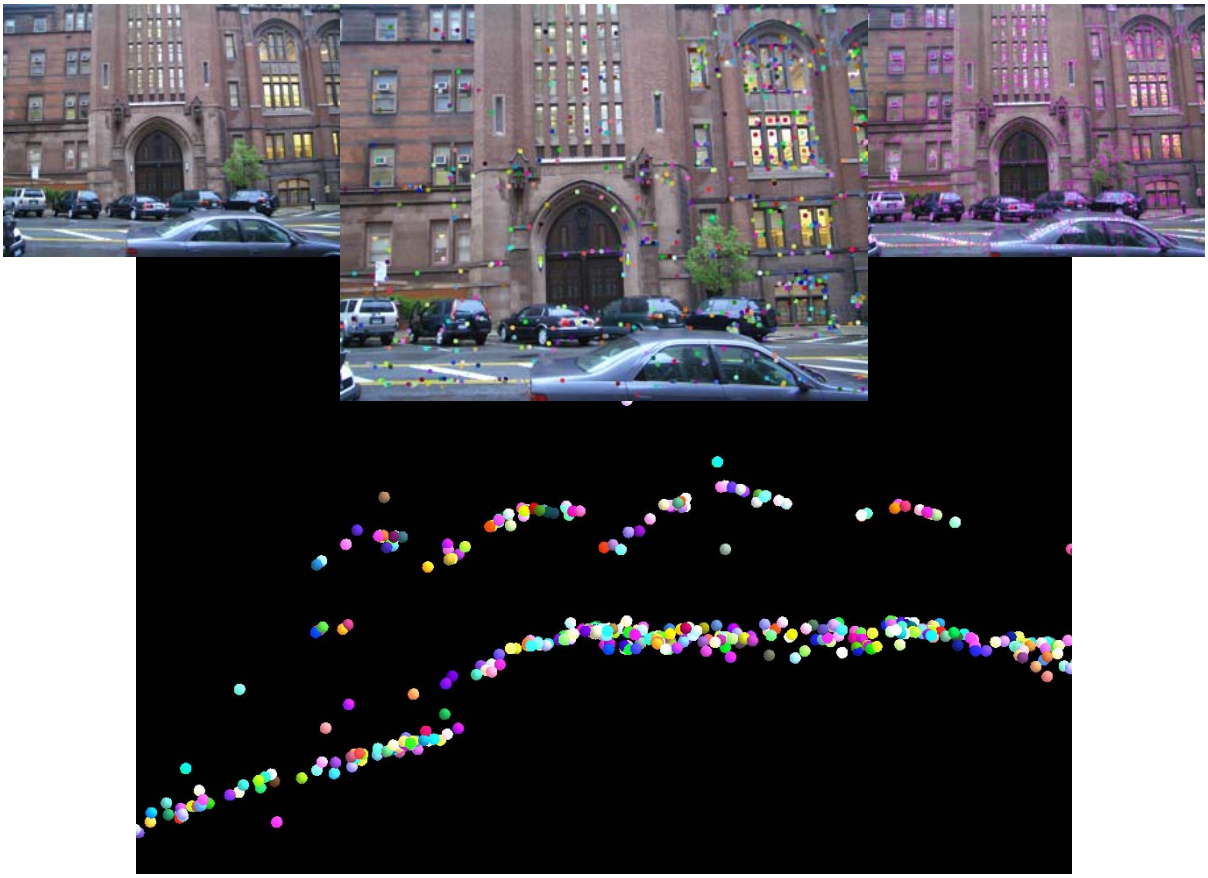
Multiple View Geometry Engine for "Street View for the People"

David Lariviere,
Kevin Chiu, Jinwei Gu, and Peter Belhumeur

Submitted as Engine Progress Report and Final Report for Computational Photography Spring 2008



Background

Problem Statement

Assume one has a database of images taken throughout a modern city, containing pictures of shops, restaurants, townhouses, skyscrapers, or anything else that one photographs from the street. Also assume that between images of the same place, point correspondences have been iteratively computed and the 3D locations of both the cameras and the locations in their photographs have been estimated.

Given a new input image, not already in the database, determine if the location photographed is already visible within any of the existing images. Given a match, determine not only where the camera was when the photograph was taken, but also where everything photographed by the camera is located in 3D space.

For a description and understanding of the intended application of the engine described in this report, please see the main project report description: "Street View for the People."

Phase One:

For the first phase of the engine, the problem begins at the genesis of the database, with the very first pair of images.

1. Given the initial two images, I_1 and I_2, determine if they are (partially) viewing the same physical location.
2. If the views of the two photographs are overlapping, establish a set of correspondences between points in one image and points in the other, that are believed to be of the same physical location.
3. Given the set of correspondences, estimate the relative locations of both the cameras and the points in 3D space.

