Recovery of 3D Models from Images

Greg Schmidt and Dave Minnigerode

CPSC 642
Computer Vision
Project Report

December 21, 1995

Abstract

Recovery of 3D coordinates from sets of images is useful in many areas. This allows for reconstruction of objects and recovery of motion characteristics from images. For this to be possible, the orientation and position of the cameras with respect to each other must be recovered first. This is the classic photogrammetric problem of relative orientation. Once this is solved, corresponding intersections from pairs of points may be found by projecting the rays from each camera through the images.

The work here involves taking a 3D model and projecting it onto different planes. The resulting images are used to determine the relative orientation of the cameras. From these, matched rays are projected and nearest intersections are computed. The resulting intersections are the recovered 3D coordinates from the original model. This plan reassures that the methods for reconstruction are working properly. Then matched points from pairs of images are used to derive the 3D coordinates.

1 Background

In reconstructing a 3D model from a pair of 2D images there are a number of problems that must be dealt with. The first of these is the correspondence problem. Here the issue is determining points in the images that correspond to the same point on the 3D object. Next is the Relative Orientation Problem. Here the goal is to find the correct orientation of one of the cameras relative to the other camera such that the pairs of corresponding rays (projected from the center of projections of each camera through the image and out into 3D space) all intersect one another. The problem of determining the relative orientation of two images is not a new one. Much of the work in solving this problem has been done in photogrammetry research[5]. Unfortunately, this work has had the benefit of having datasets that were limited in complexity. That is, the assumptions that could be made concerning the orientation of the cameras and of the subject points were rather wide ranging and greatly reduced the complexity of the problem. In current research, however, there is a need for more general solutions to the relative orientation problem so that little (if any) information is required about the initial orientation of the cameras. Once the relative orientation of the cameras has been established then the reconstruction of the 3D model may be performed.

2 Related Work

Haralick and Shapiro[4] take a traditional approach to the solution of the Relative Orientation Problem. By repesenting the unknowns in a linear system they arrive at error terms and adjustment methods that are relatively straightforward to compute. Horn presents two methods for determining the relative orientation solution. Both are iterative in nature and neither requires a "good" initial guess to find a solution. The first method[5] throws away the Euler representation of angles and uses quaternion representations instead. Methods are outlined for obtaining rough estimates of the parameters and iterative methods for refining the solutions. In the second paper on the relative orientation problem, a new method for determining the coplanarity condition is used[6]. This along with the use of the unit quaternions leads to a quicker and comparably reliable iterative algorithm.

3 Objective

The objective of this work is to provide a system in which Relative Orientation Problem can be visually evaluated. By importing model information directly and positioning virtual cameras about this model then coresponding points may be easily computed (via model vertices and camera pose). These point pairs may then be sent to any implementation of the Relative Orientation Problem. The relative orientation routine returns an estimate of the scene's configuration, the estimated pose of the cameras, and the estimated location of the model's vertices. This data may then be displayed along with the original scene. A system such as this would then provide for rapid evaluation of solutions to the Relative Orientation Problem.

To allow for the use of real world data the system will also be able to import images. Once the images are displayed on the screen the user may then manually select coresponding proints on the images and send that information to the relative orientation routine. The results of the Relative Orientation Problem provides an estimated 3D location of the chosen corresponding points. And as before this type of system would allow for a quick visual inspection of the relative orientation solution.

4 Design

The design set out to accomplish two different, yet related, tasks and uses the output of either to reconstruct a 3D model. The first task consists of taking a 3D model through the process of being "photographed" to obtain corresponding pairs of image points (all done numerically). The second task takes a pair of images and corresponding pairs of points that are manually selected. Once determined either set of the resulting image point pairs are used by the Relative Orientation Problem. Multiple starting points are then computed followed by a frequency analysis on the resulting relative orientations. Duplicate solutions are combined to determine the most frequent relative orientation. Then a routine was created to transform the image points to the camera coordinate system points. The resulting sets of points are used to construct 3D lines from the computed center of projections. Matched lines are then used to compute the nearest intersections. Finally, for the case of the 3D model, the initial model and camera orientation are displayed with the estimated model. When the image pairs are used the estimated model is displayed.

4.1 Obtaining the Corresponding Points

To allow for evaluation of various input data the system has been constructed to use Wavefront[10] object files. Currently the system only accesses the vertice information found in these files. The face and group information are ignored. This vertice information is taken to be 3D points in world coordinate space.

Given the center of projection (CP), the viewing direction ($V\vec{dir}$), the focal length (f) and the up vector ($V\vec{up}$) of each camera it is possible to construct a camera coordinate system and to project each of the 3D points to the image plane. To accomplish this a coordinate system for the camera is defined as follows:

$\vec{z'}$: unit vector (normalized viewing direction of the camera, positive in the z'-direction).

$\vec{u}$: unit vector

$\vec{v}$: unit vector

The above vectors form the coordinate system for the camera. The vector $\vec{z'}$ is determined by:

$$\vec{z'} = \frac{V\vec{dir}}{\|V\vec{dir}\|^2}$$

The vector $\vec{v}$ is determined by:

$$\vec{v} = V\vec{up} - \frac{(\vec{z'} \cdot V\vec{up})\vec{z'}}{\|\vec{z'}\|^2}$$

$$\vec{v} = V\vec{up} - (\vec{z'} \cdot V\vec{up})\vec{z'}$$

The vector $\vec{u}$ is determined by:

$$\vec{u} = \vec{z'} \times \vec{v}$$

This coordinate system is then taken to have an origin, $\vec{O}$, at the following position:

$$\vec{O} = CP + (V\vec{dir})f$$

Given this origin, focal length, and the viewing direction the image plane is defined.

Next is the projection of the 3D points of the model onto the image plane. Given a ray between a 3D point and the CP, any standard ray-plane intersection routine may be used (for example Foley, et al.[1] or Glassner[2]). For this application, the following was used:

$$t = \frac{(\vec{z'} \cdot \vec{Ro}) - (\vec{z'} \cdot \vec{O})}{(\vec{z'} \cdot \vec{Rd})}$$

where $\vec{Ro}$ is the ray origin (the current 3D point) and $\vec{Rd}$ is the ray direction defined as $\vec{Rd} = CP - \vec{Ro}$. The value t is the distance along the ray to the intersection point in $\|\vec{Rd}\|$ units. To calculate the resulting intersection point simply use the formula:

$$IP = \vec{Ro} + t(\vec{Rd})$$

where IP is the intersection point. The intersection point IP may now be used to calculate the 2D vectors from the origin $\vec{O}$. This may be accomplished by projecting the $I\vec{P - O}$ vector onto the $\vec{u}$ and $\vec{v}$ vectors as follows:

$$V\vec{pt} = IP - \vec{O}$$

$$uValue = V\vec{pt} \cdot \vec{u}$$

$$vValue = V\vec{pt} \cdot \vec{v}$$

Which results in a camera plane location of $(uValue, vValue)$ for each projected point. This procedure is performed on each of the points in the 3D model for both cameras and the results are sent to the relative orientation implementation described in Section 4.2.

