Plane Clustering from Spherical Image Sequences

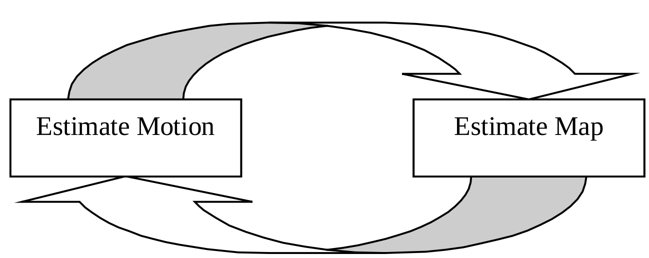
Michael Bosse

December 14, 1998

Introduction

The motivation for this project comes from the ARGUS image acquisition platform of the MIT Computer Graphics Group's City Scanning Project. This project addresses a fundamental bottleneck in the realization of convincingly realistic and complex virtual environments: the acquisition of geometric models representing built structures and other environmental features. Briefly, the project involves the construction of a platform to acquire high-resolution images of the outdoor environment using digital cameras, along with absolute position and time information about the cameras. The geo-referenced camera pose estimates are currently estimated by a navigation system using GPS, an inertial sensor, a magnetometer, and two-wheel odometry. The position estimates rely on the accuracy of the GPS data to correct for drifts in the other relative motion sensors. However, in urban environments GPS performs poorly. This is due to many error sources such as signal obstruction, multi-path, and electro-magnetic interference (EMI). Here in Cambridge, we have especially experienced major problems in GPS based navigation due to the enormous amounts of EMI from cellular phones, and TV and radio broadcasts from the Prudential Building.

One solution is to use vision-aided navigation. We have been researching the use of very large field-of-view or spherical cameras, to augment the current sensor suite, providing backup observations of velocity and position. Position estimates can be derived from tracking features and corresponding them with a map database. And velocity estimates can be computed from the estimating image sequence's optical flow from its spatio-temporal brightness gradients. Computing motion from optical flow is much easier with a good model for the depth at each image point. And computing the depth or structure from the images is easier when the camera motion is known. In this class project, we will address this latter problem of computing a map from images assuming known camera motion.



Mathematical Models

Spherical image coordinates are the unit vectors of the direction from the center of projection to each image point. Representing the points on the image with spherical coordinates provides several advantages over the traditional planar coordinates. The image model can be more generic, and can represent very large field of view's (> 180 deg), such as those from fisheye lenses, images of parabolic mirrors, or two cameras back-to-back. Planar coordinates have the disadvantage that they cannot handle cameras with fields of view larger than 180 degrees. Large field-of-view (FOV) cameras are well suited for navigation since they are better able to distinguish rotational from translational motion.

Spherical Image Motion

Projecting the 3 dimensional world onto the 2 dimensional surface of the unit sphere creates spherical images. Thus the position of a point r in the spherical image is simply the unit vector representing the direction to the 3D point *R*.

$$r = \frac{R}{\|R\|} \tag{1}$$

To compute the projected motion field, we take the time differential of the above equation, and get an image velocity field:

$$\dot{r} = \frac{\dot{R}}{\|R\|} - \frac{R\left(R^T \dot{R}\right)}{\|R\|^3} = \left(I - rr^T\right)\frac{\dot{R}}{\|R\|} \tag{2}$$

Note that the velocity $\dot{r}$ is always perpendicular to the image point r.

We now make the assumption that the camera is moving through rigid scene, and add in a rigid body motion constraint:

$$\dot{R} = -\omega \times R - t \tag{3}$$

$$\dot{r} = -\omega \times r - \left(I - rr^T\right)\frac{t}{\|R\|} \tag{4}$$

The image motion field in Equation (4) depends on the rotational velocity ω, the translational velocity t, and the distance $\|R\|$ of each scene point from the camera. We can simplify the dependence on the distance $\|R\|$, by assuming that the scene is a planar surface. We do this by adding the planar surface constraint:

$$\begin{aligned} \hat{n}^T R &= d \\ p^T R &= 1 \end{aligned} \tag{5}$$

Where *n* is the unit normal of the plane, *d* is the distance of the plane from the origin. The plane can be concisely parameterized by the vector *p*, which is the normal divided by the distance.

Thus,

$$\frac{t}{\|R\|} = \frac{t(R^T p)}{\|R\|} = tr^T p \tag{6}$$

Finally we arrive at the planar surface motion field equation.

$$\dot{r} = -\omega \times r\left(I - rr^T\right)tr^T p \tag{7}$$

Constant Brightness Constraint

The images from a camera are patterns of varying brightness values that come from the interaction of light and the scene. The apparent motion of these brightness patterns is termed *optical flow*. This is to be distinguished from the *motion field*, which is a mathematical projection. In order to make sense of the images, we have to assume that optical flow results from the motion field, and not from changing light sources.

By assuming that the total brightness in the image is conserved, we can relate the spatio-temporal brightness gradients E_x, E_y, E_t, to the image plane flow field (u,v).

$$\begin{aligned} \tfrac{d}{dt}E &= 0 \\ E_x u + E_y v + E_t &= 0 \end{aligned} \tag{8}$$

However, since we are representing the image coordinates with unit vectors, we need to convert the constant brightness constraint to use an image velocity represented as spherical coordinate velocity $\dot{r}$.

