Construction and exploitation of a 3D model from 2D image features

Karl Ni[1], Zachary Sun[12],Nadya Bliss[1], and Noah Snavely[3]

[1]MIT - Lincoln Laboratory, 244 Wood Street, Lexington, MA, USA;
[2]Boston University, One Silber Way, Boston, MA, USA;
[3]Cornell University, Department of Computer Science, Ithaca, NY, USA

ABSTRACT

This paper proposes a trainable computer vision approach for visual object registration relative to a collection of training images obtained a priori. The algorithm first identifies whether or not the image belongs to the scene location, and should it belong, it will identify objects of interest within the image and geo-register them. To accomplish this task, the processing chain relies on 3-D structure derived from motion to represent feature locations in a proposed model. Using current state-of-the-art algorithms, detected objects are extracted and their two-dimensional sizes in pixel quantities are converted into relative 3-D real-world coordinates using scene information, homography, and camera geometry. Locations can then be given with distance alignment information. The tasks can be accomplished in an efficient manner. Finally, algorithmic evaluation is presented with receiver operating characteristics, computational analysis, and registration errors in physical distances.

Keywords: Structure from Motion, Object Detection, Registration, Bundle Adjustment

1. INTRODUCTION

With increased web usage over a more diverse demographic, online media availability in the United States has risen 40% in the past year alone. (See the Nielsen Media Wire report, Sept 2, 2009). As part of the media explosion, the quantity of images has grown and become increasingly accessible due to digital photo posting websites. The accessibility has promoted opportunities for exploitation due to inherent image information now readily available.

Among the applications that have begun to realize such potential, automated tagging algorithms in photo storage sites are currently implemented (See Google's latest Picasa$^{\text{TM}}$ technology). Tagging humans is just one example of the common computer vision problem of image labeling. It provides the viewer with the answer to one of many questions that he or she might ask: "Who is in this picture?" Another, more general, tagging question the user may pose is, "What is this a picture of," to which solutions are often ingrained in semantic annotation and image retrieval techniques.[5] Such techniques have grown considerably because they address the experience defined by online image searching. It is also possible to ask "where" an image was taken. Location matching is a concept that 3D modeling techniques[13] have focused on by integrating advances in Structure from Motion (SfM), bundle adjustment, and feature representation.[7] The question we ask takes a step further and involves integration of aforementioned object annotation and SfM technologies, i.e the "where" and "what". While multi- and single-view geometry have only previously been applied to images collectively, we can expand the scope to items within the image; that is, "Where are specific objects in this image?"

The problem translates to finding the 3-D real-world coordinates of an object of interest given a single photograph. With adequate recognition software (to be discussed), the administration of the proposed algorithm spans a broad spectrum of problems. "Where was I in this photo"; "Where exactly did I leave my keys"; "Where did I park my car"; etc. Eventually, real-world coordinates, if aligned absolutely with physical landmarks and terrain a priori, can be used to geo-register objects on a global scale, a topic immensely useful to military surveillance, where target identification and geo-registration are valuable assets.

As expected, the ideas presented in this paper are intimately related to *object detection* and 3D *image registration*. Combined, we propose an *object registration* algorithm. The processing chain in object detection and image registration are similar in that both require feature identification. For faces, cars, and other objects detectable through a mixture of

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government

experts and boosting cascades,[11] prominent basis functions for features have included Haar Filters. Other object detection algorithms[1] have built features using Gabor functions. Likewise, features in Snavely et al.[13] register images collectively through SfM using scale invariant feature transforms (SIFT), originally proposed by Lowe et al.[7]

As a learning-based algorithm, we require image collections, or training sets, to generalize to unseen data points. To register objects we need sets to resolve:

1. Location
2. Objects of interest
3. Objects not of interest

Item 1 involves the same bank of co-located images used in SfM.[13] Items 2 and 3 refer to images of the objects that are and are not of interest, following often used conventions.[11] In practice,[16] their attributes include the standard 24×24 pixel sizes with real-time realizations, which can be found in digital cameras and tracking applications (see US Patent 7315631). Item 3 may (and in our case does) overlap with Item 1.