Algorithm Overview Outline:

In order to estimate the 3D locations of both the cameras and the objects in their overlapping views, a series of computations must occur:

1. Feature Points that are easily distinguishable across multiple views are located in both images.
2. Correspondences between pairs of images, using SIFT features, are established.
3. The "Fundamental Matrix" is calculated from the initial correspondences, and possibly used to re-calculate correspondences.
4. The Fundamental Matrix is upgraded to the "Essential matrix", both by estimating the camera's internal parameters and applying mathematical constraints.
5. The Essential Matrix is used to estimate the "Camera Matrices," describing both the internal configuration (focal length, pixel aspect ratio, optical center, etc) and external position (X,Y,Z) and orientation (pitch, roll, yaw) of the two cameras.
6. 3D locations of all 2D point correspondences are calculated.

Feature Points - SIFT:

Description:

In order to establish correspondences between images indicating that a pixel in one image is of the same physical real-world 3D point as that of a pixel in another image, it is necessary to select points in an image that are both distinguishable between images and describable in a manner that does not significantly change with viewpoint.

The current engine uses SIFT (Scale-Invariant Feature Transform) for choosing and matching points between images. SIFT is an algorithm for first selecting, to within sub-pixel accuracy, and then describing distinct regions of an image in a representation (or "feature space") that is invariant to both changes in lighting, scale, and some forms of rotational motion. SIFT has proven itself as a robust descriptor for matching points between images. By default, SIFT features are 128 element vectors describing the relative "edginess" of the region directly around the feature, across several resolutions.

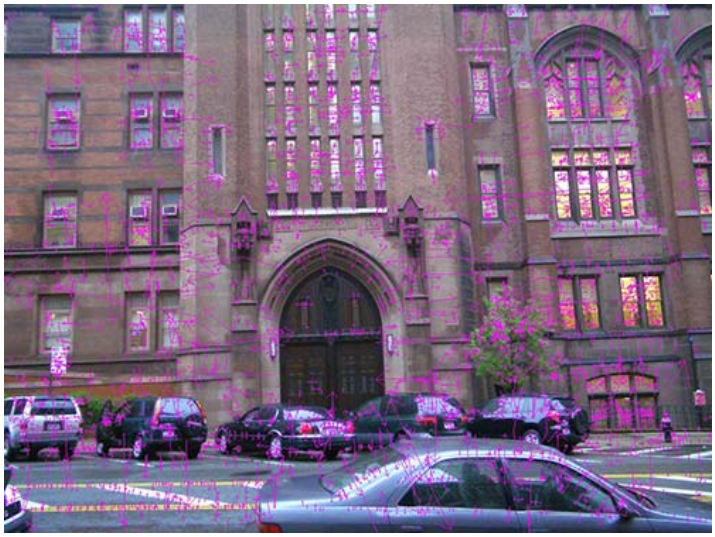


Figure Displaying SIFT features found in an image

Drawbacks:

While SIFT is often quite successful at providing a sufficient number of correct point correspondences to estimate the Fundamental matrix, it suffers from a few major drawbacks. First, SIFT features are *highly* variant to rotation, especially when viewing largely planar surfaces with a single orientation. Second, because SIFT starts by choosing points that are most often corners (at some resolution), it tends to repeatedly choose sets of points on building facades corresponding to window corners. Further, since multiple sets of identically appearing SIFT features appear in straight rows across the facade, false point correspondences between images can be established that perfectly satisfy the epipolar constraint of the Fundamental matrix, one of the primary tools in detecting false correspondences because the matrix constrains the possible region of correspondence between a point in one image to lie somewhere along a line in the other image.

Implementation:

For the engine, several implementations of SIFT were evaluated. I eventually selected an implementation written by Robert Hess of Oregon State [Hess, R.]. His implementation had several benefits:

- open source and highly configurable.
- utilized OpenCV for some image processing functions, implying increased performance for those operations.
- provided support for efficiently matching sets of SIFT features using a KD-tree
- provided support for saving and loading SIFT features to disk, allowing the ability to cache calculated SIFT features instead of recomputing on every run.

As is described in the following section, an additional matching algorithm, based upon the provided SIFT matching implementation, was implemented for this project, allowing the usage of the Fundamental matrix in constraining possible point correspondences.

Establishing Correspondences:

Given a set of SIFT descriptors for two images, I_1 and I_2, first construct a KD-Tree for the features of I_1. Then, for each SIFT feature in I_2, $f_{2,n}$, find the two closest matches in I_1. Features are compared using the Euclidian distance between their 128 element vectors. A point correspondence is accepted, if the first best match ($f_{2,m}$) is "sufficiently" better than the next best match ($f_{2,n}$):

$$\frac{dist(f_{1,i}, f_{2,m})}{dist(f_{1,i}, f_{2,n})} \geq Difference\_Threshold$$

After iterating through all features of I_1, one should be left with a set of point correspondences between I_1 and I_2.