We start by expressing the spherical coordinate r to the image coordinate $(x,y,z)^T$:

$$r = \frac{1}{\sqrt{x^2 + y^2 + f^2}}\begin{pmatrix} x \\ y \\ f \end{pmatrix} \tag{9}$$

Then differentiate both sides with respect to time:

$$\dot{r} = \frac{1}{\sqrt{x^2 + y^2 + f^2}}\left(I - rr^T\right)\begin{pmatrix} u \\ v \\ 0 \end{pmatrix} \tag{10}$$

Expanding to see the components of the matrix, and dropping the third column:

$$\dot{r} = \frac{1}{\sqrt{x^2 + y^2 + f^2}}\begin{pmatrix} 1 - r_x^{\ 2} & -r_x r_y \\ -r_y r_x & 1 - r_y^{\ 2} \\ -r_z r_x & -r_z r_y \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} \tag{11}$$

We would like the dot product of the image spatial gradient E_s with the image point in spherical coordinates to equal the analogous product with planar coordinates. We also apply the constraint that the spherical image gradient E_s is perpendicular r:

$$\begin{aligned} \dot{r}^T E_s &= E_x u + E_y v \\ r^T E_s &= 0 \end{aligned} \tag{12}$$

Now substituting Equation (11) for $\dot{r}$, we get the following system of equations:

$$\frac{1}{\sqrt{x^2+y^2+f^2}}\begin{pmatrix} 1-r_x^{\;2} & -r_x r_y & -r_x r_z \\ -r_y r_x & 1-r_y^{\;2} & -r_y r_z \\ r_x & r_y & r_z \end{pmatrix} Es = \begin{pmatrix} E_x \\ E_y \\ 0 \end{pmatrix} \tag{13}$$

Which has the solution:

$$E_s = \sqrt{x^2+y^2+f^2}\begin{pmatrix} 1 & 0 & r_x \\ 0 & 1 & r_y \\ -r_x/r_z & -r_y/r_z & (1-r_x^{\;2}-r_y^{\;2})/r_z \end{pmatrix}\begin{pmatrix} E_x \\ E_y \\ 0 \end{pmatrix} \tag{14}$$

Or more concisely:

$$E_s = \sqrt{x^2+y^2+f^2}\begin{pmatrix} E_x \\ E_y \\ \dfrac{-xE_x - yE_y}{f} \end{pmatrix} \tag{15}$$

Finally we substitute Equation (7) to get an expression relating the motion of the camera and the scene geometry to the spatio-temporal brightness gradients:

$$-E_s^{\;T}(\omega \times r) - E_s^{\;T}\left(I - rr^T\right)tr^T p + E_t = 0 \tag{16}$$

Mixture Model and EM-algorithm

Very few scenes have only one plane in them. We can get a better model of the world by assuming it to be generated from some number of planes. This is a good assumption for many urban scenes in which we expect to navigate in.

Mixture Model

In order to estimate the position of the planes in the scene we model the probability of observing the motion constraint c_k at each pixel r_k, as a mixture of probabilities $\Pr_i\left(c_k | r_k, p_i\right)$ with the mixture coefficients m_i:

$$\Pr(c_k|r_k,\bar{m},p_0,\cdots,p_N)=\sum_{i=0}^{N} m_i \Pr_i(c_k|r_k,p_i) \tag{17}$$

The best estimates for the planar parameters $p_0,\cdots,p_N$ will maximize the above likelihood. We use and Expectation-Maximization algorithm to solve for the most likely values for the planar parameters.

The probability of a particular plane matching the motion constraint is modeled with a robust error norm $\rho(\,)$.

$$\Pr(c_k|r_k,p)=\rho\left(c_k{}^T p,\sigma\right) \tag{18}$$

Where the following inner product is derived from Equation (16):

$$c_k{}^T p=\begin{pmatrix}-E_s{}^T\left(I-rr^T\right)tr^T \\ -E_s{}^T(\omega\times r)+E_t\end{pmatrix}\begin{pmatrix}p \\ 1\end{pmatrix} \tag{19}$$

This robust error norm is the same one as in Black and Jepson’s paper "Estimating Optical Flow in Segmented Images using Variable-order Parametric Models", and is defined with the scale parameter σ:

$$\rho(x,\sigma)=\frac{2}{\pi}\frac{\sigma^3}{\left(\sigma^2+x^2\right)^2} \tag{20}$$

Constraints with residuals larger than $\rho(2.5\sigma,\sigma)$ are considered outlier’s. We thus can add an outlier probability $\Pr_0$ to the mixture model with a probability fixed at $\rho(2.5\sigma,\sigma)$.

Expectation-Maximization

We use an Expectation-Maximization algorithm to iteratively obtain a maximum likelihood fit for the parameters $p_0,\cdots,p_N$ and the mixture coefficients m_i. The Expectation step determine the expected value of the hidden constraint labels q_{ik}:

$$\begin{aligned} q_{ik} &= \Pr(p_i|c_k)=\frac{\Pr(p_i)\Pr_i(c_k|p_i)}{\Pr(c_k)} \\ q_{ik} &= \frac{m_i \Pr_i(c_k|p_i)}{\sum_{j=0}^{N} m_j \Pr_j(c_k|p_j)} \end{aligned} \tag{21}$$

These constraint labels can be thought of as "ownership" probabilities of the motion constraints to each planar parameter. The sum of "ownership" probabilities over the component models is one. As the scale parameter σ in the error norm decreases, the assignment of the constraint labels becomes stricter, meaning that they are closer to being either 0 or 1.

