# 3D Reconstruction from Multiple Images

Shawn McCann

## 1 Introduction

There is an increasing need for geometric 3D models in the movie industry, the games industry, mapping (Street View) and others. Generating these models from a sequence of images is much cheaper than previous techniques (e.g. 3D scanners). These techniques have also been able to take advantage of the developments in digital cameras and the increasing resolution and quality of images they produce along with the large collections of imagery that have been established on the Internet (for example, on Flickr).

Figure 1: Photo Tourism: Exploring Photo Collections in 3D

The objective of this report is to identify the various approaches to generating sparse 3D reconstructions using the Structure from Motion (SfM) algorithms and the methods to generate dense 3D reconstructions using the Multi View Stereo (MVS) algorithms.

## 2 Previous Work

The Photo Tourism project [Ref P2] investigated the problem of taking unstructured collections of photographs (such as those from online image searches) and reconstructing 3D points and viewpoints to enable novel ways of browing the photo collection. As shown in the figure below, the well known example of this is the 3D reconstruction of the Coliseum in Rome from a collection of photographs downloaded from Flickr.

A few of the key challenges addressed by this project were

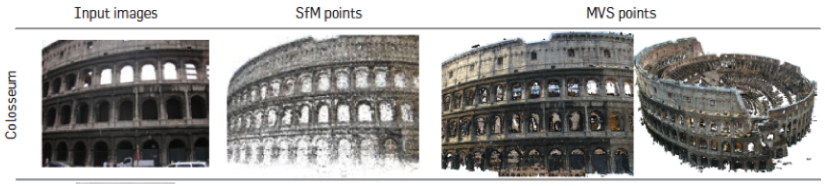


Figure 2: SFM and MVS models of the Coliseum

- how to deal with a collection of photographs where each photo was likely taken by a different camera and under different imaging conditions
- how to deal with an unordered image collection? How should the images be stitched together to produce an accurate reconstruction?
- how to deal with the running the algorithms at scale? For example, the model for the Coliseum was based on 2106 images that generated 819,242 image features.

Further elaboration of the work done in the Photo Tourism project was also described in "Modeling the World from Internet Photo Collections" [Ref P3], "Towards Internet-scale Multi-view Stereo" [Ref P4] and "Building Rome in a Day" [Ref P5].

## 2.1 Available Packages

As part of the research into previous work, a survey of the existing open-source software that has been developed by various researchers was conducted. Based on this research, it appears that the majority of the current toolkits are based on the Bundler package, a Structure from Motion system for unordered image collections developed by N. Snavely [Ref S1]. It was released as an outcome of the Photo Tourism project [Ref S1].

Bundler generates a sparse 3D reconstruction of the scene. For dense 3D reconstruction, the preferred approach seems to be to use the multi view stereo packages CMVS and PMVS, developed by Y. Furukawa [Ref S2].

Bundler, CMVS and PMVS are all command line tools. As a result, a number of other projects have developed integrated toolkits and visualization packages based on these tools. Of note are the following, which were evaluated as part of this project:

- OSM Bundler [Ref S3] - a project to integrate Bundler, CMVS and PMVS into Open Street Map
- Python Photogrammetry Toolbox (PPT) [Ref S4] - a project to integrate Bundler, CMVS and PMVS into an open-source photogrammetry toolbox by the archeological community

- Visual SFM [Ref S5] - a highly optimized and well integrated implementation of Bundler, PMVS and CMVS. Of particular note are the inclusion of a GPU based SIFT algorithm (SiftGPU) and a multi-core implementation of the Bundle Adjustment algorithm. The use of these packages allows VisualSFM to perform incremental Structure from Motion in near linear time.

Several packages are available for visualization of point clouds, notably MeshLab, CloudCompare and the Point Cloud Library (PCL) which integrates nicely with OpenCV.

# 3 Technical Approach

Given the complexity involved in creating a full scale SfM and MVS implementation from scratch, the approach taken on this project was to implement the Structure from Motion algorithms by building on top of the material covered in class and sample code found online. These results were compared with those produced by the open source packages described in Section 2.1.

## 3.1 Sorting the Photo Collection

One of the first steps involved when dealing with an unordered photo collection is to organize the available images such that image are grouped into similar views.

As described in "Building Rome in a Day" [Ref P5], their data set consisted of 150,000 images from Flickr.com associated with the tags "Rome" or "Roma". Matching and reconstruction took a total of 21 hours on a cluster with 496 compute cores. Upon matching, the images organized themselves into a number of groups corresponding to the major landmarks in the city of Rome. Amongst these clusters can be found the Colosseum, St. Peter's Basilica, Trevi Fountain and the Pantheon. One of the advantages of using community photo collections is the rich variety of view points that these photographs are taken from.