The other approach to obtaining these (u,v) coordinate pairs is to manually generate them from scanned images (Figure 5). Images may be loaded into a palette where points of interest may be chosen by mouse. The software uses the SGI Graphics Library[7] and a graphical user interface package called FORMS[8]. The resulting pairs of corresponding (u,v) coordinates may also be sent to the relative orientation routine.

4.2 Relative Orientation

The relative orientation problem was implemented following the techniques of Haralick and Shapiro[4]. The algorithm used two sets of 6 points denoted by $(u_L[i], v_L[i])$ and $(u_R[i], v_R[i])$ where L and R refer to the corresponding left and right images and i refers to the i-th corresponding point. The i-th point in the left image matches with the i-th point in the right image. Along with these points, an initial guess was made for the solutions and used as input. Since there are 5 unknowns and 3 equations, the solution must be formulated using the least squares approximation. This is an iterative process that involves using an initial guess. The unknown parameters are the baseline denoted by $\vec{b}$ and the 3 rotation angles denoted by ω, ϕ and κ. One parameter of the baseline, the x-component, is known. The x-component in general may be assumed to be one unit. The initial guesses are used to create the error terms after each iteration. These error terms are added to the old values and used by the next iteration. If the error terms fall below some threshold then the algorithm converges and halts. If they don't converge after a specified cutoff iteration, the algorithm is assumed to diverge. This part of the algorithm is formulated as follows:

1. Given the points $(u_L[i], v_L[i])$ and $(u_R[i], v_R[i])$ and initial guesses denoted by $\vec{g} = (\omega_j, \phi_j, \kappa_j, y_j, z_j)$ where $j = 0$ compute the error terms via least-squares denoted by $\vec{e} = (\Delta\omega, \Delta\phi, \Delta\kappa, \Delta y, \Delta z)$.

2. If $\vec{e} < \vec{\epsilon}$ where $\vec{\epsilon}$ contains arbitrarily small positive values then the algorithm converges returning $R_{\omega,\phi,\kappa}$ and $\vec{b}$.

3. Else if $j > cutoff$ where j is the iteration and cutoff is the max ply then the algorithm diverges for $\vec{g}$.

4.3 Generation of Initial Guesses

The initial guesses are generated at random. The ranges for $\vec{g}$ are given as

$$0 < \omega_0, \phi_0, \kappa_0 < 360$$

$$-p < y_0, z_0 < p$$

where the rotation angles are in degrees and the value p is arbitrarily set to 50. A set of guesses, $G = (\vec{g_1}, \vec{g_2}, ..., \vec{g_n})$ is used to generated n solutions including repeats. Out of the n solutions are m different solutions. These m solutions are also not necessarily unique since multiple rotations about the 3 principal axes are not unique. So, q solutions are unique and are found by computing the composited rotation matrice in the order $R(\omega, \phi, \kappa) = R(\kappa)R(\phi)R(\omega)$. Therefore $q \leq m \leq n$ and a reduction algorithm is necessary to determine the final q solutions. This algorithm is formulated as follows:

1. Generate n guesses, G, using a random number generator with constraints for ω_0, ϕ_0, κ_0, y_0 and z_0 given above.

2. Compute the relative orientation for each guess and tabulate m different solutions with a frequency counter for all solutions that converge.

3. Compute $R(\omega, \phi, \kappa)$ for each of the m different solutions.

4. Reduce the number of solutions to q by comparing $\vec{b}$ and $R(\omega, \phi, \kappa)$ values. When reducing like solutions, combine the frequency counts.

4.4 Computation of 3D Points

The most frequent unique solution is then selected. The right center of projection, $\vec{CP_R}$, is then computed from the assumed left center of projection, $\vec{CP_R}$ and the baseline, $\vec{b}$. The left center of projection is arbitrarily assumed to be at the origin, $\vec{0}$. The right image points, $(u_R[i], v_R[i])$ are then transformed to get 3D image points, $(x_R[i], y_R[i], z_R[i])$, all lying on a plane. The left image points are converted rather easily to 3D image points, $(x_L[i], y_L[i], z_L[i])$ by using the focal length as the z-values. Rays are then cast through each corresponding 3D image point and nearest intersections are computed. These are computed using a method developed by Ronald Goldman[3]. These steps are formulated as follows:

1. Select the most frequent unique solution, $R(\omega, \phi, \kappa)$ and $\vec{b}$.

2. Compute the right center of projection by $CP_R = CP_L + \vec{b}$,

3. Transform the right image points by the following steps:

$$(u_R[i], v_R[i]) \overset{T(0,0,f)}{\rightarrow} (x'_R[i], y'_R[i], z'_R[i])$$

$$(x'_R[i], y'_R[i], z'_R[i]) \overset{R(-\omega,-\phi,-\kappa)}{\rightarrow} (x''_R[i], y''_R[i], z''_R[i])$$

$$(x''_R[i], y''_R[i], z''_R[i]) \overset{T(\vec{b})}{\rightarrow} (x_R[i], y_R[i], z_R[i])$$

4. Transform the left image points by the following:

$$(u_L[i], v_L[i]) \overset{T(0,0,f)}{\rightarrow} (x_L[i], y_L[i], z_L[i])$$

5. For each point, $P_L[i] = (x_L[i], y_L[i], z_L[i])$ and $P_R[i] = (x_R[i], y_R[i], z_R[i])$, compute vectors, $\vec{v}_L[i]$ and $\vec{v}_R[i]$ going through the corresponding center of projections by

$$\vec{v}_L[i] = P_L[i] - CP_L$$

$$\vec{v}_R[i] = P_R[i] - CP_R$$

6. Find the nearest intersection points $Q_L[i]$ and $Q_R[i]$

   (a) Let $M(t)[i] = P_L[i] + t\vec{v}_L[i]$ and $M(s)[i] = P_R[i] + s\vec{v}_R[i]$

(b) Compute t and s by

$$t = \frac{\begin{vmatrix} P_R[i] - P_L[i] \\ \vec{v}_R[i] \\ \vec{v}_L[i] \times \vec{v}_R[i] \end{vmatrix}}{\|\vec{v}_L[i] \times \vec{v}_R[i]\|^2}$$

$$s = \frac{\begin{vmatrix} P_R[i] - P_L[i] \\ \vec{v}_L[i] \\ \vec{v}_L[i] \times \vec{v}_R[i] \end{vmatrix}}{\|\vec{v}_L[i] \times \vec{v}_R[i]\|^2}$$

(c) Since in general the lines are skew, t and s will represent the points of nearest intersection. So $Q_L[i] = M(t)[i]$ and $Q_R[i] = M(s)[i]$.