It should be noted that at no time is the actual distance metric in SIFT space between matching points in the two images considered, only the relative distance between the first and second best match. This is an important attribute, because it implies that the algorithm can still return a significant number of point correspondences between completely non-overlapping regions, and worse, between significantly different feature points, so long as the relative difference with the next best match is large.

Iterative Correspondence Establishment Given Constraints:

For the first iteration of finding point correspondences, only SIFT features are used. Once an initial set of point correspondences is established, it is possible to estimate the relative poisitions of the cameras and thereby restrict the locations of their corresponding points.

The initial SIFT library's correspondence matching algorithm was extended such that when given a matrix which described the relative locations of the cameras based on the initial point correspondences, it then enforced the constraint when re-selecting possible point correspondences . Instead of examining only the first two matches, a larger subset of possible correspondences are returned, and all points which too strongly violate the view constraint are eliminated as possibilities.

When finding the top 25 best possible matches in I_2 for a point in I_1, it was most often the case that only a single point among all 25 satisfied the view constraint. For example, in the image pair in Figure 1, initially only 629 point correspondences were found, of which only 598 were valid. Upon the second iteration of establishing point correspondences, taking into account the Fundamental matrix, 1923 point correspondences were found, of which 1431 were valid. In addition, of the 1923 correspondences, 76% only found one match which satisfied the epipolar constraint, bypassing the need to compare relative distance ratios with the next best match.

The next section covers in detail the mathematics of the view constraints that can be estimated from a set of point correspondences.

Figure: Example Image Pair

Epipolar Geometry:

Fundamental Matrix Estimation:

At this stage one has a set of point correspondences between I_1 and I_2: (x, x'), where x is a pixel coordinate in I_1, and x' a pixel coordinate in I_2. The next step is to calculate an entity coined the "Fundamental Matrix", F, a rank 2 matrix of size 3x3 which is defined by 8 out of the 9 parameters (the ninth always normalized to 1). F defines the possible location of a corresponding point between one image to lie along a line in another image. The fundamental matrix also constrains the relative position and orientation of the two cameras, up to a perspective distortion.

The defining equation of the Fundamental Matrix is given by:

$$\mathbf{x'^T F x = 0}$$, where x and x' are corresponding points in I_1 and I_2, respectively.

The Fundamental matrix essentially *is* the epipolar constraint, meaning that given F, and a point correspondence x in I_1, it is possible to constrain the possible location of the corresponding point x' in I_2 to a line ("epipolar line") in I_1. The line (l') in I_2 is calculated by

$$l' = F x.$$

The reprojection error is a commonly used error measurement for estimating how well a given F relates supposed point correspondences, and how well a possible choice of point correspondences adhere to the epipolar constraint of F. The reprojection error is defined as the distance between the actual point correspondence x' in I_2 and the epipolar line in I_2 that is determined by F and x. Going back to the previous section, the SIFT matching algorithm was extended to filter point correspondences that had large reprojection errors.

There exist many algorithms for estimating the fundamental matrix, each with varying requirements for the number of point correspondences, complexity, and robustness to noise.

8 Point Algorithm:

The 8 point algorithm is the simplest method of estimating the Fundamental Matrix. As the name suggests, the algorithm requires at least 8 points, though if more are available, they can be used to find a minimum least squared error of F.

The 8 point algorithm involves directly estimating the best 3x3 matrix satisfying $x'^T F x = 0$ for a set of 8 or more points, the details for which are described in Multiple View Geometry. Note that the estimate of F found with the 8 point algorithm is not guaranteed to be of rank 2.

7 Point Algorithm:

The 7 point algorithm builds upon the 8-point algorithm, also taking into account an additional constraint beyond $x' F x = 0$, namely that:

$$\det | F | = 1$$

Details of the computation are left to Multiple View Geometry, and OpenCV provides an implementation. The important aspect of the algorithm is that the additional constraint yields a polynomial equation that when solved can yield either one or three possible values for F. Hartley also notes that the 7 point algorithm has the additional advantage of being rank 2. It also is not as sensitive to normalization.

RANSAC:

The RANSAC (Random Sample Consensus) method typically relies upon the 7 point method to calculate F for randomly chosen subsets of 7 points from the full set of point correspondences between I_1 and I_2. For each estimate of F, the reprojection error is calculated for all points to provide an estimate both of good an estimate the current F may be, and to filter out incorrect point correspondences that do not agree with the estimate of F.

Upon examination of the source code (cvfundamn.cpp), the OpenCV implementation of RANSAC is implemented as follows. On each iteration, 7 points are randomly chosen and used to estimate F using the 7-point algorithm. Next, for either the one or three possible F's returned by the 7-point algorithm, the reprojection error between the point and the epipolar line is calculated for both and x and x'. The point correspondences are then separated into outliers and inliers according to a user provided threshold, where both x and x' must have a reprojection error below the threshold to be classified as inliers.

Finally the implementation then checks whether the current estimate of F, as measured by the number of inliers, is greater than the best estimate's number of inliers seen so far. If it is larger, the current estimate becomes the new estimate.