In the Maximization step, the EM-algorithm uses the constraint labels q_{ik} to determine the maximum likelihood fit for parameters m_i, p_i. We are using a weighted least-squares approach as an approximation for this step:

$$\begin{aligned}
\hat{p}_i &= \left(C^T Q_i C\right)^{-1} C^T Q_i Y \\
C^T &= \begin{pmatrix} -E_s^{\ T}\left(I - r_k r_k^{\ T}\right) t r_k^{\ T} \\ \vdots \end{pmatrix} \\
Q_i &= \begin{pmatrix} q_{i0} & 0 & \cdots \\ 0 & q_{i1} & 0 \\ \vdots & 0 & \ddots \end{pmatrix} \\
Y &= \begin{pmatrix} E_s^{\ T}\left(\omega \times r_k\right) - E_t \\ \vdots \end{pmatrix}
\end{aligned} \tag{22}$$

The updated mixture probabilities are computed as follows, and are ensured to sum to one.

$$\hat{m}_i = \frac{1}{N} \sum_k q_{ik} \tag{23}$$

The entire EM-iteration is repeated while the scale term σ is annealed until the change in parameters becomes insignificant.

Determining the number of planes

The EM algorithm does not explicitly determine the number of planes in the scene. Therefore we first run the EM algorithm using more planes than expected and prune the bogus planes. The bogus planes are pruned by clustering the planar parameters, keeping only the plane with the highest mixture probability from each cluster. Afterwards the EM algorithm is re-run with the good subset of the planes, to refine their parameters.

Adding Vertical Plane Constraint

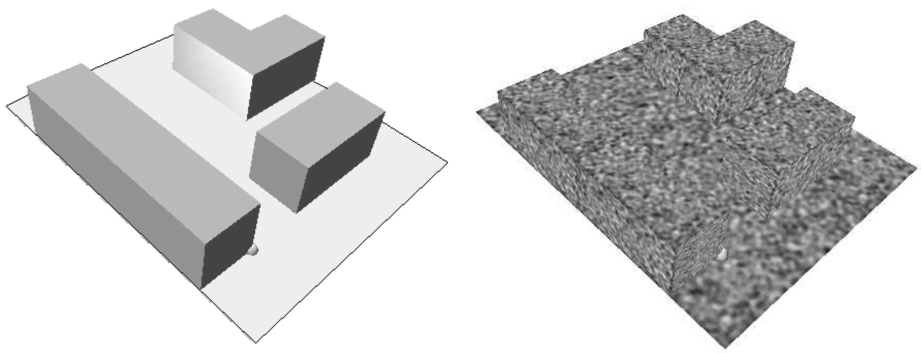
The EM algorithm is very susceptible to run into lots of local minimums. Adding some more constraints using prior assumptions can help prevent unlikely solutions. In urban scenes, we are likely to have one ground plane and multiple vertical planes from the sides of buildings. We enforce this constraint, on all but one plane, by applying a penalty term to the error norm. The penalty is proportional to the vertical component of the planar parameters. Note that the vertical component of the planar parameter is zero for vertical planes. The least-squares method in the maximization step is also modified with a LaGrange multiplier λ on the constraint:

$$\hat{z}^T p = 0 \tag{24}$$

$$\hat{p}_i = \left(C^T Q_i C + N\lambda \hat{z}\hat{z}^T\right)^{-1} C^T Q_i Y \qquad (25)$$