Once the 3D points have been computed (either from the model and cameras or from the image pairs) then they may be displayed in the 3D world coordinate space. If the 3D points come from the model then both camera's coordinate systems will be displayed along with the original 3D points. In this manner the resulting solution may be examined visually. If the computed 3D points come from the image pairs then they are displayed as if the left camera is located at the origin and looking in the positive "z" direction, and the estimated center of projection of the right-hand camera is marked.

4.5 Implementation Notes

Some of the math was computed with the help of routines from *Numerical Recipes in C*[9]. The determinant was computed using a technique called LU decomposition. This technique takes a matrix A and finds two equivalent matrices L and U that replace A by $A = L \cdot U$. The matrix L is lower triangular and U is upper triangular. When a decomposition of a row-wise permutation, B, of A is computed, the amount of row interchanges are recorded. These indicate the sign of the determinant. The determinant is then the calculated by

$$det = \prod_{i=1}^{n} B_{i,i}$$

where n is the number of rows and i is the row and column index.

5 Discussion

We have implemented the Relative Orientation Algorithm from Haralick and Shapiro. We have tested it on generated 3D models and a set of images. A 3D model of a cube was made and fed into the program with the following input parameters and data:

```
LH Camera Info:
LHCOP: (0.0,0.0,0.0)
LHDir: (0.000,0.000,1.000)
LHUp: (0.0,1.0,0.0)
LHFocal: 0.075
z' (0.00,0.00,1.00)
```

```
v (0.00,1.00,0.00)
u (1.00,0.00,0.00)

RH Camera Info:
RHCOP: (5.0,5.0,0.0)
RHDir: (-1.000,0.000,1.000)
RHUp: (0.0,1.0,0.0)
RHFocal: 0.075
z' (-0.71,0.00,0.71)
v (0.00,1.00,0.00)
u (0.71,0.00,0.71)
```

Points	LHImage	RHImage
(-2.000,-2.000,3.000)	(-0.05000,-0.05000)	(-0.03000,-0.07425)
(-2.000,2.000,3.000)	(-0.05000,0.05000)	(-0.03000,-0.03182)
(2.000,2.000,3.000)	(0.05000,0.05000)	(0.00000,-0.05303)
(2.000,-2.000,3.000)	(0.05000,-0.05000)	(0.00000,-0.12374)
(2.000,-2.000,7.000)	(0.02143,-0.02143)	(0.03000,-0.07425)
(2.000,2.000,7.000)	(0.02143,0.02143)	(0.03000,-0.03182)

The cutoff ply is 30 and the number of iterations is 120. The relative orientations found are:

	omega	phi	kappa	y	z	count
[0]	134.00	359.00	269.00	5.000028	-0.059210	8
[1]	359.00	315.00	0.00	5.000029	-0.059015	7
[2]	180.00	224.00	180.00	5.000027	-0.058594	10
[3]	314.00	181.00	89.00	5.000027	-0.058534	5

The unique relative orientations found are:

	omega	phi	kappa	y	z	count
[0]	134.00	359.00	269.0	5.000028	-0.059210	13
[1]	359.00	315.00	0.00	5.000029	-0.059015	17

The most frequent solution is 1. The output points generated are:

```
left=<-2.029475,-2.029475,3.044202>  right=<-2.133013,-1.921421,3.047213>
left=<-2.047571,2.047571,3.071346>   right=<-2.097699,2.098843,3.003746>
left=<1.998841,1.998841,2.998252>    right=<1.943533,2.055392,2.997425>
left=<1.968871,-1.968871,2.953297>   right=<1.918242,-1.915374,3.022715>
left=<1.998589,-1.998589,6.995082>   right=<1.903861,-1.898800,7.050658>
left=<2.013459,2.013459,7.047127>    right=<1.933843,2.095686,7.046381>
```

A picture of the input and output cube are displayed in Figure 1. The results here are very accurate as can be seen by the output. Surprisingly very few starting points and iterations were used to generate these results. Note that the imput model is represented by the dark points in the figures and the estimated model is represented by the dark grey points and lines. The lines represent the closest approach of the two projected rays and the points represent the center point of the lines.

The next example is a data model of a simple house. The input parameters and data are:

