Visibility Probability Structure from SfM Datasets and Applications*

Siddharth Choudhary and P.J. Narayanan

Center for Visual Information Technology
International Institute of Information Technology, Hyderabad
{siddharth.choudhary@research.,pjn@}iiit.ac.in

Abstract. Large scale reconstructions of camera matrices and point clouds have been created using structure from motion from community photo collections. Such a dataset is rich in information; it represents a sampling of the geometry and appearance of the underlying space. In this paper, we encode the visibility information between and among points and cameras as *visibility probabilities*. The conditional visibility probability of a set of points on a point (or a set of cameras on a camera) can rank points (or cameras) based on their mutual dependence. We combine the conditional probability with a distance measure to prioritize points for fast guided search for the image localization problem. We define dual problem of feature triangulation as finding the 3D coordinates of a given image feature point. We use conditional visibility probability to quickly identify a subset of cameras in which a feature is visible.

1 Introduction

Structure from motion (SfM) from community collections has created sparse point clouds and images of many large monuments and other spaces. Point-directed visualization of photo collections has been the primary use of these datasets. The dataset is being used for localization [1–4] and to build dense models of the underlying space [5, 6]. A typical SfM dataset contains many points and several camera images. A few million points and a few thousand images are common today. The visibility of each point in the images is recorded as a track. The camera matrices and point coordinates are known in a common frame.

An SfM dataset encodes a rich sampling of the geometry and appearance of the space. In this paper, we explore the probabilistic structure of visibility induced by the points and images of an SfM dataset. We define the *visibility probability* of points and images and the *joint visibility probability* among points and among cameras. We then estimate the conditional visibility probability among points and cameras. We use the probability structure to efficiently guide the search for points in the localization problem. We also introduce the dual problem of feature triangulation as estimating the 3D coordinates of a given 2D feature and solve it efficiently using the visibility probability among cameras.

* This work was partly supported by the Indian Digital Heritage project of DST.

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part V, LNCS 7576, pp. 130–143, 2012.

We extend the terminology by Li et al. [3] to describe different entities. An *image* is a photograph of the space. It contains several *features*, each of which is a 2D interest point with an associated (SIFT) descriptor. The image gets upgraded to a *camera* when its projection matrix is estimated. Similarly, a feature gets upgraded to a *point* when its 3D coordinates are known. An SfM dataset contains a number of cameras, points, and their mappings. Additional images and features of the same space are easily available from several sources. Image localization aims to estimate the camera from an image while feature triangulation aims to estimate the point from a feature.

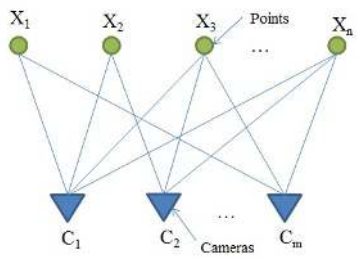
Image localization uses a few 3D points to 2D feature mappings in the query image to estimate its camera by searching the space of SfM points for match in the query image. Feature triangulation identifies a few cameras in which the query feature is visible for triangulation by searching the space of SfM cameras for visibility of the query feature. We show how these searches can be guided effectively using the conditional visibility probability among points and among cameras, the distance between them, etc. Our probability-guided exploration generates RANSAC inliers in high proportions as matching proceeds.

Our localization method successfully registers as many or more new images as the prior work, while performing about 10-20% nearest neighbor searches as them. We only need to generate less than 25% of matches to localize successfully. Our scheme can match over 4 new images per second on a normal workstation. Our feature triangulation method increases the point density at typical locations by upto 3 times and performs well in generating points in regions with repetitive appearance, where the 2D-2D match of SfM performs poorly.

2 Related Work

PhotoTourism created large SfM datasets from unstructured photographs and to interactively browse them using the points and cameras [7]. The process of SfM reconstruction has been made efficient in many ways by Agarwal et al. [8] and Frahm et al. [8, 9]. Crandall et al. used an MRF framework to model the constraints between camera and scene pairs to initialize the SfM pipeline well [10]. Image based location recognition has been attempted using databases of rectified images of buildings [11] vocabulary tree of informative features [12], distance augmented visual words [1], and probability distribution over a large collection of images [13]. SLAM literature studied image-based localization from videos [14, 15]. Alcantarilla et al. used SfM reconstructions along with camera pose priors to predict the visible 3D points in a query image [16]. Compressed scene representations using epitomes of locations [17], iconic scene graphs [18], and skeletal sets [19] have also been used for this.