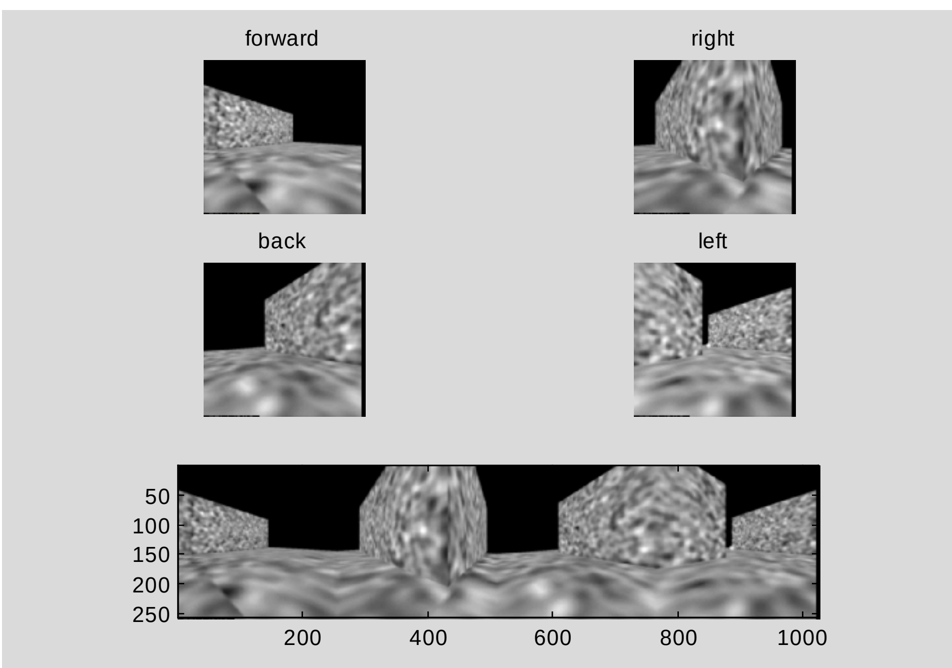
Experimental Setup

In our experiments, we used synthetically generated images to test the algorithm. The scene was generated by hand as an example of a typical urban scene. The model is then textured mapped with textures generated from smoothed Gaussian noise.

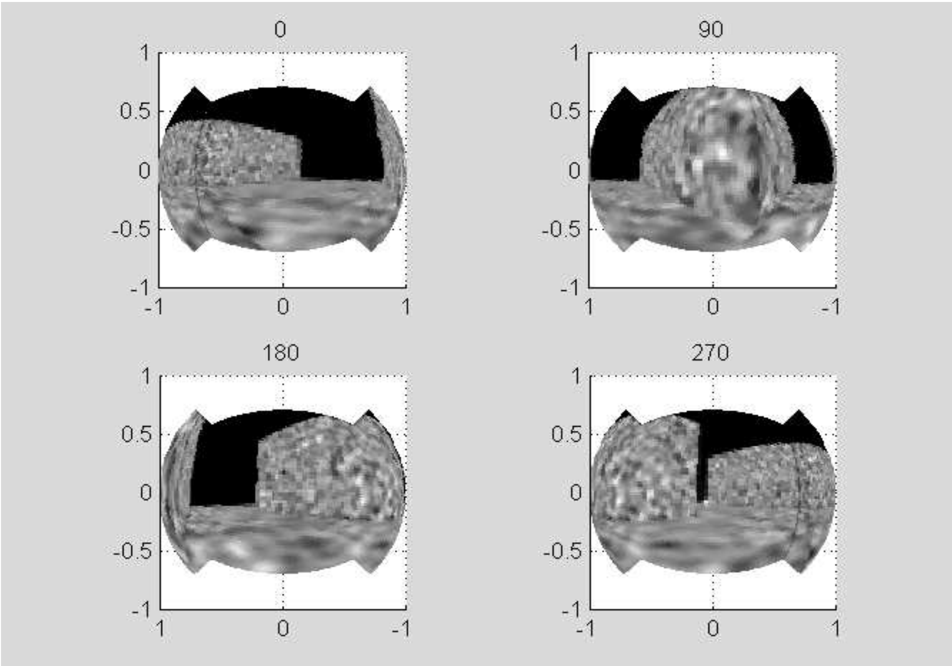


The camera's trajectory is a simple traslational motion with no rotation moving at 0.1 meter per frame. (The building dimensions are all integer multiples of 10 meters.) The spherical images were generated by piecing together four 90-degree FOV planar images with common center of projection rotated about the vertical axis.

Sample image frame:

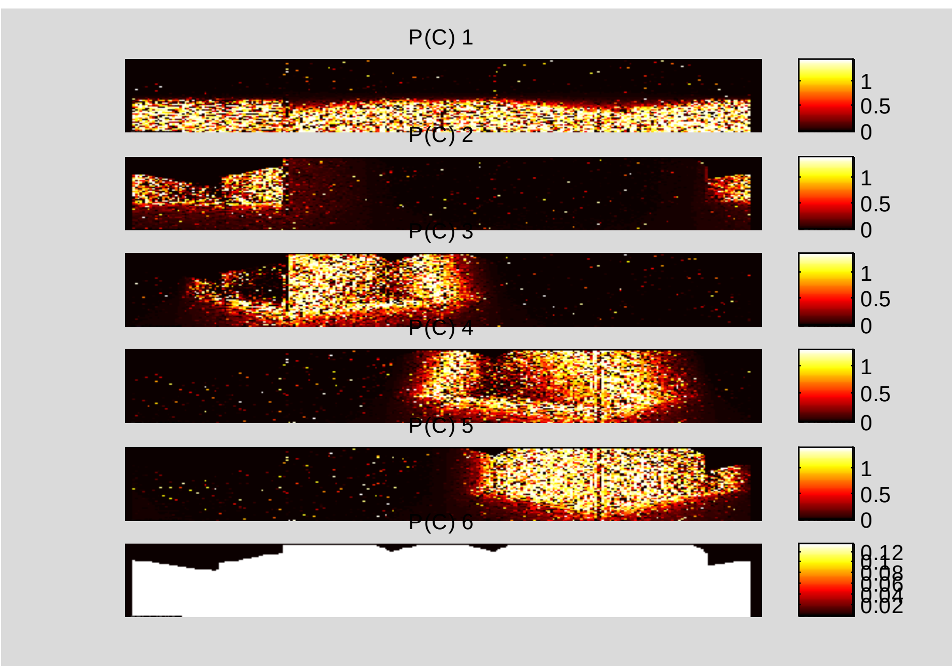


Here is a view of the images projected on to a sphere:

