribing rectangle by the corresponding node PC.
2. Back-projection and intersection are performed within the intersecting area of all frusta back-projected from circumscribing rectangles[3].
3. Plane-wise homography and intersection are not performed at those points where the previous intersection result is *'empty'*.

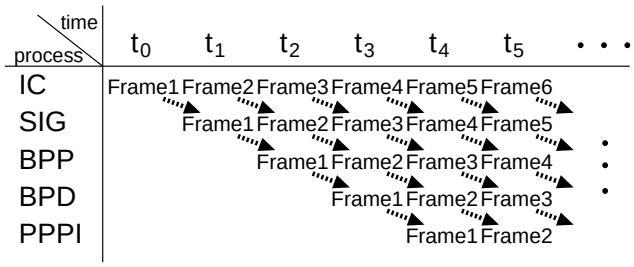


Figure 13. Pipeline processing on node PC.(IC: Image capture, SIG: silhouette image generation, BPP: base-plane projection, BPD: base-plane duplication, PPPI: parallel-plane projection and intersection)

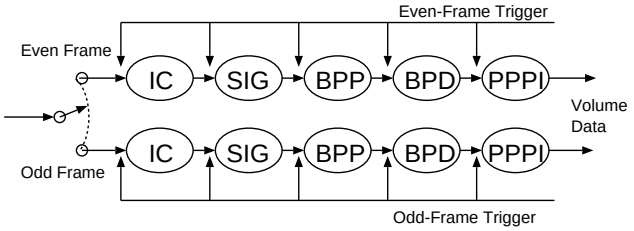


Figure 14. Multi-thread implementation of pipeline processing.(IC: Image capture, SIG: silhouette image generation, BPP: base-plane projection, BPD: base-plane duplication, PPPI: parallel-plane projection and intersection)

These functions reduce the size of the transmitted data, the number of back-projecting points, and the number of intersecting points.

Since each PC has 2 CPUs, each process on a single PC can also be parallelized. In our system, a pipeline processing is employed: VIM process on each PC is divided into chained subprocesses running in parallel. These subprocesses are: image capturing (IC), silhouette image generation(SIG), base-plane projection (BPP), base-plane duplication(BPD), and parallel plane projection/intersection(PPPI). They process different data sampled at different time simultaneously, e.g., when IC is capturing i-th input image, SIG is computing silhouette for $(i-1)$-th input, BPP is computing base-plane silhouette image for $(i-2)$-th input, BPD is exchanging base-plane image generated from $(i-3)$-th input, and PPPI is computing sliced volume for $(i-4)$-th input (see Figure 13). These subprocesses are implemented by threads as shown in Figure 14. To avoid overwriting input and output buffers, two series of threads are generated for even and odd frames, i.e., one series of threads are activated for even frames and another is for odd frames.

Of course, since each node PC doesn't have 5 CPUs, only 2 of them are executed at once. However, the operating system schedules these threads so as to maximize the CPU utilization and this pipeline processing is effective for the system throughput.

[3]Cropping is also applied to back-projected images.

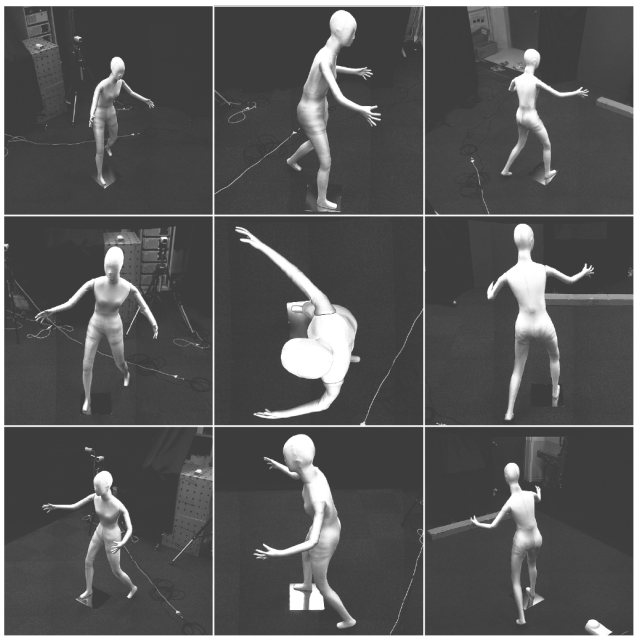


Figure 15. Examples of captured images by nine node PCs: From left-top to right-down, images captured by cameras No. 1, 5, 2, 8, 9, 6, 4, 7, 3.