The iteration loop of sampling points and estimating F terminates either when a maximum number of iterations, hard coded within OpenCV to 1000, or after a probability that the current best estimate of F is correct, given the ratio of inliers to outliers, surpasses a user provided confidence interval.

Lastly, the implementation has the optional ability to apply the overconstrained 8-point algorithm upon all of the inlier correspondences to calculate the final best estimate of F.

To summarize, the OpenCV implementation of RANSAC for estimating the Fundamental matrix has two input parameters in addition to the set of point correspondences: 1) the inlier/outlier threshold for the reprojection error, and 2) a percentage likelihood that the estimate is correct.

The RANSAC method of computing F is used in the current engine. When applying the iterative estimation method, whereby correspondencess are found in order to estimate F, which is then used to find even better point correspondences in order to better estimate F, the inlier/outlier threshold is used. Initially it is fairly large, and then is decreased once an initial estimate can be used to better filter out inliers.

Essential Matrix:

After estimating the Fundamental Matrix, the next step is to upgrade it to the Essential Matrix.

Properties of the Essential Matrix

The Essential Matrix, like the Fundamental Matrix, better relates the relative positions of two cameras and their possible correspondence by mapping a point in $I_1$to a line in I_2 along which the corresponding point must be. This relation is defined by:

$$x'^T E x = 0$$

Unlike the Fundamental Matrix, however, the Essential Matrix encapsulates the known internal parameters of the camera. The essential and fundamental matrices are related by:

$$E = K'^T FK$$

Where K' and K are the camera matrices for I_2 and I_1 respectively.

Like the fundamental matrix, the essential matrix is also of rank <= 2, however unlike the fundamental matrix, the essential matrix has equal singular values (σ), such that:

$$SVD(E) = U S V^T, \text{ where } S = diag(\sigma, \sigma, 0).$$

Lastly, note that unlike the fundamental matrix, which known up to an arbitrary perspective distortion, the essential matrix is known up to an arbitrary scale factor. As such, it is common to normalize the eigenvalue matrix to one, suchthat S = diag(1,1,0).

From Fundamental to Essential:

The most straightforward means of upgrading the Fundamental matrix is to assume or learn through calibration the internal parameters of the camera. Common assumptions on the internal parameters include zero pixel skew, square pixels, and camera center located within the optical center. For the Street View project, which initially intends to only use the iPhone cameras, it is assumed that the focus will also be roughly constant between pictures.

Camera Calibration:

In order to establish ground truth, assess the validity of the assumptions, and obtain lens distortion coefficients to correct for distortion in the images, the digital camera used for most photos in the project was calibrated using the Matlab Calibration Toolbox.

The calibration procedure reported the following internal parameters for a digital camera in "Outdoor" mode at a resolution of 2304x1728:

- Focal Length = 2333.62 +/- 7.82, 2331.13 +/- 7.85
- Optical Center = (1187.5 +/- 9.67, 816.08 +/- 13.55)
- Zero Skew
- Lens Distortion Coefficients: (-0.22526, 0.18916, 0.0012, 0.0013)

Note that given the resolution of 2304x1726, the expected optical center was (1152, 863), compared to the calibration derived value of (1187.5, 816.08).

Auto Calibration:

Auto Calibration is the process of estimating the internal parameters as part of the processing of input images, as opposed to pre calibrating with known calibration patterns before hand. An exhaustive literature of methods for auto calibration exists, differing primarily in the number of known constraints once may choose to assume beforehand.

One particular auto calibration method examined for the engine was the estimation of the focal length assuming the same focal length for both cameras, along with, zero skew, square pixels, and known optical center.

For details of the derivation, the reader is again referred to Multiple View Geometry, chapter 19. In summary, it is possible to apply the Kruppa equations (a highly non-linear set of equations relating the pair of images and their point correspondences), in order to derive the following relationship:

Let w = diag(a^2, a^2, 1), and the SVD(F) = U diag(σ_1, σ_2, 0) V^T, for the case where 'a' is the focal length (measured in pixels) for both cameras. Given both the Fundamental matrix and its SVD, the following constraint holds:

$$\frac{u_2^T w u_2}{\sigma_1^2 v_1^T w v_1} = \frac{u_1^T w u_2}{\sigma_1 \sigma_1 v_1^T w v_2} = \frac{u_1^T w u_1}{\sigma_2^2 v_2^T w v_2}$$

Where u_i is the i^{th} column of U, and v_i the i^{th} column of V. One can then either solve for the quadratic equation for a^2 explicitly, or else iteratively estimate the value of a^2 which best enforces any of the two above.

The results of the above auto calibration method were tested on several image pairs. Often the results provided fairly decent results, within 5-10% of their true value. The

results varied heavily, however, on the specific input images and set of point correspondences. Lens distortion especially seemed to play a significant negative role.