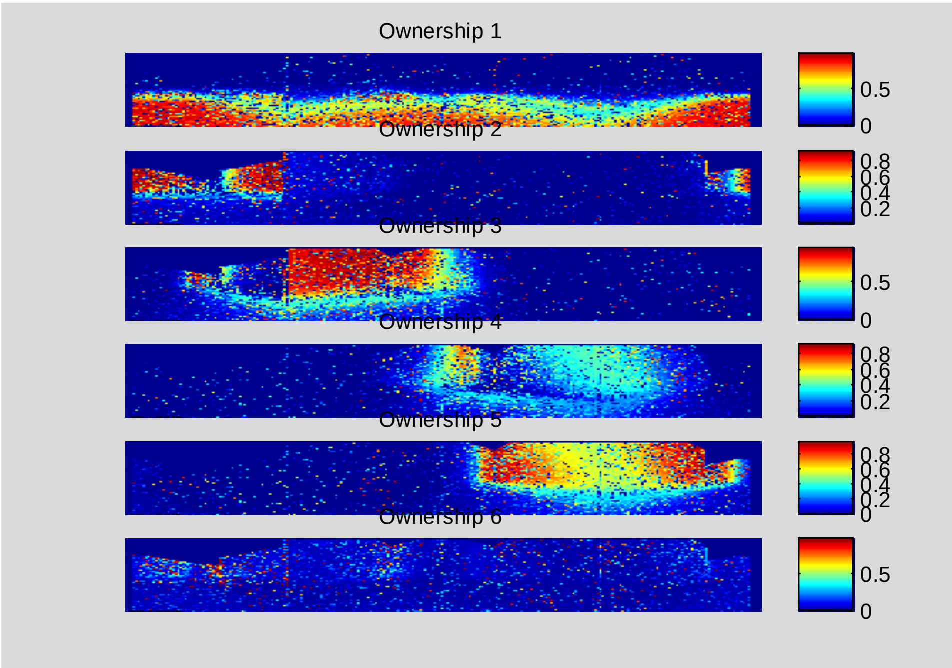


Sample Intermediate Results

Component probability for planes, $\Pr(c_k \mid p_i)$:



Ownership probability: Notice how the less likely plane number 4 has weak ownership. Also plane 2 is matching two planes, but since they are relatively distant from the camera, the match is still good.



Theses are the mixture probabilities for the five planes plus outlier:

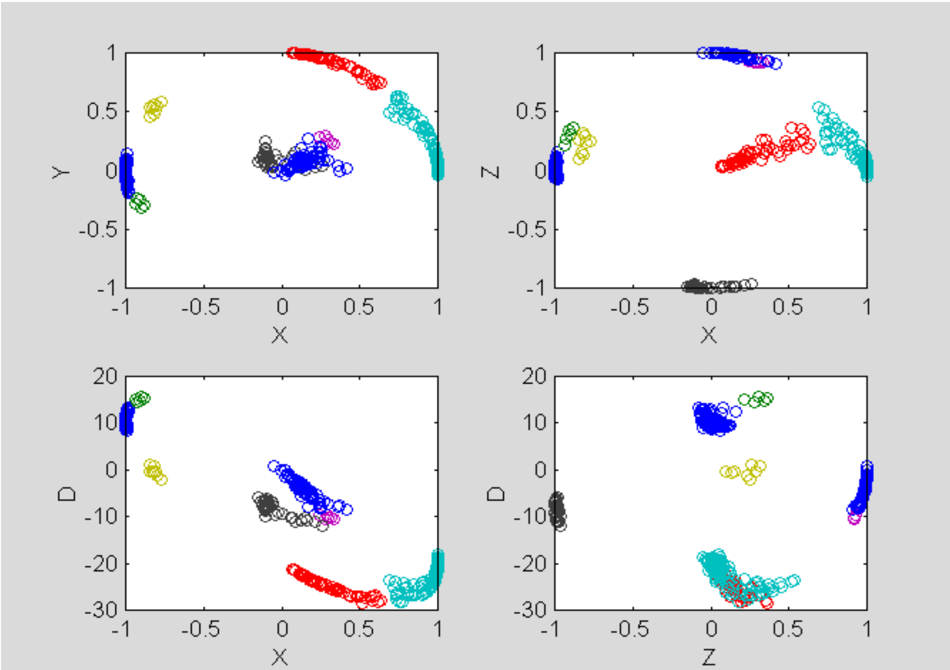
```
m =
     0.2768     0.0950     0.1926     0.1323     0.2059     0.0973
P =
    -0.0069     0.0972     0.0773    -0.2447    -0.1584
     0.0068     0.0081     0.2108    -0.0935    -0.1660
     0.5570     0.0069     0.0242     0.0432     0.0007
```