Irschara et al. used the SfM point cloud for image localization, using a vocabulary tree built on a compressed set of images [2]. Li et al. used a 3D-2D matching scheme using a prioritization on SfM points for matching points in the query image [3]. They explore the model starting with a subset of highly visible points that cover the space. The priority of points changed heuristically. Sattler et al. arranged the 3D points as lists that quantized to the same visual word [4].



$$p(X_i) = \frac{d(X_i)}{m} = \frac{\text{no. of cams that see } X_i}{m}$$

$$p(C_i) = \frac{d(C_i)}{n} = \frac{\text{no. of points seen in } C_i}{n}$$

$$p(X_i, X_j) = \frac{\text{no. of cams that see } X_i \text{ and } X_j}{m}$$

$$p(C_i, C_j) = \frac{\text{no. of points seen in } C_i \text{ and } C_j}{n}$$

Fig. 1. Bipartite graph of points and cameras. Visibility probability is approximated by fractions of cameras that see single point/camera or a pair of them.

Features in the query image were matched to these lists to get a large number of highly noisy matches. They used RANSAC to overcome the highly noisy nature. Goesele et al. [5], Furukawa et al. [6] and Fuhrmann et al. [20] presented methods to get dense matches starting with an SfM dataset. Roberts et al. [21] attempted structure recovery in the presence of large duplicate structures.

3 Visibility Probability and Joint Visibility Probability

The SfM dataset consists of m cameras, n points, and the visibility of each point's track in the cameras. This can be represented as a bipartite, visibility graph [3] as shown in Figure 1. We define the *visibility probability* of a point X_i as the probability that a given point X_i is visible in a randomly selected view. We approximate it with respect to the SfM dataset as the fraction of cameras in which the point is visible as $p(X_i) == \frac{d(X_i)}{m}$, where $d(X_i)$ is the degree of X_i in the graph (Figure 1). This approaches the true visibility probability when camera and point positions are dense. The *joint visibility probability* $p(X_i, X_j)$ of two points is similarly defined as the fraction of cameras in which both are visible (Figure 1). The conditional probability of X_i on X_j then becomes the fraction of $d(X_i)$ in which X_j is visible given by

$$p(X_j|X_i) = \frac{p(X_j, X_i)}{p(X_i)} = \frac{\text{no. of cams that see } X_i \text{ and } X_j}{\text{no. of cams that see } X_i}. \tag{1}$$

Equation 1 gives the probability of X_j being visible given that X_i is or the dependence of X_j on X_i. The dependence will be high on nearby points but could be lower if occluded in some views. We use the terms dependence and independence not in the sense of statistical independence, but to mean how one is not influenced by the other. We can measure the influence of a set S of points on the visibility of a X_j as

$$p(X_j|S) = 1 - \prod_{X_i \in S} \left[1 - p(X_j|X_i)\right], \tag{2}$$

as a point is independent of a set of other points only if it is independent of each one in the set.

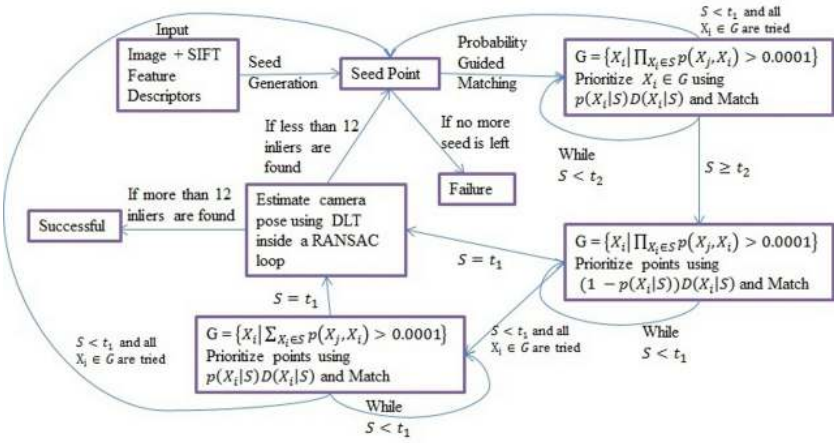


Fig. 2. Diagram showing the steps of visibility probability guided image localization