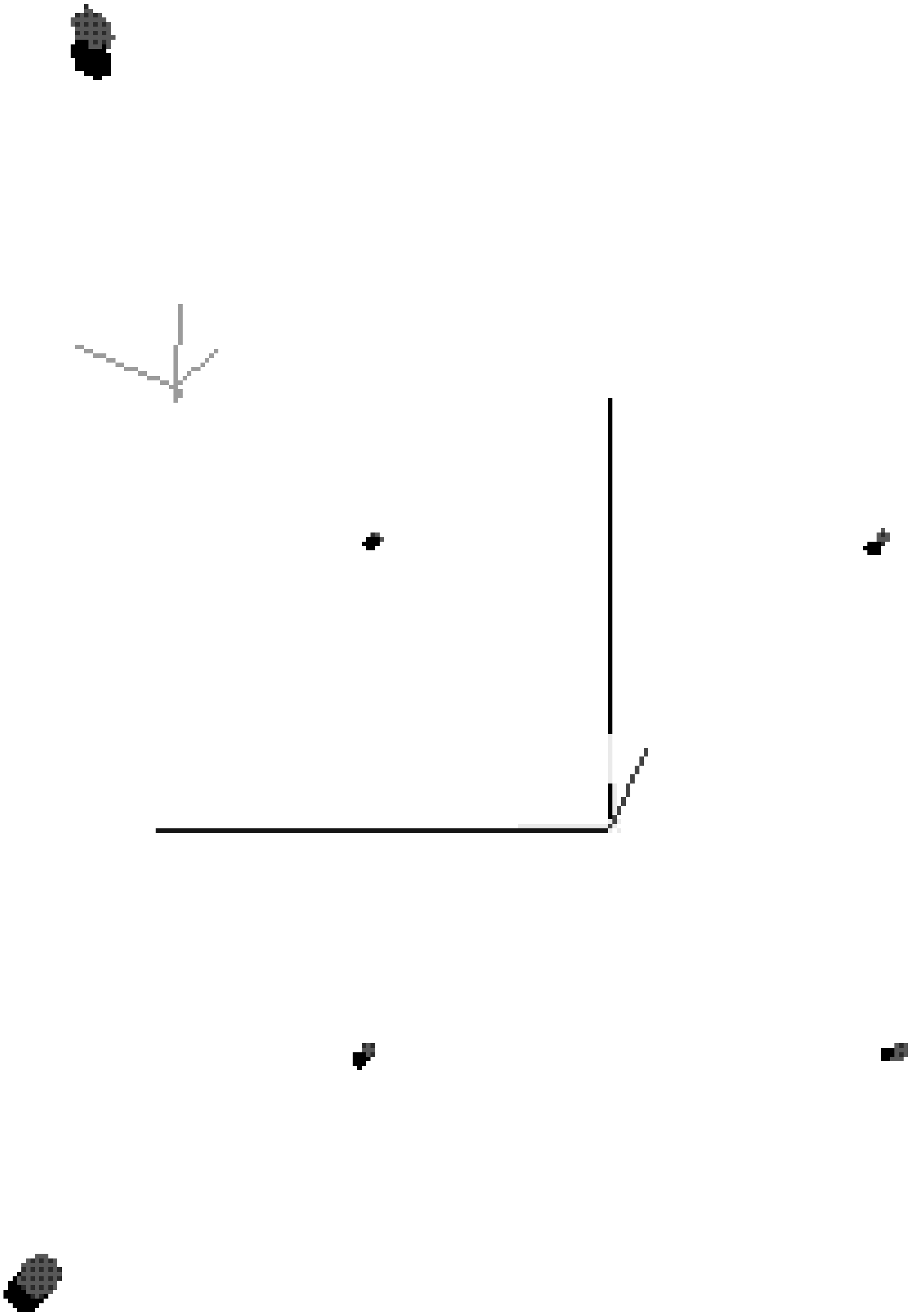

Figure 1: Input and Output Cube Together

```
LH Camera Info:
LHCOP: (0.0,0.0,0.0)
LHDir: (0.000,0.000,1.000)
LHUp: (0.0,1.0,0.0)
LHFocal: 0.075
z' (0.00,0.00,1.00)
v (0.00,1.00,0.00)
u (1.00,0.00,0.00)

RH Camera Info:
RHCOP: (7.0,0.0,5.0)
RHDir: (-1.000,0.000,0.000)
RHUp: (0.0,1.0,0.0)
RHFocal: 0.075
z' (-1.00,0.00,0.00)
v (0.00,1.00,0.00)
u (0.00,0.00,1.00)

Points                   LHImage                  RHImage
(-2.500,5.077,15.587)    (-0.01203,0.02443)       (0.08358,0.04008)
(-2.500,0.077,15.587)    (-0.01203,0.00037)       (0.08358,0.00061)
(2.500,0.077,15.587)     (0.01203,0.00037)        (0.17645,0.00129)
(2.500,5.077,15.587)     (0.01203,0.02443)        (0.17645,0.08462)
(-2.500,5.077,1.587)     (-0.11813,0.23992)       (-0.02694,0.04008)
(-2.500,0.077,1.587)     (-0.11813,0.00366)       (-0.02694,0.00061)
(2.500,0.077,1.587)      (0.11813,0.00366)        (-0.05688,0.00129)
(2.500,5.077,1.587)      (0.11813,0.23992)        (-0.05688,0.08462)
(-2.473,5.074,15.621)    (-0.01187,0.02436)       (0.08409,0.04017)
(-2.473,5.073,1.621)     (-0.11442,0.23474)       (-0.02675,0.04017)
(0.033,7.580,1.621)      (0.00154,0.35069)        (-0.03638,0.08160)
(0.033,7.580,15.621)     (0.00016,0.03639)        (0.11434,0.08160)
(0.068,7.523,15.586)     (0.00033,0.03620)        (0.11453,0.08138)
(0.068,7.522,1.587)      (0.00320,0.35561)        (-0.03693,0.08138)
(2.526,5.064,1.586)      (0.11941,0.23944)        (-0.05722,0.08489)
(2.526,5.065,15.587)     (0.01215,0.02437)        (0.17745,0.08489)
```

The cutoff ply is 20 and the number of iterations is 60. The relative orientations found are:

```
        omega       phi         kappa       y             z             count

[ 0]    4.00        320.00      334.00      -9.952770     -6.139069       1
[ 1]    159.00      212.00      166.00      2.937151      -3.959632       2
[ 2]    78.00       181.00      282.00      -9.952713     -6.139092       3
[ 3]    121.00      255.00      296.00      -1.646919     -3.342733       2
[ 4]    183.00      269.00      183.00      0.000002      4.999898        2
[ 5]    298.00      270.00      298.00      -0.000103     4.999963        1
[ 6]    159.00      269.00      160.00      -0.000035     4.999844        1
[ 7]    339.00      328.00      347.00      2.937085      -3.959657       2
[ 8]    99.00       322.00      258.00      2.937093      -3.959654       1
```

[9]	59.00	270.00	59.00	0.000014	4.999981	2
[10]	259.00	359.00	102.00	-9.952773	-6.139064	2
[11]	280.00	217.00	77.00	2.937040	-3.959674	1

The unique relative orientations found are:

	omega	phi	kappa	y	z	count
[0]	4.00	320.00	334.00	-9.952770	-6.139069	1
[1]	159.00	212.00	166.00	2.937151	-3.959632	4
[2]	78.00	181.00	282.00	-9.952713	-6.139092	5
[3]	121.00	255.00	296.00	-1.646919	-3.342733	2
[4]	183.00	269.00	183.00	0.000002	4.999898	6
[8]	99.00	322.00	258.00	2.937093	-3.959654	2

The most frequent solution is 4. The output points generated are:

```
left=<-2.424698,4.924446,15.117845>  right=<-2.493230,3.942019,15.426867>
left=<-2.550815,0.078954,15.903800>  right=<-2.423609,-0.921278,15.929170>
left=<2.556862,0.079141,15.941501>   right=<2.591212,-0.393561,15.938339>
left=<2.427003,4.929127,15.132216>   right=<2.342494,4.433497,15.307215>
left=<-2.061603,4.187018,1.308864>   right=<-2.424324,3.806228,1.955673>
left=<-2.559678,0.079228,1.625083>   right=<-2.507790,-0.918445,1.755451>
left=<4.434403,0.137255,2.815304>    right=<4.311645,-0.234657,3.026794>
left=<2.217367,4.503368,1.407755>    right=<2.013053,4.505770,1.721889>
left=<-2.398820,4.921651,15.152650>  right=<-2.467089,3.941690,15.460139>
left=<-2.042712,4.190687,1.338910>   right=<-2.404770,3.807859,1.984756>
left=<0.028566,6.510347,1.392314>    right=<-0.305884,6.393130,1.947271>
left=<0.031521,7.184312,14.805585>   right=<-0.124636,6.448895,15.162773>
left=<0.064099,7.133787,14.780642>   right=<-0.090870,6.401613,15.134694>
left=<0.058050,6.460490,1.362543>    right=<-0.273818,6.347555,1.912163>
left=<2.240970,4.493645,1.407569>    right=<2.037902,4.497178,1.719590>
left=<2.452174,4.917284,15.133069>   right=<2.368055,4.424155,15.306935>
```

A picture of the input and output house are displayed in Figure 2. The results here are fairly accurate. The results here took quite a few tries to generate. This particular dataset is not easily reproduced. Perhaps, the camera directions are such that multiple answers for the relative orientation are equally likely to obtain. One comment to mention is that a value of 20 was used for the ply, which is low in general. Higher values were considered in the analysis here.