The proposed algorithm is described in the remainder of this paper with the following sections. Sec. 2 reviews the collection of all three training sets and the building of our 3-D model through SfM. As there may be overlap between sets, we also discuss related issues and computational savings. Next, our major contributions are discussed with image registration in Sec. 3 as the first substep and object registration in Sec. 4 as the second substep towards our goal. Finally, a number of experimental results and conclusions can be found in Sec. 5 and Sec. 6.

2. BACKGROUND AND SETUP

Testing and application of object registration techniques occur in three serial phases: image registration, object detection, and object registration. As shown in Fig. 1(b), each of the stages feeds from a previous stage. Applications of two of the phases require training and initialization.

Like most algorithms, the proposed algorithm generalizes with large amounts of data to build point clouds and determine object detection parameters. The two sets can be collected independently, but to improve performance (and some added supervision), we can overlap training images between the two stages. Training object detection algorithms not only requires numerous images of the objects of interest but also of background image patches that are *not* of interest, i.e. the negative set, and could potentially generate false positives. Should we introduce such redundancy during training, test images matching to the generated 3-D point cloud are more likely to have the same image structures, which naturally implies fewer false positives. In a sense, we are telling the object detection algorithm exactly what image texture patches to ignore.

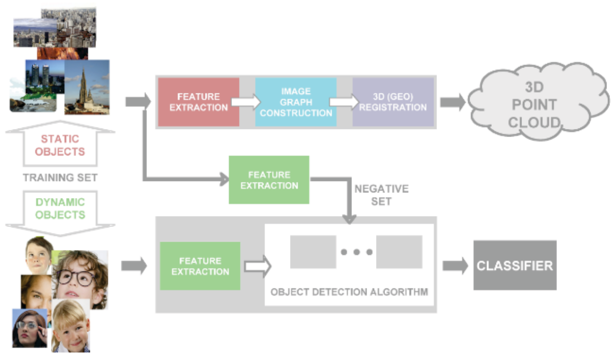
Further setup of the proposed algorithm relies heavily on several well-established techniques in the computer vision community. The overall initialization diagram is shown in Fig. 1(a), which includes the 3-D point cloud generation and detection parameter training. For the two stages requiring training, the primary techniques are embedded heavily in 3-D geometry[13] and computer vision.[11,16] This section reviews their contributions and how they fit into our framework.

2.1 3-D Scene Representation

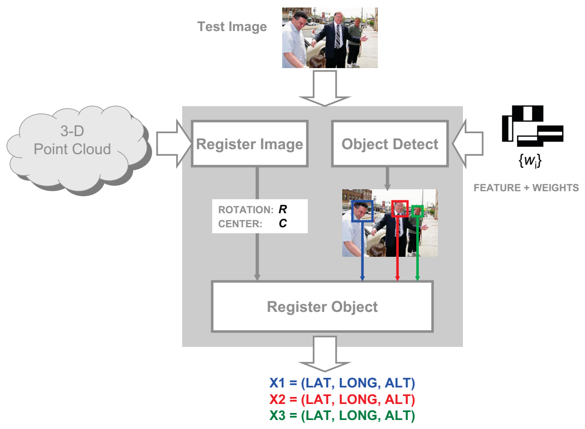
The goal of this initialization phase is to create a 3-D point cloud whose individual elements describe the 3-D point coordinates of features from a set of images. It is a collective data filtering process that simultaneously solves for scene structure while determining camera parameters and projection matrices for each image. The implementation is primarily taken from Noah Snavely's Photo Tourism work.[13] The component modules extract features from images, find the correspondences across images, and then run SfM on the matching information. The process is shown in the upper subdiagram of Fig. 1(a) with the stages consecutively labeled:

1. SIFT Feature Extraction[7]
2. Approximate Nearest Neighbor Matching[3]
3. Structure from Motion with Bundle Adjustment

The SIFT keypoint detector[7] is used because of its invariance to image transformation and robustness in matching. Not only does SIFT return a list of keypoint locations within an image, but also gives a n-dimensional descriptor vector that



(a) Overall Training Procedure



(b) Overall Testing Procedure

Figure 1. Sample of Pictures Used

can be used for matching. Lowe demonstrates that using 128-dimension vectors typically gives the best balance between speed and performance when trying to reliably match images. For typical 8MP images, SIFT returns around 8-10 thousand features. Depending on the application, other feature types[10] could also be used here.