Similarly, we can define the visibility probability $p(C_i)$ and the joint visibility probability $p(C_i, C_j)$ of a cameras C_i, C_j with respect to a SfM dataset as fraction of points visible in C_i and in C_i and C_j respectively (Figure 1). The conditional visibility probability of cameras C_i on C_j and the influence of a set S of cameras on a camera C_j become

$$p(C_j|C_i) = \frac{p(C_j, C_i)}{p(C_i)} \quad \text{and} \quad p(C_j|S) = 1 - \prod_{C_i \in S} [1 - p(C_j|C_i)]\,. \tag{3}$$

The probability structure between and among points and cameras can be used to predict which points are likely to be visible in a camera given a few other visible points and which camera see a point given a few other cameras that see it. This helps to efficiently search for points visible in images or cameras that see features.

4 Image Localization

Image localization aims to compute the camera parameters of a given query image by matching a number of SfM points in them. Our image localization algorithm is guided by the conditional visibility probabilities among points (Eq. 2). It has three parts: seed generation, probability-guided point matching, and camera estimation. Algorithm 1 and Figure 2 outline our localization procedure.

4.1 Seed Generation

Localization starts with identifying a *seed* point from the SfM dataset in the query image I. We adapt the 2D-3D searching method used by Sattler et al. [4] for this. We first build the lists of points whose descriptors quantize to the same visual

Algorithm 1. Localize (Image **I**)

1: Compute SIFT interest points and descriptors of the query image I
2: Find a seed point and set it as set S.
   If number of seeds generated exceeds `maxSeed`, declare failure and exit
3: **while** S contains fewer than t_1 points **do**
4: Update the *candidate set* G of points using joint visibility to points in S
5: If $|S| < t_2$, set $p(X_i|S)D(X_i, S)$ as the priority of points $X_i \in G$
6: Otherwise, set $(1 - p(X_i|S))\, D(X_i, S)$ as the priority.
7: Search each $X_i \in G$ in priority order by finding 2 closest descriptors in I
8: If distances to the descriptors satisfy the ratio test, add X_i to S
9: If no more points in G, proceed to Step 11
10: **end while**
11: If $|S| < t_2$ or $|S| < t_1$ in expansion mode, discard S, G and start over from Step 2
12: If $|S| < t_1$, repeat from Step 3 in the expansion mode.
   Start over from Step 2 if t_1 points can't be found.
13: Estimate camera pose using RANSAC over DLT
14: If number of inliers is less than 12, start over from Step 2
15: Declare success and return camera matrix

word. SIFT features extracted from the query image are also quantized to visual words. The lists for the visual words in the query image are arranged in decreasing order of their lengths, in contrast to Sattler et al. Each query-image feature is searched linearly in the lists in order. A feature matches a point if it clears the ratio test in the list and is considered a potential seed. Reverse matching of the potential seed using the ratio test in image I confirms it as a seed.

Sattler et al. start the search from the shortest list to exploit uniqueness [4]. We search from the longest list to prefer more popular visual words over less popular ones. The average number of reverse matching steps needed before finding a seed was 14 when searched in the descreasing order compared to 230 when search was in the increasing order. Sattler et al. did not use reverse matching for confirmation and relied on RANSAC to remove the outliers. Our seed generation can thus be described as a 2D-3D-2D matching approach.

4.2 Probability-Guided Point Matching

Guided matching starts with a set S of points (initially with the seed point) that are already matched in the query image I. The process repeatedly identifies a candidate set G of points for matching and prioritizes them based on parameters like their independence of S, distance, etc. The points are searched in the query image in the priority order. The matched ones are added to S. The process restarts with a new seed otherwise. It fails after trying a certain number of seeds. Figure 3 shows the results for a few steps.

In the normal mode, the candidate set G is the subset of SfM points that are jointly visible to all points in S (Step 4). We include a point X_j into G if the product $\prod_{X_i \in S} p(X_j, X_i)$ exceeds a threshold. A very small threshold (0.0001) is used to get more points. Other data-dependent thresholds can be used if

suitable. The set G is large when S has a single point and shrinks fast as points are added. An expansion mode is used for G by including points with the sum $\sum_{X_i \in S} p(X_j, X_i)$ exceeding a threshold when G gets empty with a few promising matches in S (Step 12). G then contains points that are jointly visible with any point in S in this mode.