The next example contains data from a scanned clay model of the metacarpal bone from a human finger. The input parameters and data are:

```
LH Camera Info:
LHCOP: (0.0,0.0,0.0)
LHDir: (0.000,0.000,1.000)
LHUp: (0.0,1.0,0.0)
LHFocal: 0.075
z' (0.00,0.00,1.00)
v (0.00,1.00,0.00)
u (1.00,0.00,0.00)
```

Figure 2: Input and Output House Together

RH Camera Info:
RHCOP: (7.0,0.0,5.0)
RHDir: (-1.000,0.000,0.000)
RHUp: (0.0,1.0,0.0)
RHFocal: 0.075
z' (-1.00,0.00,0.00)
v (0.00,1.00,0.00)
u (0.00,0.00,1.00)

Points:58	LHImage	RHImage
(0.770,0.010,5.510)	(0.01048,0.00014)	(0.00614,0.00012)
(0.530,-0.060,5.220)	(0.00761,-0.00086)	(0.00255,-0.00070)
(0.580,-0.510,5.270)	(0.00825,-0.00726)	(0.00315,-0.00596)
(0.220,-0.060,5.190)	(0.00318,-0.00087)	(0.00210,-0.00066)
(0.260,-0.520,5.200)	(0.00375,-0.00750)	(0.00223,-0.00579)
(0.460,-0.950,5.180)	(0.00666,-0.01375)	(0.00206,-0.01089)
(0.050,-0.950,5.180)	(0.00072,-0.01375)	(0.00194,-0.01025)
(-0.020,-1.300,5.140)	(-0.00029,-0.01897)	(0.00150,-0.01389)
(0.340,-1.360,5.170)	(0.00493,-0.01973)	(0.00191,-0.01532)
(0.350,-1.790,5.120)	(0.00513,-0.02622)	(0.00135,-0.02019)
(-0.080,-1.830,5.110)	(-0.00117,-0.02686)	(0.00117,-0.01939)
(0.720,-1.450,5.340)	(0.01011,-0.02037)	(0.00406,-0.01732)
(0.780,-1.790,5.220)	(0.01121,-0.02572)	(0.00265,-0.02158)
(0.730,-0.950,5.420)	(0.01010,-0.01315)	(0.00502,-0.01136)
(-0.070,-0.510,5.190)	(-0.00101,-0.00737)	(0.00202,-0.00541)
(-0.290,-0.950,5.350)	(-0.00407,-0.01332)	(0.00360,-0.00977)
(-0.260,-0.510,5.430)	(-0.00359,-0.00704)	(0.00444,-0.00527)
(-0.400,-1.340,5.350)	(-0.00561,-0.01879)	(0.00355,-0.01358)
(-0.460,-1.780,5.330)	(-0.00647,-0.02505)	(0.00332,-0.01790)
(0.730,-0.530,5.650)	(0.00969,-0.00704)	(0.00778,-0.00634)
(0.690,-0.920,5.860)	(0.00883,-0.01177)	(0.01022,-0.01093)
(0.770,-1.430,5.830)	(0.00991,-0.01840)	(0.00999,-0.01722)
(0.830,-1.730,5.620)	(0.01108,-0.02309)	(0.00754,-0.02103)
(0.670,-0.480,5.860)	(0.00858,-0.00614)	(0.01019,-0.00569)
(0.680,0.060,5.840)	(0.00873,0.00077)	(0.00997,0.00071)
(0.530,-0.430,6.160)	(0.00645,-0.00524)	(0.01345,-0.00498)
(0.470,0.080,6.110)	(0.00577,0.00098)	(0.01275,0.00092)
(0.540,-0.920,6.180)	(0.00655,-0.01117)	(0.01370,-0.01068)
(0.600,-1.350,6.230)	(0.00722,-0.01625)	(0.01441,-0.01582)
(0.800,-1.800,5.930)	(0.01012,-0.02277)	(0.01125,-0.02177)
(0.610,-1.780,6.250)	(0.00732,-0.02136)	(0.01467,-0.02089)
(0.860,-2.000,5.840)	(0.01104,-0.02568)	(0.01026,-0.02443)
(0.890,-2.190,5.990)	(0.01114,-0.02742)	(0.01215,-0.02688)
(0.710,-2.030,6.180)	(0.00862,-0.02464)	(0.01407,-0.02420)
(0.970,-1.880,5.720)	(0.01272,-0.02465)	(0.00896,-0.02338)
(0.790,-2.370,6.220)	(0.00953,-0.02858)	(0.01473,-0.02862)
(1.030,-2.350,5.150)	(0.01500,-0.03422)	(0.00188,-0.02952)

(0.570,-2.290,6.360)	(0.00672,-0.02700)	(0.01586,-0.02671)
(0.410,-2.060,6.420)	(0.00479,-0.02407)	(0.01616,-0.02344)
(0.310,-2.290,6.470)	(0.00359,-0.02655)	(0.01648,-0.02567)
(0.140,-2.260,6.330)	(0.00166,-0.02678)	(0.01454,-0.02471)
(0.100,-1.970,6.280)	(0.00119,-0.02353)	(0.01391,-0.02141)
(0.190,-1.390,6.360)	(0.00224,-0.01639)	(0.01498,-0.01531)
(0.130,-0.890,6.280)	(0.00155,-0.01063)	(0.01397,-0.00972)
(0.210,-0.420,6.210)	(0.00254,-0.00507)	(0.01337,-0.00464)
(0.150,0.110,6.110)	(0.00184,0.00135)	(0.01215,0.00120)
(-0.230,-0.080,5.480)	(-0.00315,-0.00109)	(0.00498,-0.00083)
(-0.140,-0.460,6.000)	(-0.00175,-0.00575)	(0.01050,-0.00483)
(-0.160,0.030,5.910)	(-0.00203,0.00038)	(0.00953,0.00031)
(-0.310,-0.820,5.690)	(-0.00409,-0.01081)	(0.00708,-0.00841)
(-0.390,-1.300,5.720)	(-0.00511,-0.01705)	(0.00731,-0.01319)
(-0.420,-1.810,5.680)	(-0.00555,-0.02390)	(0.00687,-0.01829)
(-0.100,-0.960,6.090)	(-0.00123,-0.01182)	(0.01151,-0.01014)
(-0.140,-1.360,6.140)	(-0.00171,-0.01661)	(0.01197,-0.01429)
(-0.250,-1.890,6.050)	(-0.00310,-0.02343)	(0.01086,-0.01955)
(-0.410,-2.080,6.010)	(-0.00512,-0.02596)	(0.01022,-0.02105)
(-0.150,-2.180,6.230)	(-0.00181,-0.02624)	(0.01290,-0.02287)
(0.020,-0.070,5.240)	(0.00029,-0.00100)	(0.00258,-0.00075)