Once all of the features have been extracted, correspondences need to be established. For each pair of images, keypoints are matched by finding the nearest neighbor vector in the corresponding image, which is traditionally defined in Euclidean L_2 space. To speed up matching, Arya and Mount's approximate nearest neighbor (ANN) package[3] can be exploited. For image I and J, ANN builds a kd-tree of the features in image J and then queries the tree for the best match of each feature in image I. Instead of defining a valid match by thresholding the distance, valid matches are determined using a ratio test:[7] find the best two nearest neighbors in image I with distances d_1 and d_2 where $d_1 < d_2$. Accept as a match if $\frac{d_1}{d_2} < 0.6$.

Photo Tourism[13] computes voxel locations of reliably matched features, while estimating the camera parameters. Tracks of matching features are then built to be triangulated later. Photo Tourism initializes with a reconstruction of two images by triangulating the tracks of two images with the most number of images and largest baseline. After reconstructing the first pair it then proceeds to add images that observe a large set of what has already been reconstructed and adds in new tracks that are observed by already reconstructed images. Like other SfM projects, Photo Tourism also runs a bundle adjustment on the reconstructed scene after each image is added.[14]

Each voxel represents a track of features from multiple images. Each feature in a track (because of the matching criterion) can be considered nearly identical. As a result, each voxel has its own set of descriptor vectors that can be used to match with the new image. For computational efficiency in image matching and reduction of noise, we take an average of the feature descriptors for a given voxel into a *representative descriptor*. (There are other data reduction techniques that could work.) The descriptors are then stored in the same format as Lowe keypoint files.

While we no longer have a concept of scale nor an absolute orientation, the storage format can be modified to reduce the four keypoint values (scale, orientation, location) down to just three (location). Given a representative set of feature descriptors, the final kd-tree can be built with representative features using Arya and Mount's ANN package.[3]

2.2 Object Detection

Depending on the application, detection is intimately associated with the type of features chosen. While SIFT features yield excellent results as a means for discriminating location, one fortuitous consequence is that they are not able to discern "interesting" foreground objects. For example, in preliminary experiments, 3-D scenes were built with an abundance of stationary cars and trucks as a significant portion of the images. No or very few meaningful features from any vehicle were involved in the 3-D point cloud reconstruction.

As it turns out, faces, humans, cars, and a number of other potential foreground objects of interest reconstruct very poorly in three dimensions on a global scale. Furthermore, should the object move, its features are even less likely to *match* with other images. The conclusion is that SIFT features of foreground objects will *not* be good discriminants of its location.

This is fortunate because the objects themselves may not remain stationary. (Local motion between images introduces system noise that image registration is sensitive to.) Therefore, at least for faces and cars, the presence of objects in an image will not impact the overall image registration performance, and hence can be done *independently*.

Though it is conceivable to register an arbitrary type of object, the results in Sec. 5 reflect human face registration data. The features used in the most cited face detection algorithms[11] are Gabor functions and Haar Wavelets.[8] Additionally, recent works have suggested that skin tone is relatively robust to changes in luminance. Consequently, for face detection, the proposed algorithm combines simple skin tone template matching after an openCV real-time implementation of Viola and Jones.[16]

3. INCREMENTAL IMAGE MATCHING

After the initialization phases of Fig. 1(a), we are presented with both 3-D information and object of interest (face) detection information. This section is concerned with the image registration box in Fig. 1(b), where the two inputs are the test image and the 3-D point cloud setup by Sec. 2.1. The goal is to produce a single rotation matrix and translation vector that represents where the camera was located and in what direction it was pointing at the time the picture was taken.

Ideally, SfM and bundle adjustment can decide whether or not an image belongs to a set by incorporating it into the set, recomputing the correspondences, and observing how well it correlates. Yet, to do so is both unnecessary and prohibitively expensive because SfM[13] and related algorithms focus on *joint optimization* of images within the set. We are, instead, concerned with incrementally registering and matching a single test image without augmenting the point cloud, foregoing any *online* training.