The priority assigned to a point in G should reflect its potential contribution to localization. Points independent of the already matched ones can contribute more. Since $p(X_i|S)$ gives the dependence of X_i on points in S, $1-p(X_i|S)$ as the priority will consider independent points early (Step 7). Distant matching points contribute more to the linear estimation of camera matrix than proximate ones. Independent and distant points can be preferred with $(1 - p(X_i|S))D(X_i, S)$ as the priority to points $X_i \in G$ in Step 5, where $D()$ a distance measure. We use a triangle function of the Euclidean distance of X_i to the closest point in S as $D(X_i, S)$. The function increases linearly till distance equals 0.25 times the model extent and falls linearly beyond it, emphasizing distant points while de-emphasizing very far off ones. The use of the distance measure achieves the same localization error using half the number of point matches. As each matched point is added to S, the candidate set changes and the priorities are updated. A safe mode is used in the initial phase to include points close to point in S by using $p(X_i|S)D(X_i, S)$ as the priority. This continues until t_2 (currently 5) points are matched (Step 6). Table 1 shows the performance using different t_2 thresholds on the three datasets.

Points are considered in the descending order of priority for matching (Step 7). A point matches if its two nearest neighbors of I in the descriptor space satisfies the ratio test [22]. We proceed to a RANSAC-based camera estimation after finding t_1 matches (currently 20). Table 2 shows the impact of increasing t_1. The high RANSAC inlier percentage indicates that our scheme searches promising points early. If G gets empty before matching t_2 points, the seed selection is faulty and we restart the process (Step 11). An expansion mode is used if fewer than t_1 matches are obtained, which happens less than 10% of the time. A new seed is tried if these efforts cannot produce sufficient matches (Step 12).

The threshold t_1 controls the matching effort and the accuracy of localization. A value of 20 to 30 provides a good balance between accuracy and effort (Table 2). The matching methods by Li et al. [3] and Sattler et al. [4] need to

Table 1. Localization performance: The inlier ratio is high, number of seeds is low, and the number of nearest neighbor queries is low. $t_2 = 5$ enhances performance greatly.

Dataset	Threshold t_2	# Imgs Localized	Avg Inlier Ratio	With 1 Seed	Avg # Seeds	Avg # NN queries	Avg Loc. accuracy	Avg. Regn time (sec)
Dubrovnik	0	786	0.73	621	1.76	1323	42.32	0.25
	5	788	0.78	692	1.3	840	34.79	0.25
Rome	0	976	0.78	539	2.94	3647	-	0.31
	5	977	0.81	755	2.74	3253	-	0.27
Vienna	0	218	0.75	144	2.22	1918	-	0.42
	5	219	0.74	144	2.25	972	-	0.40

Fig. 3. Steps shown over the point cloud. Query image is the inset. From left: Seed (green) and ground truth camera, candidate set G with priority (increasing from blue to red), matched point and new G with priority, prioritized G after 3 matches, and after 5 matches. See the electronic version.

Table 2. Performance for different t_1 values on the Dubrovnik dataset. Matching 20-30 points provides the best performance-effort tradeoff.

No. points in S	15	20	30	50	100
Inlier percentage	79.9	78.12	76.15	75.32	66.76
No. images registered	727	788	789	789	789
No. points searched	523	839	1679	2893	4597
Error in localization using RANSAC					
Median	0.0037	0.0023	0.0023	0.0022	0.0021
70th percentile	0.011	0.0073	0.0058	0.0063	0.006
Mean	0.1177	0.0267	0.0498	0.0261	0.0512
Error in localization without RANSAC					
70th percentile	0.0231	0.0243	0.0263	0.0345	0.0425

produce 100 or more potential matches before estimating the camera. If RANSAC produces fewer than 12 inliers, the seed is discarded (Step 14). If the number of seeds tried exceeds a limit, the localization of the query image fails. Around 88% of the query images in the Dubrovnik dataset are successfully localized using the first seed. The average number of seeds tried was 1.3 for successful query images and 5 for unsuccessful ones (Table 3).

Discussion: Li et al. use a prioritized selection of points for matching [3]. They update the priorities of jointly visible points when a new point is matched by a factor inversely proportional to its degree. Our probability structure allows better pruning, prioritizing and reordering of the remaining points than the above heuristic. We search fewer points (Table 8) and get better localization accuracy (Table 4) as a result. Sattler et al. [4] extend the 2D-3D matching until a large number of matches are found, with static priorities for points. While they perform well on images that localized well, they do much more work on rejection. Our rejection and registration times are comparable (Table 5).