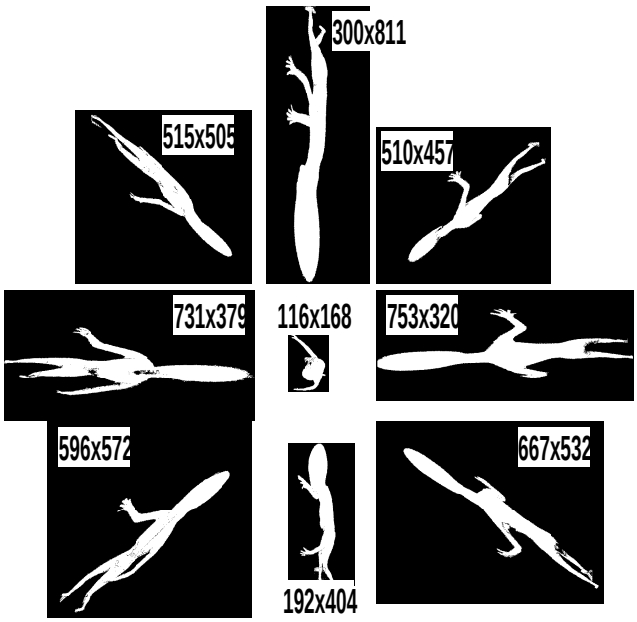


Figure 16. Examples of base-plane images: From left-top to right-down, images generated by node PCs No. 1, 5, 2, 8, 9, 6, 4, 7, 3.

5.3. Experimental results

The first experiment is on the accuracy of volume reconstruction. We first capture the background images of the scene, then put a mannequin in the scene and capture its images by using 9 cameras. Each image size is 640×480 pixels. The object (mannequin) is circumscribed by $70 \times 70 \times 180 cm^3$ rectangular parallelepiped, and the voxel resolution is $1 cm^3$. Examples of captured images are shown in Figure 15. The silhouette images are generated by background subtraction and projected onto the base-plane corresponding to the floor. The base-plane silhouette images projected by node PCs are shown in Figure 16. The volume reconstruction result is shown in Figure 17. Note that 1) some gaps are caused by misdetection of silhouettes, but 2) small parts of the object, such as fingers, are reconstructed in this experiment.

The second result is the elapsed time. We measured elapsed time in subprocesses for some voxel sizes (Figure 18) and processing speed (Table 2).

From Figure 18, we can notice that

- Elapsed time in Image Capturing(IC) and Silhouette Image Generation (SIG) doesn't depend on the voxel size,
- Elapsed time in Base-plane Projection(BPP), Base-plane Duplication(BPD), and Parallel Plane Projection and Intersection(PPPI) depends on voxel size.

Especially, the elapsed time in PPPI is almost in inverse proportion to the voxel volume (cube of voxel size) and it decreases drastically while the voxel size increases. The elapsed time in BPP is almost in inverse proportion to the square of the voxel size. The elapsed time in BPD should be in inverse proportion to the square

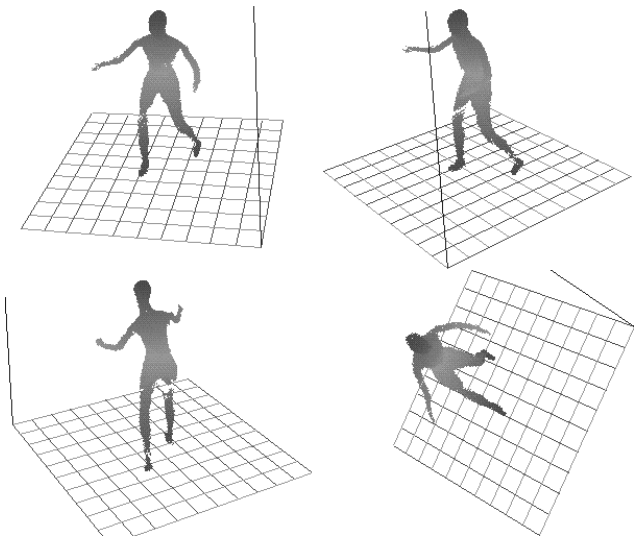


Figure 17. The reconstructed volume of the object. (viewed from different viewpoints).

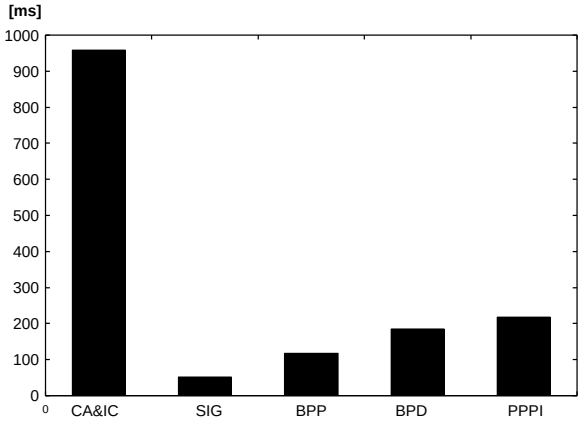


Figure 18. Averaged elapsed time in subprocesses for 50 frames. (IC: Image capture, SIG: silhouette image generation, BPP: base-plane projection, PPPI: parallel-plane projection and intersection, BPD: base-plane duplication)