Clustering Planes

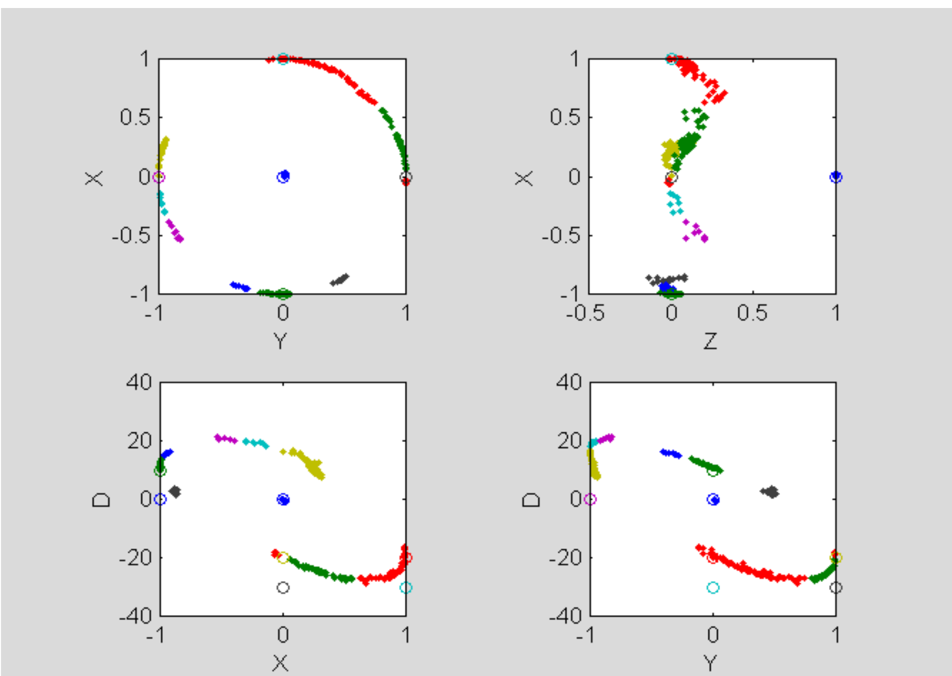
When clustering the planes from the entire image sequence, it doesn't make sense to use the planar parameter *p* from the camera coordinate system. Instead we convert the planar parameter into world coordinates separating the normal and the distance into 4 parameters which describe the equation of the plane. The distance is rescaled by one over its standard deviation to remove its scale dependence. Connected parameters are then defined as parameter vectors with distance less than a threshold. We then take the transitive closure of the connections to find the clusters. Clusters with less than a minimum membership are pruned.

Even though we have done the clustering as a post-processing step, this clustering could be run in an online fashion. Then the planes from the current frame can be matched with a list of planes maintained by a Kalman filter.

Here are the results of the clustering plotted two dimensions at a time:



Clustering results with vertical plane constraint applied, notice that there are less random planes: (The true plane parameters are dipicted by the circles.)



Results and Conclusions

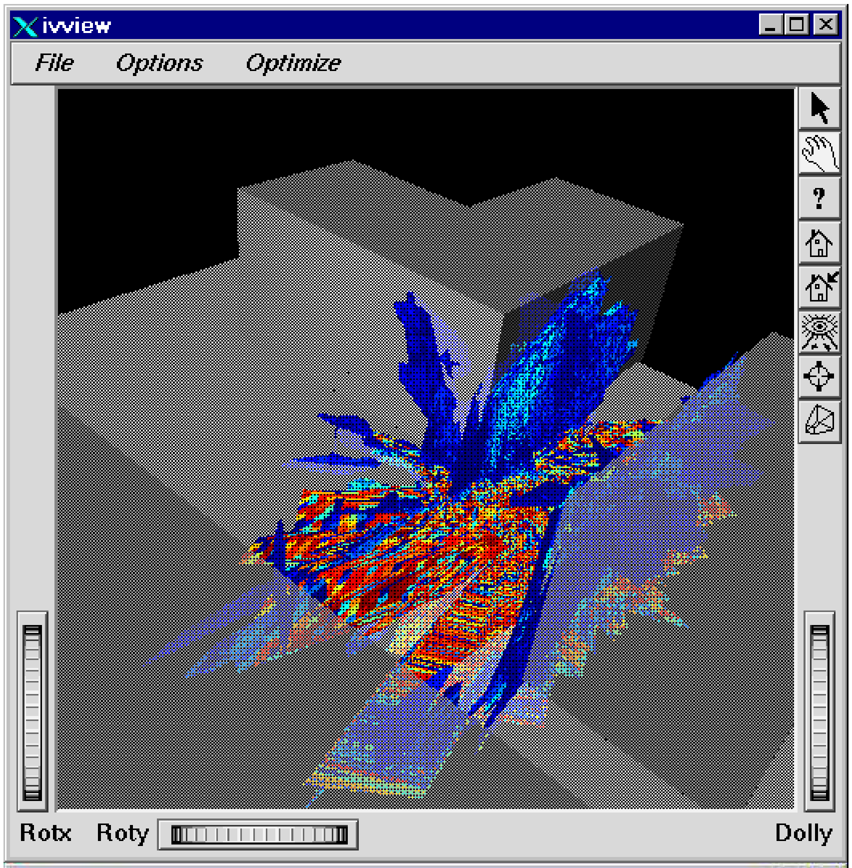
There are a number of difficulties with the algorithm. For example local minimums in EM algorithm are very easy to fall into, especially with scenes that have lots of inside corners. This is because wherever a bogus plane intersects true structure, it matches the motion constraints. When a plane is draped across the middle of an inside corner, the error gradient of the parameter is very small, so the plane gets stuck.