Given the significant impact that lens distortion can have on all algorithms described in this report, it was decided necessary to calibrate and pre-estimate the lens distortion of the camera. The decision is made for the initial development phase, knowing that methods of estimating the lens distortion directly from images exist.

Enforcing Rank and Singular Value Constraints

After multiplying F by the camera matrices for I_1 and I_2, the final step is to enforce the equal singular . Assuming the SVD($K'^T F K$) = U diag(σ_1, σ_2,0) V^T, the final step is to enforce the constraint that the two singular values of E should be equal, thus replacing the diagonal singular value matrix with diag(σ, σ,0), where

$$\sigma = \frac{\sigma_1 + \sigma_2}{2}$$

Camera Matrices Extraction:

Given the Essential Matrix, E, it is now possible to extract the camera matrices. The camera matrix, P_i, for Image I_i, which are of the form $P_i = K[R | t]$, where R is the rotation matrix, and t the translation vector, mapping the coordinate system of the i^{th} camera into a global reference frame. Note that t is known only up to an arbitrary scale factor, and as such, is by convention normalized such that $|| t || = 1$.

The first camera can be chosen as the world reference frame: $P_1 = [I | 0]$.

The second camera matrix can then be extracted via the following factorization.

Let SVD(E) = U diag(σ, σ,0) V^T, *and* $E = SR$ where $S = UZU^T$ and $R = UWV^T$ or UW^TV^T

where W and Z are orthogonal and skew - symmetric matrices, respectively :

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Lastly, it is possible to define P_2, given $P_1 = [I | 0]$ as:

$P_2 = [UWV^T | +u_3]$ or $[UWV^T | -u_3]$ or $[UW^TV^T | +u_3]$ or $[UW^TV^T | -u_3]$.

Finally to choose which of the four possible solutions is correct, one must test a single point, by triangulating its location (see next section) in 3D space using P_1 and P_2, for all four possibilities, and selecting the P_2 such that the 3D point is in front of both cameras, meaning the Z coordinate is positive in both P_1 and P_2's reference frames.

3D Triangulation of Correspondences:

Once $P_1$and P_2 are known, the last step is to estimate the 3D location of each correspondence.

For a given point correspondence (x, x') between two images (I_1, I_2), both 2D image points of the same 3D point X = (x,y,z), the entities are related by:

$$x = P_1X, \text{ and } x' = P_2X$$

Manipulating the equations, it is possible to form a system of equations:

$$AX = 0, \quad where\ A = \begin{bmatrix} x(p^{3T}X) - (p^{1T}X) \\ y(p^{3T}X) - (p^{2T}X) \\ x'(p'^{3T}X) - (p^{1T}X) \\ y'(p'^{3T}X) - (p^{1T}X) \end{bmatrix} and\ X = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

One method of solving for X subject to AX=0 is to apply the Direct Linear Transform method described by Hartley and Zisserman, utilizing the additional constraint that || X || = 1, which is allowed given that E and therefore P and X are all known up to an arbitrary scale.

To solve for X, one takes the SVD(A) = $U \Sigma V^T$, and X is equal to the last column of V.

This method has been implemented on a per point, and works to some degree. Unfortunately, in using the homogenous DLT method, all 3D points are individually normalized such that || X || = 1. The result is that the relative scales of points are not yet correct, and an effect similar to a lens distortion occurs in the estimated 3D locations, whereby the relative scales of points are related || x || as measured from the optical center of the frame.

One desirable aspect of the above formulation is that it immediately generalizes for N views of the same point, where each view contributes two more rows of A.