Because the reconstructed point cloud is based on the matched features from the images themselves, this lends itself to a different way of matching. Rather than matching to images, the image can be matched to the point cloud. That is to say, we match to the representative feature introduced at the end of Sec. 2.1 instead of the collective set of images.

Let $\{f_i^{(T)}\}$ be the set of features obtained in the test image T. We search for the smallest and second smallest distances, d_1 and d_2, and test the ratio $\frac{d_1}{d_2} < 0.6$ to find matches defined by (1):

$$\begin{aligned} X_{match,i} &= \underset{F_j}{\operatorname{argmin}} \|f_i^{(T)} - F_j\|^2 \\ d_1, d_2 &= \min_{F_{j,1},F_{j,2}} \|f_i^{(T)} - F_j\|^2 \end{aligned} \qquad (1)$$

4. PROJECTIVE OBJECT REGISTRATION

In this section, we address the task of registering an object to absolute and standard metrics, the bottom box in Fig. 1(a). Object registration requires several inputs, which can be fed directly from both object detection and image registration modules. From the image registration module (Sec. 3), object registration takes matching image information (in the form of matched features) and camera information (i.e., camera rotation matrix R and coordinates C). From the object detection module (Sec. 2.2), object registration takes the pixel locations of an object's bounding box. Object registration determines real-world measurements (3-D coordinates) that describe the location of the object.

The algorithm begins by taking the image plane P_f that contains ranging information from an "optimal" feature f based on accuracy and preciseness critera in Sec. 4.1. Once real-world coordinates of f^* can be attained, single-view camera techniques extract individual object registration information in Sec. 4.2. Finally, geo-registration can occur by aligning distances to known absolute positions in Sec. 4.3.

4.1 Optimal Feature Selection

Registering objects requires a sense of *positioning* in the generated 3-D world. An abundance of such information is presented as matched features, and a single point, the optimal point, can register a detected item relative to the point cloud. There are numerous factors that we consider in choosing such a point: re-projection error, camera distance, radial angle, and the number of correspondences with the associated feature. The most accurate and precise point will yield the best 3-D object coordinates.

Accuracy depends on re-projection error as well as the number of feature correspondences per related voxel. The assumption is that any chosen feature must foremost be both correct (re-projection error) and "interesting" (in terms of the number of images in which a feature is present and matches).

Meanwhile, *precision* depends on camera distance and radial angle from the center of the image. Information could potentially be lost in resolution as images are traditionally represented on a regularly sampled grid. That is, a pixel corresponding to a far away voxel represents a larger physical distance than one that corresponds to a closer feature. Thus, the pixel to information content is higher in the latter as is its resolution. Likewise, pixels on the edge of an image may not yield as precise results as those closer to the center.

Because 3-D construction is done a priori, obtaining the "best" feature is simply a *discrete* minimization over several number of variables and hence can be solved iteratively. As a pre-processing step, we threshold the re-projection error so that only handful of points are relevant. Let S be the set of image voxels that have been matched with the image I. The optimal feature f_i^* can be determined by:

$$i^* = \underset{i:x_i \in S}{\operatorname{argmin}} \lambda_1 \Delta x_i + \lambda_2 V^{-1}(x_i) + \lambda_3 ||x_i|| + \lambda_4 ||X_i - C||. \tag{2}$$

Here, x_i denotes the 2-D pixel location of the i^{th} extracted feature, X_i the 3-D matched voxel location, C the camera location, and $V(\mathbf{x})$ the feature count per voxel. The terms in (2) can be summarized for the i^{th} candidate feature as:

1^{st} term:	Δx_i	Accuracy Metric:	Reprojection Error
2^{nd} term:	$V^{-1}(x_i)$	Accuracy Metric:	Inverse feature per voxel count
3^{rd} term:	$\lvert\lvert x_i \rvert\rvert$	Precision Metric	2-D Offset from center pixel
4^{th} term:	$\lvert\lvert X_i - C \rvert\rvert$	Precision Metric	3-D Distance from Camera