4.3 Camera Estimation

We estimate the camera parameters using RANSAC over DLT from the 3D-2D matches obtained. Our approach generates reliable matches, with over 78%

Table 3. Localization performance for registered and rejected images for Dubrovnik, Rome and Vienna datasets (from top to bottom). The computation times are similar.

Successfully registered images						Rejected images		
# of Images	Inlier Ratio (Avg)	With 1 Seed	Avg # Seeds	Avg # NN queries	Time (sec)	Avg # Seeds	Avg # NN queries	Time (sec)
788	0.78	692	1.3	839.6	0.25	5.01	4199	0.51
977	0.81	755	2.74	3253	0.27	6.21	5876	0.61
219	0.74	144	2.25	972	0.40	4.23	3369	0.49

RANSAC inliers. About 20 matches are sufficient in most cases, but an accuracy-speed tradeoff is provided by adjusting t_1 as seen in Table 2. We use the basic p6p method for camera estimation. Techniques like p4pfr and p3p do not improve the accuracy by much, but require higher computation time.

4.4 Pre-computation and Representation

We pre-compute the visual-word based sorted lists for the seed generation step [4]. SfM datasets contain large numbers of points, all of which may not be informative. We use the method by Li et al. to select 3-5% of the points in the dataset using the K-Cover algorithm that includes the most valuable points [3]. In practice, using all points showed no major gain in accuracy but suffered in speed. The $n \times n$ sparse visibility matrix between all pairs of points with (i, j) entry storing the number of cameras that see points i and j is precomputed and stored in a compressed row format.

4.5 Experimental Results

We use the Dubrovnik (6044 images, 2M points), Rome (15179 images, 4M points), and Vienna (1324 images, 1M points) datasets for experiments [2–4]. We first select 75K, 100K and 50K points respectively using the K-cover algorithm We use the same query images (800 for Dubrovnik, 1000 for Rome and 266 for Vienna) to facilitate direct comparison. We match more images than previous efforts and do so with lesser effort in terms of the number of nearest neighbor queries (Table 3). We also generate promising matches early as seen from the RANSAC inlier ratio. We match 4 images per second on the average.

Table 4 compares the localization accuracy of our approach with prior work on the Dubrovnik dataset. We achieve better localization compared to Li et al. [3]. The mean error seems worse because we match more images than them, with most of them being of low quality for matching. Sattler et al. [4] perform better on median error, but has very high mean error due to several images localized very badly. This is true even though they register fewer images.

Images from Rome dataset were searched in Dubrovnik and Vienna datasets and images from Dubrovnik in Rome to study the rejection performance. Table 5 shows the results. The candidate set gets exhausted fast and the search terminates quickly in our approach. Rejection involves more nearest-neighbor queries

Table 4. Comparision of localization accuracy in meters for Dubrovnik dataset

Method	#Reg. Images	Average	Median	1st Quartile	3rd Quartile
Our Method	788	34.79	3.104	0.88	11.83
Sattler et al. [4]	784	53.9	1.4	0.4	5.9
Li et al. [3]	753	18.3	9.3	7.5	13.4

Table 5. Rejection performance on negative examples from other datasets

Dataset	# Images	# Rejected	Avg #Seeds	Avg #NN Queries	Time (sec)
Dubrovnik	1000	1000	2.8	3451	0.4
Rome	800	800	2.70	5026	0.54
Vienna	1000	1000	3.44	2810	0.22

than registration. Our rejection times are only slightly more than registration times. This is in contrast to rejection being typically an order of magnitude slower for the previous methods [3, 4] (Table 6). Rejecting images from other datasets is slightly faster than rejecting images from the same dataset.

The performance of localizing new images of the same space is given in Table 7. We collected general images from Flickr using relevant keywords for Dubrovnik, Rome, and Vienna. These images were taken in 2011 or later. They registered quickly and produced low reprojection errors. Six registered and four rejected images of Dubrovnik are shown in Figure 4.

Our localizing performance is compared with methods by Li et al. and Sattler et al. in Table 8. Our method successfully registers more images in each data set while searching far fewer points. We search only 10-20% of the points as Li et al. The visibility probability is thus successful in guiding the search to the successful points and in rejecting matches.