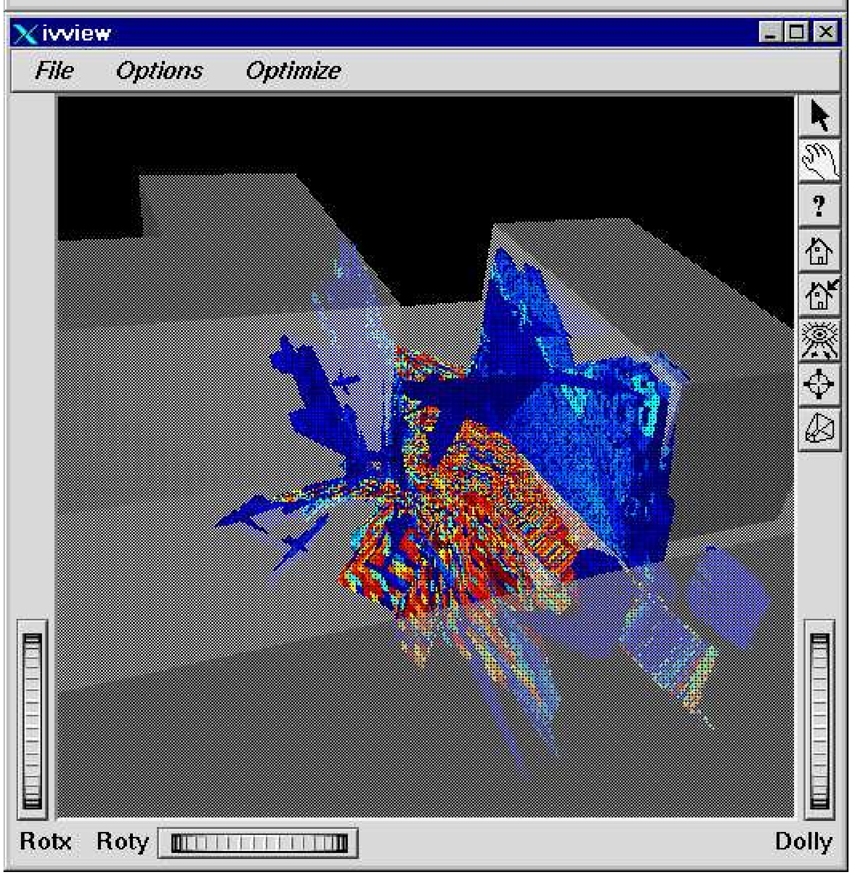
The EM algorithm doesn't take into account any spatial coherency among the pixels. There is no explicit representation of *where* the planes are, therefore determining extent of planes is nontrivial.

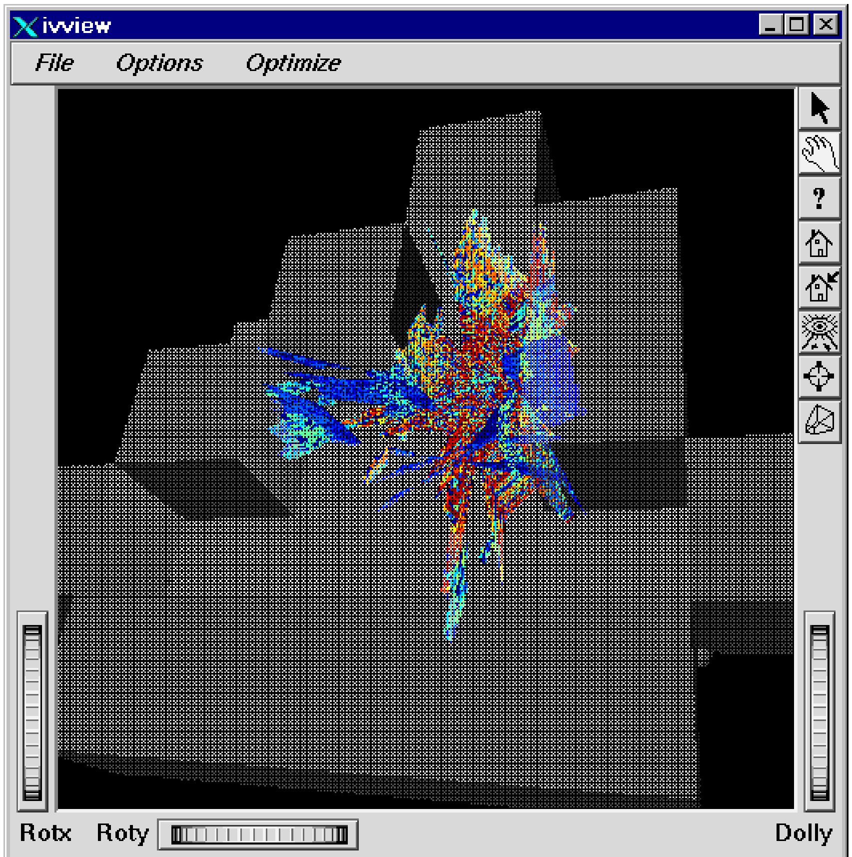
Care must be taken to tune the outlier probability, otherwise the algorithm becomes unstable and classifies everything as an outlier when it is too high.

