Visibility Learning in Large-Scale Urban Environment

Pablo F. Alcantarilla, Kai Ni, Luis M. Bergasa, Frank Dellaert

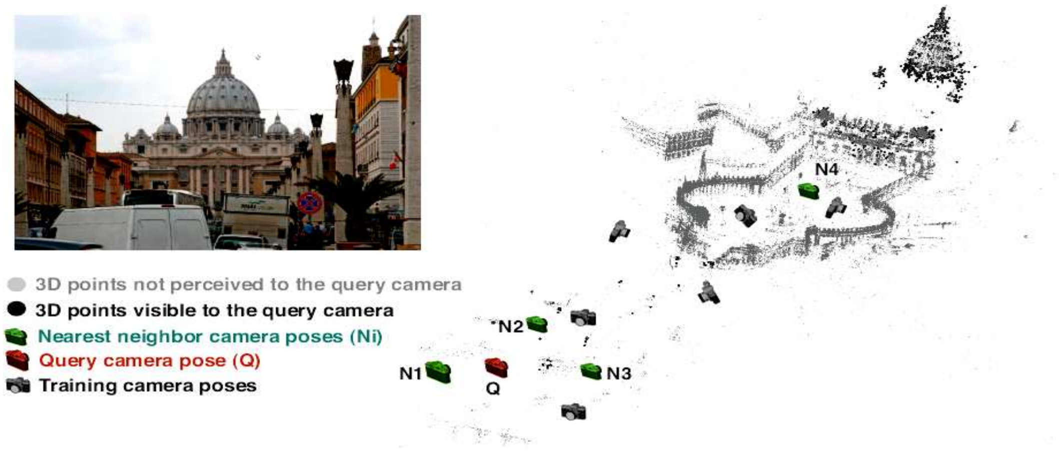

3D points not perceived to the query camera
3D points visible to the query camera
Nearest neighbor camera poses (Ni)
Query camera pose (Q)
Training camera poses
N1
N2
N3
N4
Q

Fig. 1. Given a large city-scale 3D reconstruction, we predict the visible 3D points for a query camera view by fusing both geometric and appearance information from multiple neighbor cameras. Best viewed in color.

***Abstract*— A crucial step in many vision based applications, such as localization and structure from motion, is the data association between a large map of known 3D points and 2D features perceived by a new camera. In this paper, we propose a novel approach to predict the visibility of known 3D points with respect to a query camera in large-scale environments. In our approach, we model the visibility of each 3D point with respect to a camera pose using a memory-based learning algorithm, in which a distance metric between cameras is learned in an entirely non-parametric way. We show that by fully exploiting the geometric relationships between the 3D map and the camera poses, as well as the related appearance information, the resulting prediction is much more robust and efficient than conventional approaches. We demonstrate the performance of our algorithm on a large urban 3D model in terms of both speed and accuracy.**

I. Introduction

Large-scale 3D applications, such as robot localization and structure from motion (SfM), have recently become a more and more popular topic in robotics and computer vision. With the introduction of handheld devices equipped with cameras, accelerometers and GPS sensors (e.g., mobile phones), extending these 3D applications to such devices is very much desired. However, most of the existing approaches are designed for offline processing due to their high computational cost.

One of the most computationally expensive steps in vision-based localization is data association, in which matching candidates between a large map of 3D points and 2D features are retrieved and then usually validated by geometric constraints using RANSAC [1]. For the environments with highly repetitive textures, such as cities, the traditional methods mainly depend on the appearance information, which results in a very large number of matching candidates due to the ambiguities introduced by visually similar features [2].

Visibility prediction is a commonly used technique [3], [4] to greatly reduce the ambiguities and speed up the data association by making an accurate and robust prediction of the most likely visible features for a given camera pose. More specifically, in the problem of visibility prediction, we want to determine whether a certain 3D point in the known 3D environment can be perceived by a given query camera. In this paper, we propose a novel way to predict the visibility of 3D points efficiently and robustly.

There are two main types of information that are used for aiding data association. First, we have geospatial information, which is widely used in tracking based localization approaches, such as the work by Klein and Murray [5] and Davison et al. [6]. These methods can be quite efficient in a limited-size indoor space but do not generalize well to large outdoor scenes. Moreover, they need small baselines between the consecutive images hence are not proper for the wide-baseline images taken

Pablo F. Alcantarilla and Luis M. Bergasa are with Department of Electronics, University of Alcalá. Alcalá de Henares, Madrid, Spain. e-mail: pablo.alcantarilla, bergasa@depeca.uah.es

Kai Ni and Frank Dellaert are with School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA. e-mail: nikai,dellaert@cc.gatech.edu

by most of the handheld devices.