Voxel Size	Total Elapsed time	Throughput Time	Volume /sec.	Pipeline Factor
$1cm^3$	651.019ms	564.413ms	1.77	1.15
$2cm^3$	145.127ms	114.065ms	8.77	1.27
$3cm^3$	107.031ms	80.873ms	12.37	1.32
$5cm^3$	80.484ms	64.013ms	15.62	1.25

Table 2. Processing speed. (Total Elapsed Time: sum of the elapsed time in all subprocesses, Throughput Time: time interval between the outputs, Volume/sec.: number of volumes reconstructed per 1 second, Pipeline Factor=(Total Elapsed Time)/(Throughput Time).)

of the voxel size, however, it gradually decreases while the voxel size increases.

Table 2 shows total elapsed time in all subprocesses and throughput time of the system. Since we employ pipeline processing, throughput time is shorter than total elapsed time. This pipeline effect is shown as "Pipeline Factor" and the average volume reconstruction rate "Volume/sec." in this table. In spite of the pipeline factor can be regarded as constant, the volume reconstruction rate increases gradually and it seems saturating while the voxel size increases. This is because subprocesses IC and SIG have constant elapsed time which dominates the total elapsed time at bigger voxel size.

Since our current system don't use MMX instructions, SIG process can be accelerated. Also, the image size 640×480 is too big for volume reconstruction, because a pixel corresponds to about $5mm^2$ area on the base plane. By reducing the image size to 320×240, image capturing can be accelerated. If the elapsed time in IC and SIG can be negligible, the volume reconstruction rate becomes 1.9, 13.6, 26.6, and 87.6 [volume/sec.], at 1cm^3, 2cm^3, 3cm^3, and 5cm^3 voxels, respectively. As well, since our current system dynamically allocates and releases memory area for each input image, BPP, BPD, and PPPI can also be accelerated.

The third experiment is the continuous volume reconstruction of a moving object. Figure19 shows an example of continuous human body reconstruction. In this figure, each surface geometry is generated from the volume which is reconstructed by applying our method to stored images, and the surface texture is computed from the observed images. The reconstructed moving object can be observed from any viewpoint, view direction, and zoom. In this case, these images are generated by changing the viewpoint from left above to front of the object with constant zoom.

6. Conclusion

This paper presents a homography based parallel 3D volume reconstruction method. In this paper, we didn't employ table look-up projection acceleration, because we are planning to realize active camera control to enlarge the working space of objects and the camera action will change the relationship between the image plane and 3D scene. We also didn't use octree volume representation so as to retain the possibility of extending the binary silhouette images to gray scale or color images[4]. We have shown that the life-sized mannequin can be reconstructed precisely using 1cm spatial resolution, and the system performance at different voxel size is analyzed.

The major contributions of this paper are summarized as:

1. It is shown that plane-to-plane homography is computationally effective for VIM.
2. Two types of acceleration method of homography computation are shown, one is our proposed *linear-wise homography*, and the other is *plane-wise homography* known as *dilation*[4][3] .
3. Parallel intersection method named *base-plane duplication* is proposed, which enables synchronization free parallel volume reconstruction on independent computers.

Based on this work, it is possible to capture the geometric information of a moving object as well as its photometric information. The captured object data can be observed from any viewpoint as shown in Figure 19. In other word, the extension of the motion capture system mentioned in Introduction means "total object capture system".
However, since our current system is still under construction, the frame-rate volume reconstruction and camera action have not been realized. These are the objectives of our current implementation effort, and we expect the current system can be improved to achieve the real-time volume reconstruction.

References

[1] H. Baker. Three-dimensional modelling. In *Fifth International Joint Conference on Artificial Intelligenc*, pages 649–655, 1977.

[2] B. G. Baumgart. Geometric modeling for computer vision. Technical Report AIM-249, Artificial Intelligence Laboratory, Stanford University, October 1974.

[4]This is unrelated to the morphological dilation operator.

Figure 19. Example of continuous human body reconstruction.

[3] R. T. Collins. A space-sweep approach to true multi-image matching. In *IEEE Computer Vision and Pattern Recognition*, pages 358–363, 1996.

[4] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. In *IEEE International Conference on Computer Vision*, pages 307–314, 1999.

[5] A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

[6] W. N. Martin and J. K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1987.

[7] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Computer Vision,Graphics, and Image Processing*, 40:1–29, 1987.

[8] J. Semple and G. Kneebone. *Algebraic Projective Geometry*. Oxford Science Publication, 1952.

[9] P. Srivasan, P. Liang, and S. Hackwood. Computational geometric methods in volumetric intersections for 3d reconstruction. *Pattern Recognition*, 23(8):843–857, 1990.

[10] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, 1993.