Results

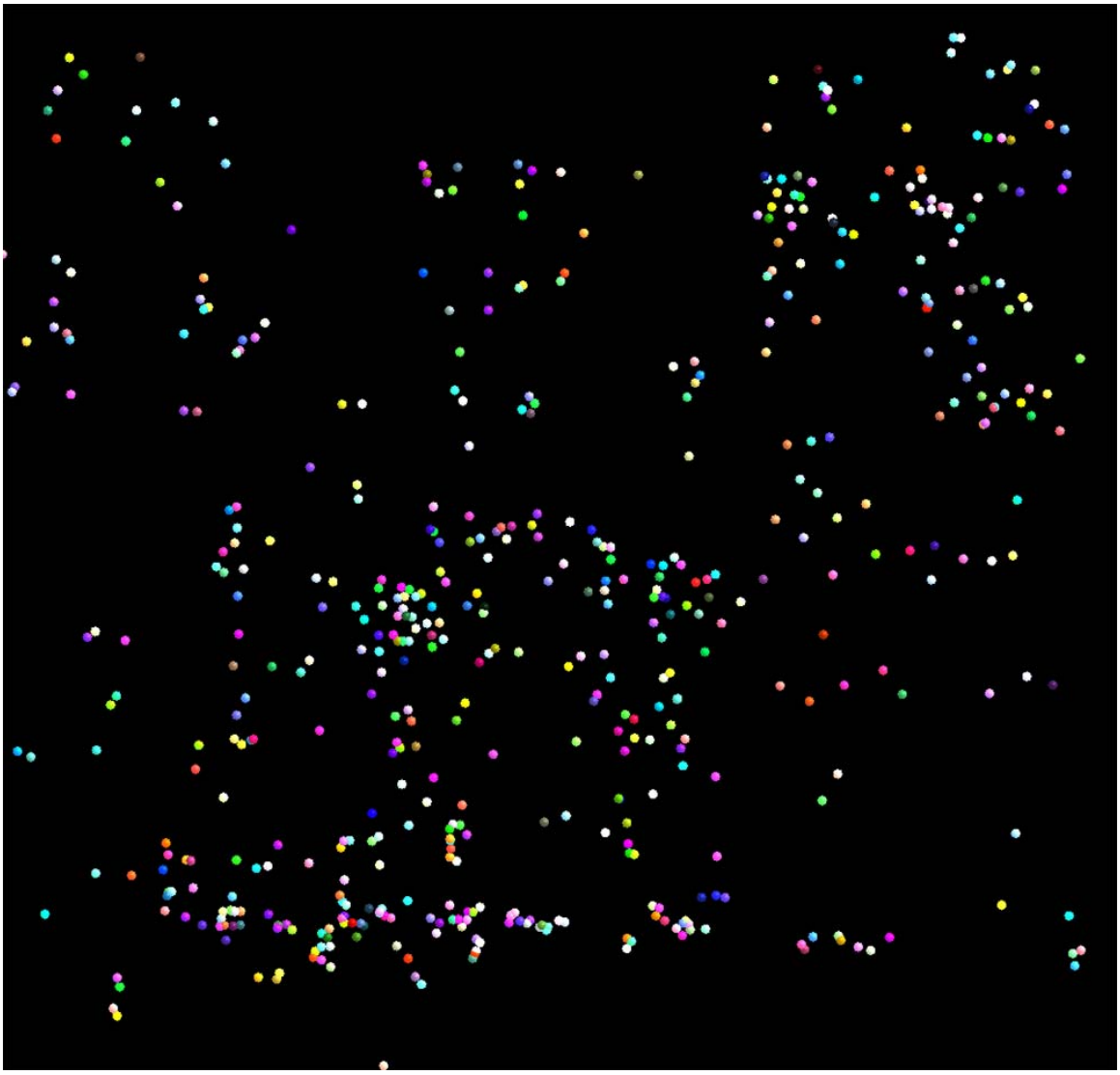