Determining λ_i values involves understanding the relationships between the metrics. For example, it is not unrealistic to assume that the positioning of landmarks are generally oriented alongside the principle plane (the plane orthogonal to the camera pointing direction). With this assumption, should we wish to, we can reduce a degree of freedom for λ_3 with respect to λ_4. Let α be a single pixel angle given by:

$$\alpha = \frac{2}{\text{image size}} \times arctan\left(\frac{\text{film size}}{2 \times \text{focal length}}\right), \tag{3}$$

then the distance that pixel subtends, which we wish to minimize, is directly relatable to a features' distance from the camera center and its angular offset:

$$p_i = ||X_i - C|| \left\{ sin\left(\alpha x_i\right) - sin\left(\alpha(x_i - 1)\right)\right\}. \tag{4}$$

Substituting the precision terms, the optimal feature can then be written as:

$$x^* = \underset{x_i \in S}{\operatorname{argmin}} \lambda_1 \Delta x_i + \lambda_2 V^{-1}(x_i) + \lambda_3 p_i. \tag{5}$$

4.2 Relatively Registering the Object

As a geometric-intensive study, our computer vision notation follows that of the well-written and cited source, Hartley and Zisserman.[6] The ideas presented in this subsection determine, in single-view camera geometry, the real-world coordinates of an object from an image that is known to have been taken of a particular scene. The basic steps that we reference or derive to meet our goal can summarized as follows:

1. Take the optimal feature x^* from Sec. 4.1 and find its distance along the camera point direction to our camera coordinates C
2. Find the plane P_f through X_f^* that is parallel to the image plane
3. Back project the detected object's coordinates onto P_f
4. Using similar triangles and assumptions on the object of interests' size, derive its location

From meta-data, the camera intrinsics can be found and labeled as K, and from software developed for Sec. 3, the rotation R and translation t can be applied to produce a general projective camera model P. The forward projection matrix $P = K[R|t]$ maps world points in X to image points $\mathbf{x}$ according to $\mathbf{x} = PX$.

In Steps 1 and 2, once selected, determining the horizontal distance to X_f^*, the distance along the camera pointing axis, is simply the third coordinate of PX_f. Let us call this distance d_f. The full descriptor of the pointing vector to the feature plane P^4 is thusly known with normal vector, m_3, determined by

$$m_3 = \begin{bmatrix} R^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}^T \\ 0 \end{bmatrix} \tag{6}$$

The pseudo-inverse (Moore-Penrose solution) of projective matrix P is $P^+ = P^T \left(PP^T\right)^{-1}$. The ray that connects a world-point and the camera is determined by the join of the camera center $C = -KRt$ and the feature point $P^+\mathbf{x}$. As it happens, $C \in \mathcal{N}(P)$, the null-space of the forward projection matrix, and $PC = \mathbf{0}$. Thus, adding a vector $P^+\mathbf{x}$ to any scalar multiple of C will always project back onto $\mathbf{x}$ since $PP^+ = I$. The set of world-points that project onto the image at $\mathbf{x}$ is thus given by:

$$X(\eta) = P^+\mathbf{x} + \eta C. \tag{7}$$

Backprojecting into the real-world is an ill-posed solution without knowledge of how far to project. That is, we can backproject an arbitrary distance unless η is known in (7). In Steps 3 and 4, we can use the optimal feature X_f as a reference point that describes what how large the object would appear were it projected onto the image plane on which X_f lies. Values of η could easily be calculated in (8) by using the dot product of the ray described by (7) and the unit normal m_3. Given a point $\mathbf{x}_{obs}$, then its projection onto P_f is given by

$$\eta(P_f, \mathbf{x}_{obs}) = \frac{d_f - (P^+\mathbf{x}_{obs}) \cdot m_3}{C \cdot m_3} + 1. \tag{8}$$

The features in the experiments are often a few thousand meters away. In face detection, even the smallest detection box of 24×24 pixels are typically not descriptive of a human faces (unless their heads are thousands of meters long.) If we assume a typical or average size of a human face, then the ratio of his/her face size can be applied to the distance projected onto the P_f.