For this project, the SIFT algorithm was used to compare the images in the collection and images with a high number of correspondences were considered to be "close together" and therefore good candidates for the SfM process.

## 3.2 Feature Detection and Matching

In the Photo Tourism project, the approach used for feature detection and mapping was to :

- find feature points in each image using SIFT

- for each pair of images match keypoints using the approximate nearest neighbors, estimate the fundamental matrix for the pair using RANSAC (use 8 point algorithm followed by non-linear refinement) and remove matches that are outliers to the recovered fundamental matrix. If less than 20 matches remain, then the pair was considered not good.
- Organize the matches into tracks, where a track is a connected set of matching keypoints across multiple images.

For this project, the following techniques were investigated:

- the first approach used the SIFT algorithm to detect features in each image and then the features were matched using a two-sided brute force approach, yielding a set of 2D point correspondences.
- the second approach used the SURF algorithm to detect keypoints and compute descriptors. Again, the two-sided brute force approach was used to match the features.
- the third approach used optical flow techniques to provide feature matching. This uses a k nearest neighbor approach to matching features from image 1 with image 2. The optical flow approach is faster and provides more match points (allowing for a denser reconstruction) but assumes the same camera was used for both images and seems more sensitive to larger camera movements between images.

### 3.3 Structure From Motion

In the Photo Tourism project, the approach used for the 3D reconstruction was to recover a set of camera parameters and a 3D location for each track. The recovered parameters should be consistent, in that the reprojection error is minimized (a non linear least squares problem that was solved using Levenberg Marquardt algorithm) Rather than estimate the parameters for all cameras and tracks at once, they took an incremental approach, adding one camera at a time.

The first step was to estimate the parameters for a single pair of images. The initial pair should have a large number of feature matches, but also a large baseline, so that the 3D locations of the observed points are well-conditioned Then, another image was selected that observes the largest number of tracks whose 3D locations have already been estimated. A new camera's extrinsic parameters are initialized using the DLT (direct linear transform) technique inside a RANSAC procedure. DLT also gives an estimate of K, the intrinsic camera parameter matrix. Using the estimate from K and the focal length estimated from the EXIF tags of the image, a reasonable estimate for the focal length of the new camera can be computed.

The next step is to add the tracks observed by the new camera into the optimization. A track is added if it is observed by at least one other camera and if triangulating the track gives a well-conditioned estimate of its location. This procedure is repeated, one image at a time until no remaining image observes any the reconstructed 3D points. To minimize the objective function at each iteration, they used the Sparse Bundle Adjustment library.

The run times for this process were a few hours (Great Wall - 120 photos) to two weeks (Notre Dame, 2635 images).

### 3.3.1 SfM using Two Images

Structure from Motion techniques using a pair of images were covered in class. In particular, estimation of the fundamental matrix F from point correspondences and solving the affine Structure from Motion problem using the Factorization Method proposed by Tomasi and Kanade [Ref P1] were implemented in problem set 2.

The general technique for solving the structure from motion problem is to

- estimate structure and motion up to a perspective transformation using the algebraic method or factorization method
  - estimate the m 2x4 projection matrices $M_i$ (motion) and the n 3D positions $P_j$ (structure) from the mxn 2D correspondences $p_{ij}$ (in the affine case, only allow for translation and rotation between the cameras)
  - This gives 2mxn equations in 8m+3n unknowns that can be solved using the algebraic method or the factorization method
- convert from perspective to metric via self-calibration and apply bundle adjustment

For this project, two approaches were investigated for the scenario where the camera matrices are known (calibrated cameras):

The first approach is based on the material given in [Ref B5]:

- Compute the essential matrix E using RANSAC
- Compute the camera matrices P
- Compute the 3D locations using triangulation. This produces 4 possible solutions of which we select the one that results in reconstructed 3D points in front of both cameras.
- Run Bundle Adjustment to minimize the reprojection errors by optimizing the position of the 3D points and the camera parameters.

The second approach utilizes OpenCV and is based on the material given in [Ref B6]:

- Compute fundamental matrix using RANSAC (`OpenCV: findFundamentalMat()`)
- Compute essential matrix from fundamental matrix and K (HZ 9.12/9.13) `OpenCV: Compute` $E = K.T * F * K$
- Decompose E using SVD to get the second camera matrix P2 (HZ 9.19) (first camera matrix P1 is assumed at origin - no rotation or translation)
- Compute 3D points using triangulation (`OpenCV: no function for triangulation, code your own`)

When dealing with the situation where the intrinsic camera parameters are unknown, one can run the Self Calibration (also known as Auto Calibration) process to estimate the camera parameters from the image features. Possible techniques for Self Calibration include using the single-view metrology constraints, the direct approach using the Kruppa equations, the algebraic approach or the stratified approach. See H&Z Ch 19 [Ref B1] or SZ Ch 7 [Ref B2] for further details.