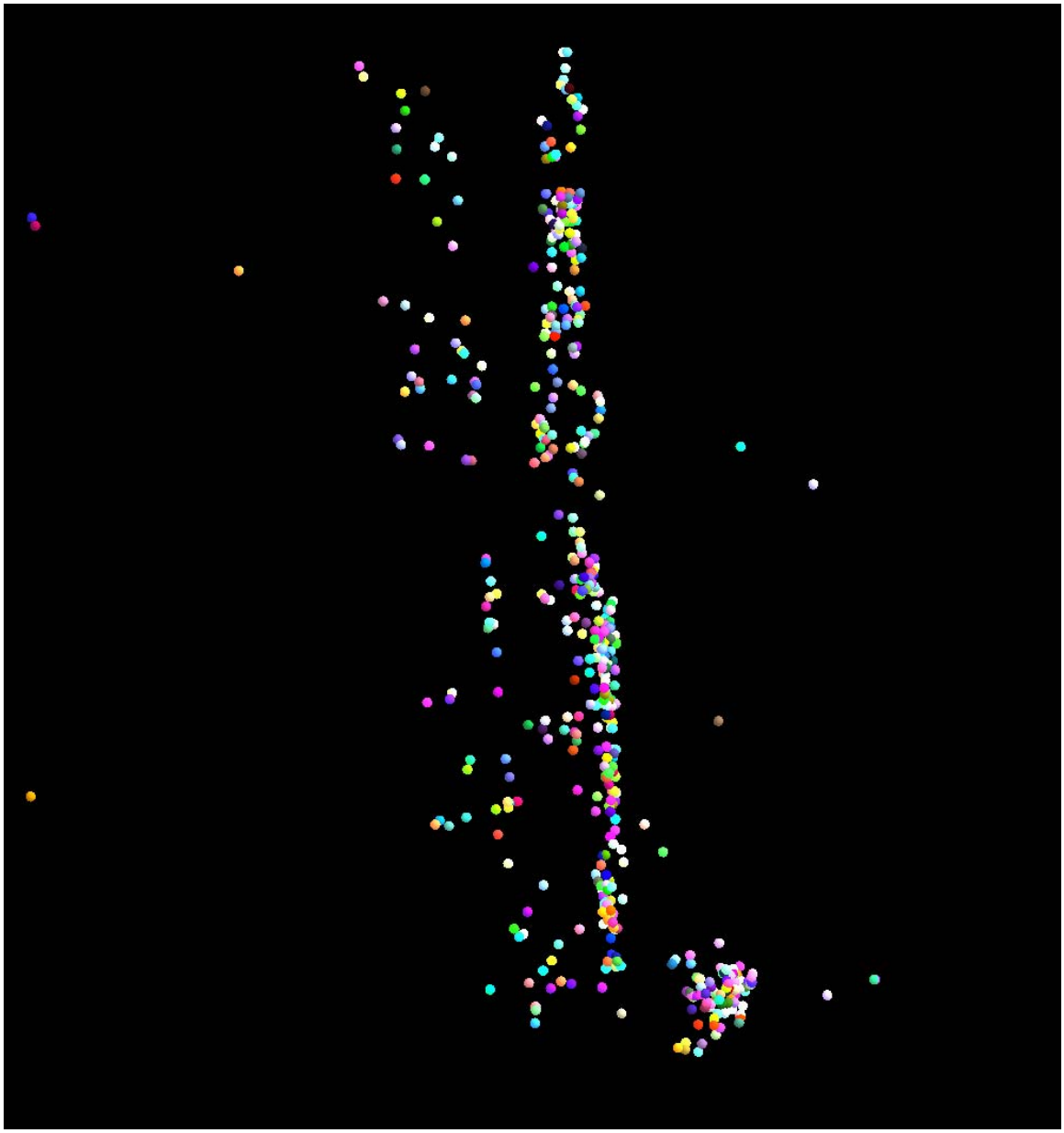
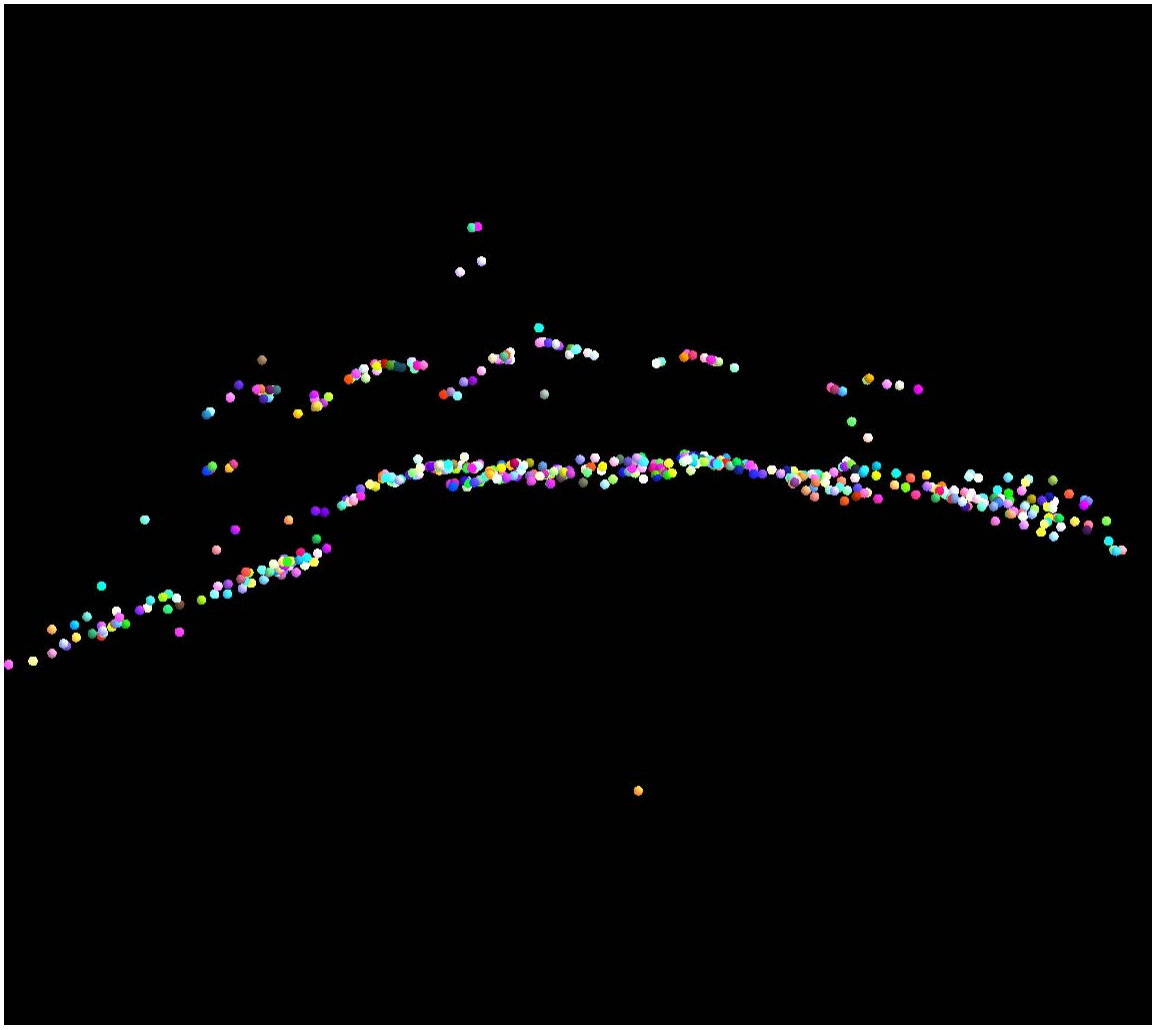