Let $x^{(TL)}$ and $x^{(BL)}$ be the top-left and bottom-left corners of the detected object returned by the object detector in Sec. 2.2. We can use (8) and (7) to obtain $X_f^{(TL)}$ and $X_f^{(BL)}$, the respective real-world projections. Then, if l is the assumed length of an object (e.g. someone's face), then the ratio can be applied to obtain the object's real-world 3-D coordinates

$$X^{(TL)} = (KR)^{-1} \left(\frac{l}{||X_{(TL)} - X^{(BL)}||} \eta \begin{bmatrix} x^{TL} \\ 0 \end{bmatrix} \right) - C, \tag{9}$$

and

$$X^{(BL)} = (KR)^{-1} \left(\frac{l}{||X_{(TL)} - X^{(BL)}||} \eta \begin{bmatrix} x^{BL} \\ 0 \end{bmatrix} \right) - C. \tag{10}$$

4.3 Distance Alignment

To geo-register the object, that is, to gain an absolute sense of where it lies in the real-world, a scaling metric must be defined to transfer 3-D coordinates into tangible locations and distances. This can be done in a number of ways. Recently, we have begun to relate 3-D point clouds with ladar data, an integration work that is still in progress. At present, a scaling factor has been chosen simply with approximations through Google Maps$^{\text{TM}}$ and some pedometer tools. One interesting cue specific to the photo-location aiding in distance alignment involves the nonstandard unit of length "Smoot", which has been incorporated as an optional unit of measure in Google Earth$^{\text{TM}}$. As imprecise as the entire process may seem, the approximation can be put into perspective. With issues pertaining to object size, pixel resolutions, and 3-D reprojections, errors due to distance approximation may be mitigated to an extent, but only insofar as the exactness of the parameter estimations of other factors.

5. RESULTS

There is currently a large collection of images available on the internet and various other sources. SfM reconstructions such as the "Rome in a Day" project[2] are uncooperative in the sense that they draw from images that were not taken with the sole purpose of 3-D image reconstruction. We utilize a *cooperative* data set of 1201 images (Table 1) of the Boston skyline taken from across the Charles River by Kilian Court on MIT campus. Of the 1201 images, we partition the set into 871 images to reconstruct a 3-D point cloud (S) and 330 images for testing and cross-validation (Figure 3(b)). In addition another 319 images are gathered of a similar New York City skyline from Flickr to be utilized as images that should not be part of the set (Figure 3(c)). To test across image scales, the image set consists of origimal images and their counterparts that have been decimated by two and by four, as part of the original 1201 images. The 1201 images are reconstructed and a representative set of features is generated as our initialization phase. Fig. 2 shows the reconstruction results of the 1201 images.

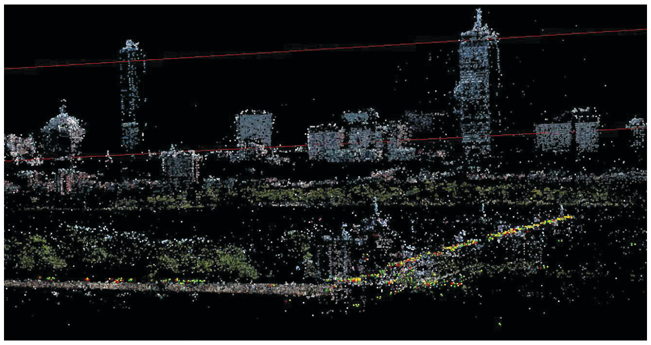


Figure 2. Point Cloud Reconstruction

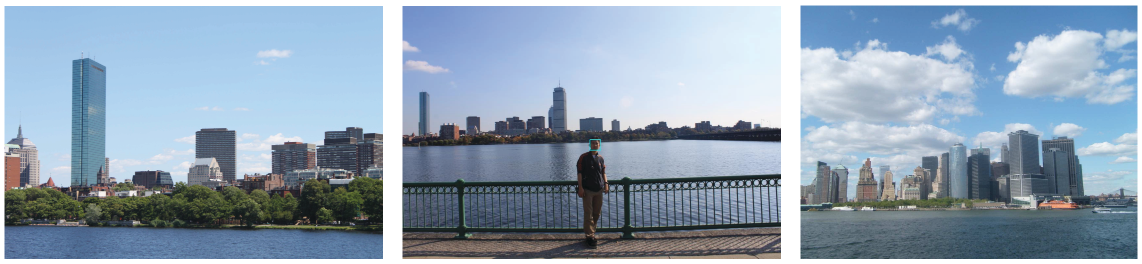

(a) Training Image, Boston Skyline (b) Testing Image, Boston Skyline (c) Testing Image, New York Skyline