5 Feature Triangulation

We define this problem as follows: Given a feature-point with 2D coordinates and a descriptor, find its 3D coordinates. By assuming that the feature is taken from the space of the SfM dataset, we can match it in two or more cameras. Triangulation can be used to give the point coordinates after matching in cameras (whose calibration matrices are known). This is a dual problem of image localization, with the roles of cameras and points reversed. However, there usually are at least two orders of magnitude more features than images in a dataset. Hence,

Table 6. Rejection performance comparison on 3 datasets for images from same dataset and negative examples from other datasets

Method	**Dubrovnik**		**Rome**		**Vienna**	
	Negative examples	Same dataset	Negative examples	Same dataset	Negative examples	Same dataset
Ours	0.4	0.51	0.54	0.61	0.22	0.49
Sattler at al. [4]	-	1.70	-	1.66	-	2.43
Li et al. [3]	3.96	-	4.67	-	3.62	-

Fig. 4. Six registered (left) and four rejected (right) new images of Dubrovnik

Table 7. Registration performance on new images obtained from internet

Dataset	# Query Images	#Registered	Avg # Seeds	Avg #NN Queries	Avg Inlier Ratio	Avg Reproj. Error	Avg Time (sec)
Dubrovnik	70	64	2.81	950	0.80	1.25	0.21
Rome	44	40	3.1	3834	0.78	1.7	0.3
Vienna	40	27	2.59	2381	0.81	0.5	0.36

it will be more useful to triangulate several features from an image. Our feature triangulation algorithm is the dual of our localization algorithm. It has has three steps: feature selection, view selection and two-way matching. Algorithm 2 outlines our triangulation approach.

Feature-Point Selection: This step identifies the 2D feature-points in an image to be triangulated. We use a camera-image C_0 from the SfM dataset as the image from which features are selected. We detect interest points and descriptors in the selected image first. We restrict the features to lie sufficiently far from the image borders. Alternately, the user can draw a rectangle to restrict the features. We discard interest points that are within 4 pixels of an existing point in the image. This ensures we triangulate new features. The guided camera selection starts with an image and a feature in it.

Probability-Guided Camera Selection: This step starts with a set S of selected cameras (initially only C_0) which contains the query feature F. Cameras that are jointly visible to every camera in S are selected as the candidate set G (Step 3). We include cameras C_j in G if $\prod_{C_i \in S} p(C_j, C_i) \geq 0.0001$. The candidate set is very large initially as typical cameras contain many points. Cameras of that are most independent of S are most useful for triangulation and hence $1 - p(C_i|S)$ is a good priority measure for C_i. Cameras at greater distances in 3D and angular spaces from the cameras in S are more useful in triangulation. We also use a *local*

Table 8. Localization performance comparison on three datasets. The query set had 800, 1000, and 266 images for Dubrovnik, Rome, and Vienna datasets.

Method	Dubrovnik			Rome			Vienna		
	# registered	# points searched	Time (sec)	# registered	# points searched	Time (sec)	# registimage	# points searched	Time (sec)
Ours	789	839.6	0.25	977	3253	0.27	219	972	0.40
Sattler [4]	784	-	0.28	975	-	0.25	207	-	0.46
Li [3]	753	9756	0.73	921	12963	0.91	204	6245	0.55

Algorithm 2. Feature Triangulation (Camera C_0, Feature F)

1: Initialize S as the camera C_0
2: **while** S contains fewer than q_1 cameras **do**
3: Update the *candidate set* G of cameras using visibility to cameras in S
4: Assign $lv(C_i)(1 - p(C_i|S))\alpha(C_i, S)$ as the priority to cameras $C_i \in G$
5: Examine camera $C_i \in G$ in priority order. Match F in it.
6: If C_i satisfies two-way matching, add it to S
7: If no more cameras in G, proceed to Step 9
8: **end while**
9: **if** $|S| \geq 2$ **then**
10: Triangulate camera projections to estimate a 3d point
11: **else**
12: Declare Failure
13: **end if**

view prior $lv(C_i)$ based on the feature F. We first identify K points (typically 10) that are closest to F in C_0 using image distances. The $lv(C_i)$ for C_i is the fraction of these points visible in C_i. This promotes cameras that see more points that are close to F. We can also use a distance measure $D(C_i, S)$ and an angle measure $\alpha(C_i, S)$ in the priority, both of which are triangle functions based on the distance from cameras in S. In practice, the angle measure does the better than the distance measure. Thus, the priority of C_i is $lv(C_i)(1 - p(X_i|S))\alpha(C_i, S)$ (Step 4). The use of local view prior typically results in the successful triangulation of twice the number of features.