Figure: Triangulated 3D Point cloud viewed head on, with doorway in center.



Triangulated 3D Point Cloud, Viewed from the Left side. Note the multiple clear planar regions conforming to the different wall depths in the image pairs. Also note the cluster of points in the bottom right corresponding to the cars in front of the buildings.



Triangulated 3D point cloud viewed from the bottom lookup. Note the circular point clouds near the top, one per car. Also note the separate planes mapping to the different depths of the wall.

Future Algorithmic Work:

The first phase of the multiple view engine is almost done. For some point cases, it is able to construct a 3D point cloud which accurately captures the 3D structure of the scene. Unfortunately unknown degenerate cases also occur, perhaps related to the use of the homogenous DLT method for triangulating 3D points. The particular method assumes that fourth homogenous coordinate of X, w, is not zero. In the case of nearly pure translational motion, however, w may be close to zero, resulting in instability. The first step going forward is to implement a robust method of triangulation which can handle degenerate conditions.

After robust triangulation, it is then necessary to expand the engine beyond pairs of images and into sets. Several experiments have already been run, in which a set of 20 images taken at differing rotations and translations while on the sidewalk, photographing the opposing building façade. Then each image is taken, SIFT features extracted, and correspondence matching run between all other images. Following Phototourism authors' lead, the quality of the match is then taken strictly as the number of the SIFT correspondences. The simple method of choosing the largest number of matches correctly matched overlapping views in every single case, despite photographs of an extremely repetitive building façade that was difficult even for a human to locate by.

Once multiple views and estimates of cameras are integrated into a single global reference frame, global optimizations relying on bundle adjustment will be added, simultaneously refining both the camera and point locations. Several opensource libraries for performing bundle adjustment have been located, including the library utilized by Phototourism.

Additional work will likely also be required on how to best choose and match feature correspondences across an increasingly vast set of images. It is believed, and confirmed by recent publications, that SIFT features alone cannot scale in distinguish an extremely large set of scenes, especially when the majority are composed of extremely similar corner detectors along doors and windows. I am personally very interested in examining methods of detecting such repetitive structures, not only to avoid picking individual SIFT features from among them, but also in the possibility of grouping them into a higher level Feature of Features which may prove more robust at disambiguating images amongst a large dataset.

Acknowledgements:

Many thanks to Jinwei and Professor Belhumeur for the many hours of help each week in understanding perspective geometry and the related computer algorithms. A large thank you also to Kevin for coming up with the Wiki-based 3D modeling engine, getting the project started in the first place.

References:

The vast majority of technical information contained in this report is derived directly from the two books below.

Hartley, R., and Zisserman, A. Multiple View Geometry. Cambridge University Press. Copyright 2003.

Ma, Y., Soatto, S., Kosecka, J., and Sastry, S.. An Invitation to 3-D Vision. From Images to Geometric Models. Springer. Copyright 2004.

Papers:

Lowe, D.. "Object Recognition from local scale-invariant features." International Conference on Computer Vision, Corfu, Greece (Sept. 1999), pp. 1150-1157.

Software:

A bounty of different software implementations, libraries, and matlab packages were found while working on the project. Some of the most useful ones, either already, or in perhaps in the future are listed below.

Intel Corporation's OpenCV Libraries: http://sourceforge.net/projects/opencvlibrary/

Hess, R. Implementation of SIFT (used in the project).
http://web.engr.oregonstate.edu/~hess/index.html.

Bougeut, J. Camera Calibration Toolbox for Matlab. (used for calibration)
http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/index.html

Lourakis, M. Argyros, A. SBA: A Generic Sparse Bundle Adjustment C/C++ Package based on the Leveberg-Marquardt Algorithm.
http://www.ics.forth.gr/~lourakis/sba/

"Known Implementations of SIFT". *A list of various SIFT implementations.*
http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT

Zisserman, A. Matlab Functions for Multiple View Geometry.
http://www.robots.ox.ac.uk/~vgg/hzbook/code/

Mariottini, G. Prattichizzo, D. Epipolar Geometry Toolbox.
http://egt.dii.unisi.it/

VXL (Vision-*Something*-Libraries): http://vxl.sourceforge.net/

Kovesi, P. Matlab and Octave Functions for Computer Vision.
http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/index.html

Affine Covariant Features. http://www.robots.ox.ac.uk/~vgg/research/affine/