### 3.3.2 SfM using Multiple Images

With two images, we can reconstuct up to a scale factor. However, this scale factor will be different for each pair of images. How can we find a common scale so that multiple images can be combined?

One approach is to use the Iterative Closest Point (ICP) algorithm, where we triangulate more points and see how they fit into our existing scene geometry. A second approach (and the one used on this project) is to use the Perspective N-Point (PnP) algorithm (also known as camera pose estimation) where we try to solve for the position of a new camera using the scene points we have already found. OpenCV provides the solvePnP() and solvePnPRansac() functions that implement this technique.

## 3.4 Multi View Stereo

The Multi View Stereo algorithms are used to generate a dense 3D reconstruction of the object or scene. The techniques are usually based on the measurement of a consistency function, a function to measure whether "this 3D model is consistent with the input images" ? Generally, the answer is not simple due to the effects of noise and calibration errors.

Examples of consistency functions are:

- color: do the cameras see the same color? This approach is valid for Lambertian surfaces only and is based on a measurement of color variance.

- texture: is the texture around the points the same? This approach can handle glossy materials, but has problems with shiny objects. It is based on a based on a measurement of correlation between pixel patches.

One of the following two approaches are generally used to build the dense 3D model:

- build up the model from the good points. Requires many views otherwise holes appear.
- remove the bad points (start from the bounding volume and carve away inconsistent points). Requires texture information to get a good geometry.

There are usually several different 3D models that are consistent with an image sequence. Usually, we turn this into a regularization problem by assuming that objects are smooth. Then we optimize to find the best "smooth" object model that is consistent with the images. See "Multi-View Stereo Revisited" [Ref P6] and "Photorealistic Scene Reconstruction by Voxel Coloring" [Ref P7] for further details.

# 4 Experiments

## 4.1 Structure from Motion using the Fountain Dataset

For the first set of tests, the fountain dataset, a set of 24 images from the Dense Multi View Stereo datasets (EPFL Computer Vision Lab) was used.

Figure 3: Three images from the fountain dataset

This dataset was a good choice for testing with because a complete set of images was available (24 images at approx 5 deg increments), the camera matrices are available, ground truth is available and a dense 3D model available for comparison.

The reconstructions are shown below, with the estimated camera positions also shown in the sparse model. Note that keypoints that do not appear in a sufficient number of images are dropped from the model. Hence the model is focused on the fountain itself, as it is the common element in all the images.

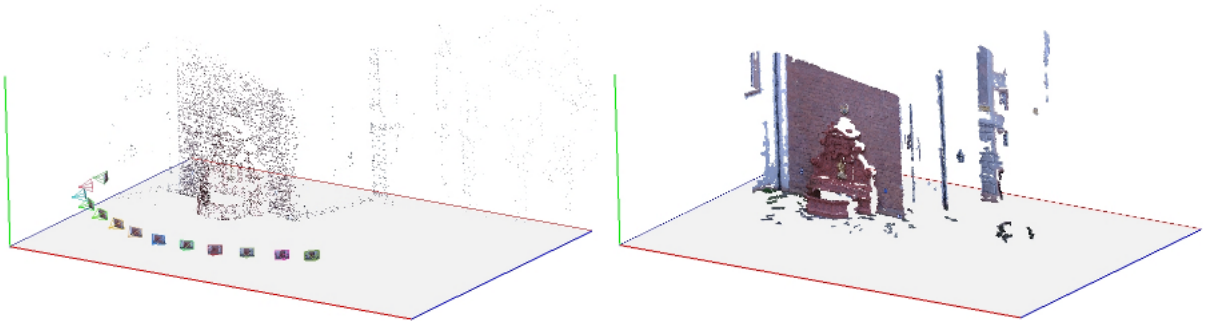

Figure 4: Sparse and Dense 3D models for the Fountain dataset

Note the gaps that appear at the top and around the sides of the fountain in the dense reconstruction. For the top of the fountain, there were some fine details there that changed appearance significantly between the frames, resulting in insufficient keypoint matches and exclusion of those points from the model. For the sides of the fountain, those points are missing simply due to the fact that there not sufficient matches found to keep those points in the sparse model. This could be dealt with by adjusting the SfM parameters, but by dropping the number of matches required for a point to be included in the model, we will find more points (and hence more noise) accumulating in the model.

### 4.2 Structure from Motion using the Topoi Loewen Dataset

The second set of tests was based on the Topoi Loewen dataset, a set of 16 images of a sculpture from the Python Photogrammetry Toolbox [Ref S4].