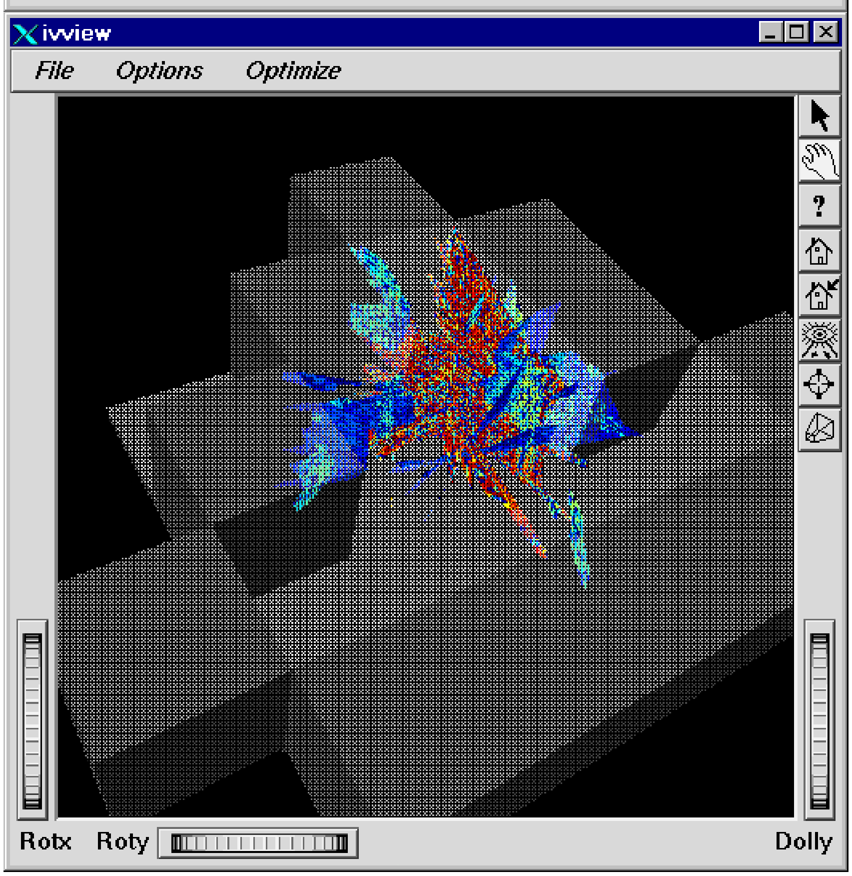
Here are the normalized square errors of the estimated planes when compared against the true planes. The highlighted cells represent the true plane that best matches the estimated plane.

	True plane							
Estimated planes	1.9999	2.4296	3.6474	5.7242	2.0351	3.6122	5.6890	0.0004
	0.6109	0.0224	8.2846	11.3797	2.4659	6.4296	9.5247	2.6084
	6.8418	9.5616	0.5674	0.3542	6.2339	1.1753	0.9621	4.8908
	2.9984	1.7828	9.1324	12.8546	1.6910	10.4398	14.1620	3.4894
	2.7572	1.4405	9.9833	13.8067	2.1445	10.5960	14.4194	3.3873
	2.8719	2.4738	5.2277	8.1324	0.4605	7.6391	10.5438	2.2555
	0.2521	0.4953	5.8044	8.0677	2.9560	3.1004	5.3637	2.0524
	1.0875	0.2207	9.1277	12.5011	2.3709	7.8442	11.2177	3.0875
	5.3363	7.8515	1.4052	1.3966	6.4931	0.2484	0.2397	4.4235
	3.2196	5.1343	2.1395	2.7314	5.3373	0.0218	0.6137	3.3463


ivview
File
Options
Optimize
Rotx
Roty
Dolly


ivview
File
Options
Optimize
Rotx
Roty
Dolly


ivview
File
Options
Optimize
Rotx
Roty
Dolly


ivview
File
Options
Optimize
Rotx
Roty
Dolly

Future Work

Use Intersection of planes to hypothesis boundaries

Use a Kalman filter to maintain list of planes in online fashion

Combine with egomotion algorithm using estimate map

Determine position by comparing distance from planes with a map

More realistic simulations with realistic textures, uncertain motion

Implement with omni-directional cameras on ARGUS platform

Bibliography

S. Akaho, "Mixture model for image understanding and the EM algorithm", TR-95-13, Electrotechnical Laboratory 1-1-4 Umenzono, Tsukuba, Ibaraki, 305, Japan, Apr 1995

M. Black, Allan Jepson, "Estimating Optical Flow in Segmented Images Using Variable-Order Parametric Models With Local Deformations", IEEE PAMI, vol 18, no. 10, Oct 1996

S. Ju, M. Black, A. Jepson, "Estimating Image Motion in Layers: The "Skin and Bones" Model", submitted paper, 1998

S. Thrun, W. Burgard, D. Fox, "A Probalistic Approach to Concurrent Mapping and Localization for Mobile Robots", *Machine Learning* and *Autonomous Robots*, 31/5, 1-25 (1998)

Y. Weiss, "Smoothness in Layers: Motion segmentation using nonparametric mixture estimation", CVPR 97, pp. 520-527