The cutoff ply is 50 and the number of iterations is 100. The relative orientations found are:

	omega	phi	kappa	y	z	count
[0]	180.00	342.00	359.00	0.000535	4.998161	5
[1]	345.00	231.00	165.00	1.769547	-2.439333	6
[2]	186.00	195.00	201.00	1.769544	-2.439322	3
[3]	7.00	344.00	21.00	1.769548	-2.439338	3
[4]	164.00	308.00	344.00	1.769544	-2.439323	8
[5]	12.00	270.00	12.00	0.000053	4.999681	2
[6]	286.00	270.00	286.00	0.000010	5.000001	1
[7]	359.00	198.00	179.00	0.000476	4.998072	8
[8]	127.00	270.00	126.00	0.000013	4.999945	1
[9]	332.00	270.00	333.00	0.000007	5.000015	2
[10]	190.00	269.00	190.00	0.000038	4.999755	4
[11]	296.00	270.00	295.00	0.000010	5.000004	1
[12]	207.00	269.00	207.00	0.000033	4.999938	2
[13]	91.00	271.00	90.00	0.000012	4.999998	1
[14]	204.00	166.00	329.00	-9.297971	39.053604	1
[15]	263.00	271.00	263.00	-0.000025	5.000052	1
[16]	138.00	271.00	137.00	0.000008	5.000008	1

The unique relative orientations found are:

	omega	phi	kappa	y	z	count
[0]	180.00	342.00	359.00	0.000535	4.998161	13

[1]	345.00	231.00	165.00	1.769547	-2.439333	14
[2]	186.00	195.00	201.00	1.769544	-2.439322	6
[5]	12.00	270.00	12.00	0.000053	4.999681	16
[14]	204.00	166.00	329.00	-9.297971	39.053604	1

The most frequent solution is 5. The output points generated are:

```
left=<0.783009,0.010169,5.603087>    right=<1.797560,-2.306221,5.465511>
left=<0.541998,-0.061357,5.338166>   right=<1.627792,-2.451747,5.200447>
left=<0.616284,-0.541905,5.599681>   right=<1.833161,-2.810213,5.246241>
left=<0.222570,-0.060700,5.250612>   right=<1.363810,-2.569338,5.173234>
left=<0.274318,-0.548636,5.486364>   right=<1.559413,-2.943094,5.182665>
left=<0.505159,-1.043262,5.688541>   right=<1.873037,-3.236686,5.164802>
left=<0.054193,-1.029671,5.614423>   right=<1.510299,-3.401516,5.165377>
left=<-0.022202,-1.443117,5.705899>  right=<1.546756,-3.746657,5.129401>
left=<0.382421,-1.529684,5.815035>   right=<1.881542,-3.656480,5.156977>
left=<0.404441,-2.068425,5.916404>   right=<2.000920,-4.057566,5.111850>
left=<-0.091675,-2.097065,5.855738>  right=<1.625112,-4.274147,5.102956>
left=<0.813820,-1.638941,6.035840>   right=<2.236303,-3.589801,5.314319>
left=<0.903362,-2.073099,6.045564>   right=<2.385037,-3.879883,5.204596>
left=<0.798402,-1.039015,5.927826>   right=<2.098081,-3.136733,5.385092>
left=<-0.072890,-0.531054,5.404230>  right=<1.272651,-3.061393,5.173731>
left=<-0.308338,-1.010071,5.688314>  right=<1.214042,-3.535124,5.322463>
left=<-0.266851,-0.523439,5.573085>  right=<1.110269,-3.133979,5.393835>
left=<-0.435229,-1.458017,5.821193>  right=<1.224026,-3.934713,5.324919>
left=<-0.513737,-1.987937,5.952672>  right=<1.279286,-4.376943,5.309590>
left=<0.769542,-0.558707,5.956022>   right=<1.949746,-2.779981,5.595168>
left=<0.739860,-0.986480,6.283453>   right=<2.025440,-3.144155,5.793332>
left=<0.851310,-1.581004,6.445611>   right=<2.241552,-3.576245,5.772593>
left=<0.940329,-1.959963,6.367060>   right=<2.387248,-3.824479,5.579419>
left=<0.698440,-0.500375,6.108723>   right=<1.870941,-2.765861,5.789096>
left=<0.682547,0.060224,5.861874>    right=<1.694057,-2.302199,5.768367>
left=<0.544743,-0.441961,6.331360>   right=<1.724806,-2.781510,6.066516>
left=<0.466458,0.079397,6.063941>    right=<1.509801,-2.364181,6.015679>
left=<0.571475,-0.973625,6.540223>   right=<1.880402,-3.212112,6.092614>
left=<0.649510,-1.461398,6.744110>   right=<2.049306,-3.585114,6.149106>
left=<0.897083,-2.018435,6.649603>   right=<2.358228,-3.921047,5.874960>
left=<0.673735,-1.965980,6.903026>   right=<2.170071,-3.990748,6.180329>
left=<0.977045,-2.272199,6.634806>   right=<2.466807,-4.087685,5.793680>
left=<1.013213,-2.493188,6.819272>   right=<2.530813,-4.273631,5.942839>
left=<0.795811,-2.275346,6.926922>   right=<2.323352,-4.194078,6.121166>
left=<1.103187,-2.138134,6.505367>   right=<2.542227,-3.918395,5.676210>
left=<0.897227,-2.691679,7.064241>   right=<2.474889,-4.507529,6.171971>
left=<1.228176,-2.802150,6.140860>   right=<2.745807,-4.326990,5.141531>
left=<0.640686,-2.573981,7.148670>   right=<2.258044,-4.516110,6.304427>
left=<0.454548,-2.283827,7.117557>   right=<2.061644,-4.347390,6.352782>
left=<0.345933,-2.555443,7.219978>   right=<2.031143,-4.616665,6.409684>
left=<0.156312,-2.523317,7.067516>   right=<1.884043,-4.644381,6.272020>
left=<0.110332,-2.173543,6.928831>   right=<1.781528,-4.368578,6.213647>
```