The cameras are selected in their priority order to match the query feature F in them. We use a two-way matching process (Step 5). To match F in C_i, we find the 10 nearest neighbors of F in C_i in the descriptor space. The feature-point p closest to the epipolar line of F in C_i is considered a potential match if its epipolar distance in C_i is less than 4 pixels. We then find the nearest neighbor of p in camera C_0. If its nearest neighbor is the feature F, we consider F to be matched in C_i. This method finds matches of even repetitive feature-points, which are rejected by the traditional ratio test. Finding 3D points among difficult repetitive structures is the primary advantage of our feature triangulation approach. As each camera is added to S, the candidate set shrinks (Step 3) and

Fig. 5. Steps shown over the point cloud with chosen camera and query feature as inset. From left: First camera, candidate set G (blue), next matched camera and new G, prioritized G after 3 matches, and after 4 matches. See the electronic version.

the priorities of cameras change. The guided camera selection continues while there are cameras left in G or when q_1 cameras are in S. A value of q_1 in the range 7 to 10 performs well in practice. Triangulation fails if F matches in less than 2 cameras. We try to match only 20 cameras for each query feature. Figure 5 shows results of a few steps of the process.

Point Estimation: The 3D coordinates of the query feature F are computed by triangulating its projections in the cameras in S. We use RANSAC over a simple 2-view triangulation for this. Robust triangulation is a well studied problem [23]. However, our problem involves highly noisy cameras for which these methods aren't suitable. We get adequate accuracy using the simple method, though better techniques should be explored for this step in the future.

5.1 Experimental Results

We used the already triangulated points from 55 random Dubrovnik images to evaluate the performance of triangulation, with their original coordinates treated as the ground truth. The 2D feature point of each point from an image in its track was used as the query for triangulation. Tables 9 and 10 give the triangulation results, grouped by their track lengths and reprojection errors. Triangulation error is the Euclidean distance between original point and the point generated by our method. Table 9 shows that points with longer tracks are triangulated better. Better triangulated points involve larger variances in viewing angles and show larger reprojection errors (Table 10). That is, low reprojection error is not indicative of better triangulation. The limit on using 20 images based on the local-view prior resulted in very fast triangulation with low error.

Table 11 compares two-way matching and ratio-test matching on 9K randomly selected points from Dubrovnik. Both methods worked well on the SfM points because they passed the ratio test in the original SfM process. The performance of adding new points to images with repetitive structures (Figure 6) is better for two-way matching. It adds 8 to 20 times more points on images with repetitive structures than ratio test. The point density at these locations increases by upto 3 times. Image 1 (Fig. 6, left) had 1946 points to which 1692 new points were added. Image 2 (Fig. 6, center) had 1747 triangulated points to which 3359 new points are added. Image 3 (Fig. 6, right) had 187 triangulated points to which 613 new points were added.

Table 9. Triangulation results by track lengths on 55 random Dubrovnik images

Track Length	No. pts triangulated	No. images tried	Triangulation error	Reprojection error	Running time [s]
2	6000	20	0.0276	0.2108	0.10
3	5887	20	0.0201	1.5218	0.11
4	4631	20	0.0125	1.9844	0.11
5	3071	20	0.0060	2.1592	0.12
> 5	6547	18.84	0.0030	2.2799	0.12

Table 10. Triangulation results by reprojection errors on 55 random Dubrovnik images

Reprojection Error	No. pts triangulated	No. images Tried	Triangulation error	Reprojection error	Running time [s]
< 1 pixels	9697	19.89	0.0205	0.3968	0.10
< 2 pixels	17005	19.80	0.0144	0.8664	0.11
< 3 pixels	22440	19.74	0.0153	1.2541	0.11

Table 11. Effect of different matching techniques on repetitive structures

Dataset	Two Way Matching					Using Ratio Test				
	#Pts added	#Images Tried	Tri. error	Reproj. error	Time (sec)	#Pts added	#Images Tried	Tri. error	Reproj. error	Time (sec)
9126 Pts	8830	267.36	0.0105	1.40	0.728	8950	270.29	0.0097	1.7	0.69
Image 1	1692	19.99	-	0.81	0.13	215	20	-	0.94	0.08
Image 2	3359	19.99	-	0.85	0.50	172	19.97	-	1.02	0.20
Image 3	613	20	-	1.13	0.36	49	20	-	1.35	0.19