Figure 3. Sample of Pictures Used

There are existing trained cascades for frontal faces and in profiles, but it is unlikely that any of them were trained with the negative set of Boston in the background. Therefore, the positive frontal face training set is obtained from a mix of the 4916 hand labeled faces created by Peter Carbonetto,[4] each of which are scaled and aligned to 24 $\times$ 24 resolution, and some post-processed versions of the UCD database.[12] The negative set was hand selected from the 1201 skyline images used in SfM.

5.1 Image Matching

From the tests we determine probability of detection and false alarms with thresholds ranging from 0 to 5% feature match. The false alarm rate is peaky around 0% and tends to drop off at a rapid rate as we increase our threshold, resulting in the sharp slope to the far left of the ROC curve in Fig. 4. Naturally, the attributes pertaining to probability curves are highly dependent on the types of images used. For example, it is easier to classify an image as not belonging to the reconstructed 3-D Boston skyline scene if it is of an entirely unrelated location (e.g., a forest or indoors), which yields $P_{FA} = 0$ thus far in our experiments. In terms of P_{miss}, further review of the images reveals that missed images have been taken at extreme angles or highly obstructed fields of view. Hence, the number of features of the background is significantly reduced.

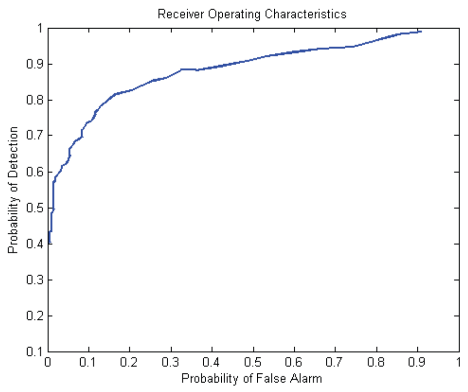


Figure 4. *Image Detection* Receiver Operating Characteristic

5.2 Image and Object Registration Error

Object detection and registration was performed with a variety of 420 test images, a subset of which, was used in object registration. Our true position for each test point recording was generated by coordinating a combination of Google Earth GPS coordinates, some markers generated by non-standard units on Harvard bridge (termed "Smoots"), physical landmarks, and a large number of physical markers.

In response to the detector's insensitivity to small scale and translation, Viola and Jones[16] filters out multiple detections per single object. A single bounding box is ideal, but the fact that it is necessary is of concern as we derive much of our registration information from the bounding box size. The consequent approximations have been imprecise as can be seen in Fig. 5 and Fig. 6 and are a topic of interest for future work.

The overall registration error can be measured in meters as the distance from the estimated position to the actual target position. As expected, the curve reflects a growing trend with respect to distance from the camera in both figures. The primary source of error, we have noticed, occurs from the actual image registration itself, and not object registration. This is reflected in the first figure, where the trend is upwards and near-linear.

Note that our detections span three dimensions, each of which introduces some error. The seemingly randomness at

Table 1. Images Used for Reconstruction

Number of Images for Reconstruction (R)	1201 at 8MP
Number of Voxels	83,796
Average Number of Features per Voxel	5.3693
Test Statistic	Percentage of Features in new image that matched to voxel features
H_1 Event	Test Statistic above a threshold
H_0 Event	Test Statistic below a threshold
Number of H_1 Test Images	330 Images (110 at 10MP, decimated by 2, decimated by 4)
Number of H_0 Test Images	319 Flickr Images

Table 2. Data Reduction Rate for 1201 Images

Data	**Number of Points**	**Percent of Original**
Raw Pixels	9,563,111,424 pixels	100%
SIFT	12,976,125 features	0.1357%
3D SIFT	83,796 features	0.000876%

further distances, especially in the feature distance, is due in most part to pixel resolution coverage. We roughly approximated one of pixels at a distance of 200 meters to be 43 meters on the horizon of the image plane, which would give us the estimation results in Fig. 6.