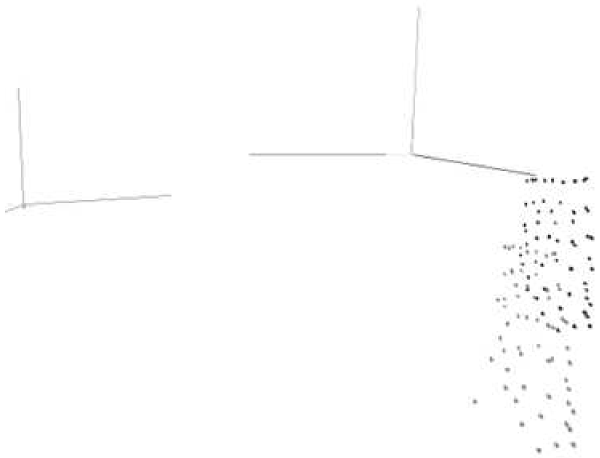

Figure 3: View1: Input and Output Finger Together

```
left=<0.203669,-1.490000,6.817538>   right=<1.707982,-3.779755,6.272162>
left=<0.135690,-0.928950,6.554830>   right=<1.525621,-3.339087,6.184493>
left=<0.213729,-0.427458,6.320289>   right=<1.456312,-2.890671,6.111676>
left=<0.147166,0.107922,5.994505>    right=<1.242052,-2.453541,6.013741>
left=<-0.227711,-0.079204,5.425486>  right=<0.998740,-2.751838,5.437946>
left=<-0.141964,-0.466452,6.084181>  right=<1.187232,-3.049930,5.917129>
left=<-0.156602,0.029364,5.784492>   right=<1.021234,-2.631900,5.829889>
left=<-0.322840,-0.853965,5.925668>  right=<1.156111,-3.428582,5.635209>
left=<-0.417719,-1.392395,6.126553>  right=<1.217273,-3.897066,5.668787>
left=<-0.463687,-1.998269,6.270811>  right=<1.317098,-4.393555,5.639202>
left=<-0.104676,-1.004884,6.374705>  right=<1.360367,-3.483097,6.008106>
left=<-0.149613,-1.453387,6.561628>  right=<1.432043,-3.867516,6.062968>
left=<-0.274735,-2.076993,6.648571>  right=<1.473330,-4.413587,5.990860>
left=<-0.453505,-2.300707,6.647703>  right=<1.384743,-4.658431,5.957122>
left=<-0.166487,-2.419617,6.914800>  right=<1.622443,-4.667361,6.171345>
left=<0.020069,-0.070240,5.257953>   right=<1.199469,-2.652522,5.218956>
```

Two views of the input and output of the finger are shown in Figures 3 and 4. This solution is fairly accurate as can be seen by the two output pictures. Once again, though, this result is not easily reproducible. If a small value for the ply is used then the results generally favor a solution that is 180 degrees off about the y-axis. This solution appears to be a reflected image of the correct solution. However, by increasing the ply value more accurate results come out more often. An interesting point about the reflected model case is that the resulting model could be scaled by a negative value to obtain the correct model (assuming that the model is the reflected model it appears to be).

The last example contains data from two images of a cup. The orientation appears to be off by 90 degrees about the y-axis with some other slight variations around the other axes. The two images and points selected are shown in Figure 5. The input 2D model parameters and data are:

```
imgfocal: 50.0
imgbasex: 600.0
```

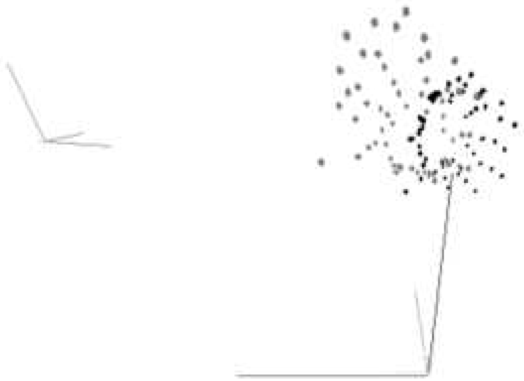

Figure 4: View2: Input and Output Finger Together

Figure 5: Input Images of a Cup with Selection Points

```
0: (205.000,235.000) -- (53.000,265.000)
1: (267.000,249.000) -- (99.000,243.000)
2: (212.000,71.000)  -- (59.000,99.000)
3: (268.000,89.000)  -- (109.000,68.000)
4: (75.000,303.000)  -- (163.000,346.000)
5: (233.000,327.000) -- (245.000,299.000)
```

The cutoff ply is 50 and the number of iterations is 200. The relative orientations found are:

```
       omega      phi      kappa    y             z           count

[ 0]   192.000    194.00   333.00   211.130951    129.225388   4
[ 1]   190.000    346.00   87.00    -184.705933   -85.217903   5
[ 2]   6.00       358.00   51.00    -184.705826   -85.217766   5
[ 3]   185.00     183.00   231.00   -184.706024   -85.217819   4
[ 4]   11.00      194.00   267.00   -184.706009   -85.217804   4
[ 5]   187.00     2.00     112.00   211.130905    129.225388   3
[ 6]   7.00       177.00   291.00   211.130920    129.225372   2
[ 7]   13.000     345.00   153.00   211.130890    129.225357   2
```

The unique relative orientations found are:

```
       omega   phi     kappa   y             z           count

[ 0]   192.00  194.00  333.00   211.130951   129.225388   6
[ 1]   190.00  346.00  87.00   -184.705933   -85.217903   9
[ 2]   6.00    358.00  51.00   -184.705826   -85.217766   9
[ 5]   187.00  2.00    112.00   211.130905   129.225388   5
```

The most frequent solution is 1. The output points generated are:

```
left=<-209.985387,-240.714956,-51.215948>   right=<-122.686187,-357.477959,139.642807>
left=<-379.311694,-353.740119,-71.032152>   right=<-230.070800,-556.570074,142.114875>
left=<428.164617,143.394754,100.982221>     right=<583.899083,-196.637738,-76.485404>
left=<651.400669,216.323357,121.529976>     right=<747.894028,84.236178,-160.559323>
left=<-120.418264,-486.489786,-80.278843>   right=<-80.253465,-517.536427,47.616739>
left=<-737.673691,-1035.275953,-158.299075> right=<-595.537980,-1171.491563,70.199896>
```

The output 3D model points are shown in Figure 6. The view shown is not very easily reproducible once again. The focal length and baseline x-value had to be adjusted to the size of the data points coming in. The output 3D points are fairly accurate in this example. The ply was increased to 50 and the iterations of the algorithm to 200 to eliminate bad matrice cases and divergent cases.

6 Conclusion

The Relative Orientation Problem was successfully implemented. The image data had some minor problems at first with choosing an appropriate focal length and baseline x-value, but a few good outputs resulted. The implementation proved to be fairly accurate, but perhaps the image data could have produced better results with another implementation such as Horn's.

Figure 6: Output 3D Model Points of Cup

Finding the initial starting values for the Relative Orientation Algorithm showed to be sufficient by picking random values on the sphere and on a limited interval for the baseline. The frequency analysis, in general, chose the most accurate solution. The whole implementation, overall, will serve as a good prototype for future computer vision experiments.