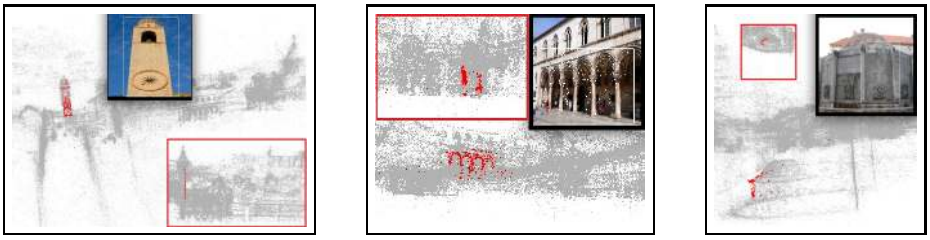


Fig. 6. Triangulated points shown in red from different view points. Side view is shown with red borders. The starting image is the inset.

6 Conclusions

In this paper, we defined the visibility probability structure of an SfM dataset and applied it to the dual problems of image localization and feature triangulation. Our method produced fast and efficient point and camera matches for both registration and rejection, increasing the point density by upto 3 times in places with repetitive structures. In the future, reliable image features can be reconstructed using th SfM process with the rest of the features triangulated using our approach. The high storage requirements for the joint visibility information is a drawback of this work. The formulation depends on the probability structure to guide the matching process. This can have difficulty in sparse regions of space, where the empirical probabilities are unreliable.

Acknowledgements. We thank Noah Snavely for the datasets and Torsten Sattler for insightful discussions over email.

References

1. Fraundorfer, F., Wu, C., Frahm, J.M., Pollefeys, M.: Visual word based location recognition in 3d models using distance augmented weighting. In: 3DPVT (2008)

2. Irschara, A., Zach, C., Frahm, J.M., Bischof, H.: From structure-from-motion point clouds to fast location recognition. In: CVPR (2009)
3. Li, Y., Snavely, N., Huttenlocher, D.P.: Location Recognition Using Prioritized Feature Matching. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 791–804. Springer, Heidelberg (2010)
4. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2d-to-3d matching. In: ICCV (2011)
5. Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S.: Multi-view stereo for community photo collections. In: ICCV (2007)
6. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Towards internet-scale multi-view stereo. In: CVPR (2010)
7. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. ACM Trans. Graph. 25, 835–846 (2006)
8. Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building rome in a day. In: ICCV (2009)
9. Frahm, J.M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.H., Dunn, E., Clipp, B., Lazebnik, S., Pollefeys, M.: Building Rome on a Cloudless Day. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 368–381. Springer, Heidelberg (2010)
10. Crandall, D., Owens, A., Snavely, N., Huttenlocher, D.P.: Discrete-continuous optimization for large-scale structure from motion. In: CVPR (2011)
11. Robertson, D., Cipolla, R.: An image-based system for urban navigation. In: BMVC (2004)
12. Schindler, G., Brown, M., Szeliski, R.: City-scale location recognition. In: CVPR (2007)
13. Hays, J., Efros, A.A.: im2gps: estimating geographic information from a single image. In: CVPR (2008)
14. Fraundorfer, F., Engels, C., Nistér, D.: Topological mapping, localization and navigation using image collections. In: IROS (2007)
15. Chli, M., Davison, A.J.: Active Matching. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 72–85. Springer, Heidelberg (2008)
16. Alcantarilla, P.F., Ni, K., Bergasa, L.M., Dellaert, F.: Visibility learning for large-scale urban environment. In: ICRA (2011)
17. Ni, K., Kannan, A., Criminisi, A., Winn, J.: Epitomic location recognition. In: CVPR (2008)
18. Li, X., Wu, C., Zach, C., Lazebnik, S., Frahm, J.M.: Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 427–440. Springer, Heidelberg (2008)
19. Snavely, N., Seitz, S.M., Szeliski, R.: Skeletal sets for efficient structure from motion. In: CVPR (2008)
20. Fuhrmann, S., Goesele, M.: Fusion of depth maps with multiple scales. ACM Trans. Graph. 30, 148:1–148:8 (2011)
21. Roberts, R., Sinha, S., Szeliski, R., Steedly, D.: Structure from motion for scenes with large duplicate structures. In: CVPR (2011)
22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision (2004)
23. Seo, Y., Hartley, R.I.: Sequential L_∞ Norm Minimization for Triangulation. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part II. LNCS, vol. 4844, pp. 322–331. Springer, Heidelberg (2007)