Figure 5: Three images from the Topoi Loewen dataset

The reconstructions are shown below, with the estimated camera positions also shown in the sparse model. In this case, we have sample images taken at two different elevations, allowing a more complete 3D model to be constructed.

For both of these datasets, sparse models were generated using the sample code and compared with the models produced by the Bundler, CMVS, PMVS chain. In general, the results produced by Bundler were more accurate and more complete than those gener-

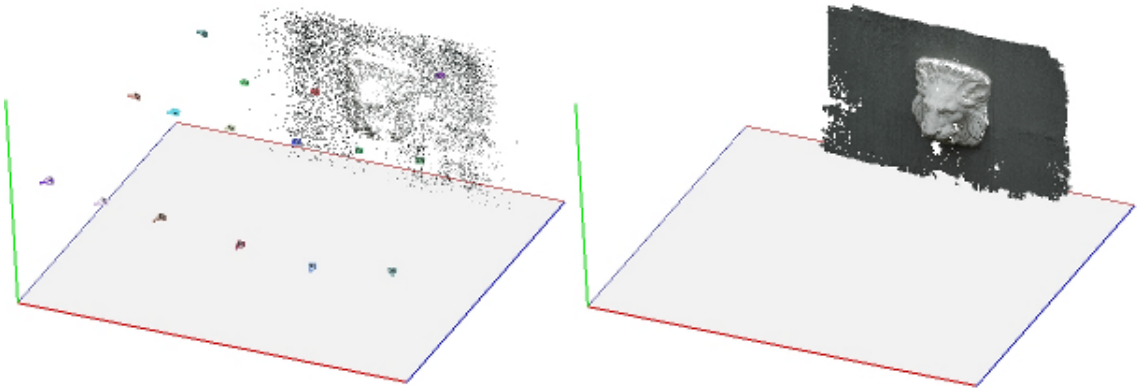

Figure 6: Sparse and Dense 3D models of Topoi Lowen images

ated using the sample code (which used simpler algorithms than those implemented in Bundler).

### 4.3 Other Datasets

Additional tests were run using the older Oxford datasets, but those results tended to be not as good as those above, due to the smaller number of images in each dataset.

Some sample datasets were also created using images downloaded from Flickr (Tower of Pisa, Coliseum, Sydney Opera House). However, a large number of images were needed to get decent results, which resulted in long processing times. In this case, we also had to deal with images from different cameras, which made metric reconstruction difficult without self-calibration capabilities.

## 5 Conclusions

The Structure from Motion and Multi View Stereo algorithms provide viable methods for building 3D models of objects, buildings and scenes. The key issues with the algorithms are they are fairly CPU and memory intensive, especially when trying to do reconstruction at large scale as was done in the Photo Tourism project.

The Visual SFM implementation was found to be much faster than the other toolkits (a few seconds per image vs many seconds to a few minutes per image for others). This was mainly due to use of the GPU for SIFT and the implementation of a multicore bundle adjustment algorithm.

More work is required to determine how to reconstruct complete 3D models and transform them into meshes that can then be imported into standard 3D modelling software.

## 6 References


P1: C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. IJCV, 9(2):137-154, November 1992.

P2; N. Snavely, S. Seitz, R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. ACM Transactions on Graphics (2006)

P3: N. Snavely, S. Seitz, R. Szeliski. Modeling the World from Internet Photo Collections. In IJCV Vol 80 (2008)

P4: Y. Furukawa, B Curless, S Seitz, R Szeliski. Towards Internet-scale Multi-view Stereo. In CVPR (2010)

P5: S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S Seitz, R Szeliski. Building Rome in a Day. In CACM Vol 54 (2011)

P6: Michael Goesele, Steven M. Seitz and Brian Curless. Multi-View Stereo Revisited, Proceedings of CVPR 2006, New York, NY, USA, June 2006.

P7: Photorealistic Scene Reconstruction by Voxel Coloring S. M. Seitz and C. R. Dyer, Proc. Computer Vision and Pattern Recognition Conf., 1997, 1067-1073.

S1: Bundler: `https://www.cs.cornell.edu/~snavely/bundler/`

S2: CMVS/PMVS: `http://www.di.ens.fr/cmvs/`

S3: OSM Bundler: `https://code.google.com/p/osm-bundler/`

S4: Python Photogrammetry Toolbox: `http://www.arc-team.homelinux.com/arcteam/ppt.php`

S5: VisualSFM: `http://ccwu.me/vsfm/`

B1: Multiple View Geometry, Hartley and Zisserman

B2: Computer Vision: Algorithms and Applications, Szeliski

B3: Computer Vision: A Modern Approach, Forsyth and Ponce

B4: Learning Open CV, Bradski and Kaehler

B5: Programming Computer Vision with Python, J. E. Solem

B6: Mastering OpenCV with Practical Computer Vision Projects, Baggio et al.