A User Manual

This section contains information on how to use the software. If there are any comments or questions send email to:

Greg Schmidt
HRBB 520B
gschmidt@cs.tamu.edu

Dave Minnigerode
HRBB 419A
minniger@cs.tamu.edu

In the default mode the system (executing without any command line arguments) takes a simple cube-like model, default cameras and camera position and runs them through the relative orientation routine and displays the results.

For more interesting cases the user must supply a configuration file. This configuration file may contain a detailed scene definition or only a few items. Any part of the scene that the user does not specify has a default setting. If the user wishes to use the image selection portion of the system, then this must be indicated in the configuration file. The two modes of operation are currently exclusive. That is, if the user selects image mode then any information in the configuration file concerning the modeling mode is ignored. A list of valid configuration commands are listed in Section A.4.

A.1 Selecting Image Points

Selecting image points may be performed by setting the option for image retrieval in the configuration file. With this option selected the software will then open a window and the interface will appear. There are three buttons which are described below:

read button This button is used to select an image file from disk. The format of the file should be an SGI file. A file browser will appear and files may be selected by either double clicking on the name or pressing the "accept" button.

redraw button This button will refresh the screen if one or more of the images becomes marred.

exit button This button is used when the user is done selecting points in the first two images read in.

The image points are selected by using the left mouse and pressing it while in one of the images. The order in which you pick is important. Points may be selected in the first image followed by the second image or corresponding points may be selected one at a time. An error will occur if the second image has less points selected then the first upon exiting. The amount of points chosen is the number in the first image.

A.2 Displaying Results

When the system has processed the given input a viewport window will be placed on the screen. A view of the origin and model will be shown. The controls are as follows:

Left Mouse Button This button is used to rotate the display so that interesting points of the model can be moved to a better viewing position.

Middle Mouse Button This button scales the scene. Effectively moving the viewpoint to and form the origin.

Right Mouse Button This button activates the menuing system. Currently there is only a single level of menus. Essentially the menus allow for the toggling on and off of the various items in the scene.

A.3 Sample Walkthrough of the System

```
%ro house.cfg
```

This results in:

1. Configuration file *house.cfg* being read.

2. The model in the *house.obj* Wavefront file being loaded.

After the processing is done then the scene will be displayed. The user could then use the left mouse button to rotate the right camera to the front of the viewport. Then use the middle mouse button to scale the scene so that the detail of the right hand camera can be seen easily. To get rid of the green lines that will be obscuring the scene the right mouse button could be used to select the "Point-COP ray" item. Once this item is selected, then the system will toggle the display of the rays to off. When the user desires to exit the system they may simply select the "Quit" menu item.

A.4 Configuration Commands

A list of available commands (and default values) are:

\#
: This line is a comment.

allfocal: <*real*>
: Sets the focal length of both cameras to the same value.
: Default: 0.075

lhfocal: <*real*>

rhfocal: <*real*>
: Sets the left (right) focal length to the given value.

lhcop: <*real*> <*real*> <*real*>

rhcop: <*real*> <*real*> <*real*>
: Sets the center of projection of the desired camera.
: Default: 0.0 0.0 0.0
: Default: 5.0 0.0 5.0

lhdir: <*real*> <*real*> <*real*>

rhdir: <*real*> <*real*> <*real*>
: Sets the viewing direction of the desired camera.
: Default: 0.0 0.0 1.0
: Default: -1.0 0.0 0.0

lhup: <*real*> <*real*> <*real*>

rhup: <*real*> <*real*> <*real*>
: Sets the up vector of the desired camera.
: Default: 0.0 1.0 0.0
: Default: 0.0 1.0 0.0

pushmodel: <*real*> <*real*> <*real*>
: Allows the .obj model to be repositioned from it's initial coordinates.
: Default: 0.0 0.0 0.0

objfile: <*filename*>
: Wavefront .obj file that is to be loaded and used.
: Note: There is no default for this and it must be specified unless using images.

img:
: Tells the system to skip loading the model and to activate the image routine for manually picking corresponding points.

imgfocal: <*real*>
: Same as "allfocal:" but nicer to use.
: Default: 0.075

imgbasex: <real>

Sets the x-portion of the baseline.

ply: <int>

Determines the number of guesses the relative orientation routine will make at a starting solution. This can also be used with the *img:* command.
Default: 30

iter: <int>

Determines the number of iterations that the relative orientation routine will allow for the solution to converge. This can also be used with the *img:* command.

A.5 Sample Configuration Files

A.5.1 house.cfg

```
#This file is a configuration file for the "ro" program
#it is a sample of how one can set the cameras and the models
# into differnet locations (and change any other parm)
#for the "ro"
#If any of the following are not give by the user then the default
#options will be used

#number of guesses at the starting point for the rel_or code
ply: 20
#number of iterations allowed for rel_or to converge
iter: 60

#Location of the RH camera center of projection
rhcop: 7.0 0.0 5.0

#and direction it's looking
rhdir: -1.0 0.0 0.0

#Ways of defining focal length (Most recent is used)
lhfocal: 1.0
rhfocal: 2.0
allfocal: .075


#wavefront file that has the model
# if not specified then just defaults to a cube like thing
objfile:  house.obj


# to allow the user to move the models pos. if needed.
pushmodel:  0.0 -5.0 5.0
```

A.5.2 image.cfg

```
# image.cfg
#
# This is a sample configuration file for the "ro" program
#


# indicate that the image portion of the system should be
# used
img:

# crank up the number of iterations and ply,
# since have only limted data
ply:40
iter:120

# and send the guess at this stuff
imgfocal: .5
imgbasex: 10.0
```

References

[1] Foley, J. D., Van Dam, A., Feiner, S. K., and Hughes, J. F. *Computer Graphics.* Addison-Wesley Publishing Company, 1990.

[2] Glassner, A. S. *An Introduction to Ray Tracing.* Academic Press, 1989.

[3] Glassner, A. S. *Graphics Gems.* Academic Press, Inc., 1990.

[4] Haralick, R. M., and Shapiro, L. G. *Computer Vision and Robot Vision.* Addison-Wesley Publishing Company, 1993.

[5] Horn, B. K. P. Relative Orientation. *International Journal of Computer Vision* (1990), 59–78.

[6] Horn, B. K. P. Relative Orientation Revisited. *Journal of Optical Society of America* (October 1991).

[7] McLendon, P. *Graphics Library Programming Guide.* Silicon Graphics Inc., Mountain View, CA, 1991.

[8] Overmars, M. *FORMS, A C-Library for Dialogues.* Department of Computer Science, Ultrecht University, The Netherlands, 1991.

[9] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C.* Cambridge University Press, 1992.

[10] Rose, L., and Ramey, D. *Advanced Visualizer User's Guide.* Wavefront Technologies Inc., Santa Barbara, CA, 93103, March 1991.