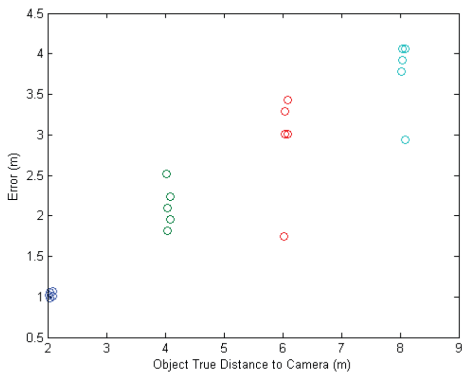


Figure 5. Error based on *Object Proximity*. Fixed feature distance at 756m.

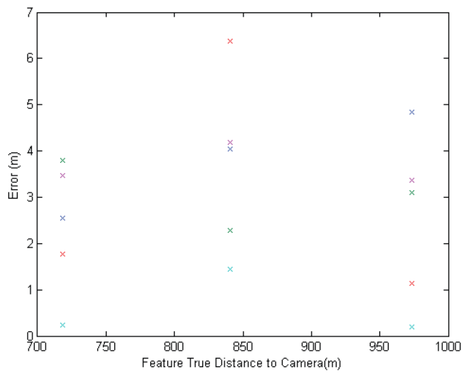


Figure 6. Error based on *Background Feature Proximity*. Fixed object distance at 8m.

6. CONCLUSIONS AND FUTURE WORK

An algorithm has been proposed for 3-D object registration given an image taken of a particular location. As an integration application with many parts, the potential for improvements is in accordance with all three phases of Fig. 1(b).

For image registration, particular image attributes for acceptable inclusion into SfM optimization remains an interesting problem. (Skeletal graph construction papers observing connectivity[15] are useful.) On a lower level, we are also currently investigating the image content that produces the most salient features.

For object detection, classifiers for object detection are often obtained by thresholding continuous functions, which is theoretically optimal as it is akin to Bayes decision rule. It would be beneficial for our applications to be as flexible as possible in terms of what targets to detect and how sensitive we are to detecting them. While adaboost is cost *insensitive*, and we can turn to asymmetric boosting techniques.[9]

In terms of the actual object registration, there is an abundance of data to be fused yet, including ladar and aerial imagery and videos that we have not begun to incorporate. Once integrated, an improvement in accuracy and precision is espected to be drastic as the available amount of information will dramatically grow. Such questions are addressable and open.

7. ACKNOWLEDGEMENTS

We would like to thank Noah Snavely for all the help and advice he has given us in regard to working with his Photo Tourism SfM project. A copy of his code that we have been working with can be downloaded from the Photo Tourism website (http://phototour.cs.washington.edu/). In addition we would also like to thank Peter Cho at MIT Lincoln Laboratories for his insights into applications with the Photo Tourism work.

REFERENCES

1. R. Alterson and M. Spetsakis. Object recognition with adaptive gabor features. *Image and Vision Computing*, 22(12):1007 – 1014, 2004. Proceedings from the 15th International Conference on Vision Interface.
2. S. Argawal, N. Snavely, I. Simon, S. Seitz, and S. Szeleski. Building rome in a day. In *International Conference on Computer Vision*, Kyoto, Japan, 2009.
3. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, 1994.
4. P. Carbonetto. Viola-jones training data. `http://www.cs.ubc.ca/\~pcarbo`.
5. G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):394–410, March 2006.
6. R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
7. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
8. S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, 1989.
9. H. Masnadi-Shirazi and N. Vasconcelos. Asymmetric boosting. In *International Conference on Machine Learning*, pages 609–619, 2007.
10. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.
11. C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. *International Conference on Computer Vision*, 1998.
12. P. Sharma. The UCD colour face image database for face detection download page. `http://ee.ucd.ie/\~prag/`.
13. N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press.
14. N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, November 2008.
15. N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal sets for efficient structure from motion. In *Proc. Computer Vision and Pattern Recognition*, 2008.
16. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. CVPR*, 1:511–518, 2001.