Second, the appearance of the 3D structure is another important source of information, which is typically exploited by using feature descriptors such as SIFT [7]. However, for very large databases, the computation time to match the features in the current image to the 3D points in the database can be prohibitively expensive, and even the most advanced algorithms [8], [9] do not run in real-time. One important reason is that those methods tend to ignore the previous visibility information, i.e. the 3D points and their corresponding cameras poses. This information can directly lead to a very fast prediction based on the weak priors of the current pose and the appearance of the corresponding images without involving expensive descriptors.

In this paper, we introduce a memory-based learning framework to predict, for each 3D point, its visibility with respect to a query camera pose. Our approach memorizes camera poses for each feature in the dataset and uses a non-parametric model to efficiently capture the visibility of landmarks in cluttered and city-like environments. Figure 1 shows an example featuring a large city-scale 3D reconstruction, comprising of recovered 3D points and camera poses. A new camera with noisy pose prior is queried, and our algorithm predicts the visibility of the 3D points for that camera by fusing the information from the nearby cameras, exploiting all the geometric information available from the 3D environment. In this way, we can considerably improve the data association between the large map of 3D points and the features in the current image, yielding higher quality matches than conventional approaches.

In the remainder of the paper, we describe the related work in Section II and introduce our probabilistic modeling of visibility in Section III. Then, we describe our metric learning framework in Section IV and how to obtain a very fast visibility prediction in Section V. In Section VI, we show the experimental results of our algorithm in a large-scale 3D reconstruction of St. Peter's Basilica in Rome. Finally, main conclusions and future work are described in Section VII.

II. Related Work

Zhu et al. [10] showed how to build an optimal 3D landmark database and how to use this database for real-time global localization. Through an intelligent subsampling of the landmark database based on geometry constraints, the size of the database was reduced without sacrificing the accuracy in localization. In this approach landmarks are characterized by their appearance using Histogram of Oriented Gradients (HOG) [11]. The selection of putative matches for pose estimation relies vastly on appearance descriptors without exploiting the available geometry information from the reconstruction. Data association between current features in the image and 3D points in the database is done by means of a vocabulary tree, which is built by hierarchical *K-means-clustering* [8]. In order to speed-up their hierarchical database search strategy, they perfor two different pruning stages of the large database by means of geo-spatial constraints (3D camera pose location and its uncertainty) and via a vocabulary tree.

In [4] Wuest et al. showed an augmented reality application in which they modeled for each feature the probability of the locations from where every feature can be tracked successfully. This probability is modeled by means of a finite set of Gaussian mixtures. The extension of this method for larger environments is difficult and computationally expensive. Moreover, since they only take into account the camera translation for their visibility prediction, their method only works with small scenarios where the degrees of possible camera orientations are very limited. As we will show later in our experimental results (see Section VI), the viewing direction or camera orientation has a much stronger impact than camera translation when predicting whether two cameras share common features or not. This is also one of the most important drawbacks of the mentioned work by Zhu et al. [10], since their geo-spatial pruning only takes into account camera translation and its corresponding uncertainty, ignoring viewing directions.

Alcantarilla et al. [3] proposed a non-parametric framework for learning the visibility of reconstructed 3D points and used this visibility prediction for robust and fast vision-based localization under small indoor scenarios and baselines. Their algorithm learns a kernel function that measures the similarity between two camera poses, combining euclidean distance and normalized dot product between camera translations and viewing directions respectively. They learn the kernel parameters by fitting a sigmoid function from the training data using nonlinear regression techniques [12], ignoring the correlations between the different cues that are used in the metric. This makes the convergence of the nonlinear regression highly dependent on the initial values of the unknown parameters, and good guesses of the final value of the unknown kernel parameters are necessary for convergence and feature scaling problems [13].

In this paper, we propose a new metric learning algorithm combining Gaussian kernel and Mahalanobis distance and show results over large-scale urban environment 3D reconstructions. Our algorithm does not suffer from initialization problems and learns its own scaling, being more suitable for the addition of new cues to the proposed metric. We have further investigated the addition of new cues such as local histograms, which have been succesfully applied to location recognition problems [14], [15] recently.

III. Probabilistic Visibility Model

In the visibility prediction problem, we are interested in the posterior distribution of the visibility v_j for a certain 3D point x_j given the query camera pose θ, denoted as $P(v_j|\theta)$. For this purpose, we propose to use a form of lazy and memory-based learning known as *Locally Weighted Learning* [13]. This technique is a simple memory-based classification algorithm and can be implemented very efficiently. The idea is very simple: given the training data that consists of a set of reconstructed camera poses $\Theta = \{\theta_1 \ldots \theta_N\}$, the 3D point cloud $X = \{x_1 \ldots x_M\}$ and a query camera pose θ, we form a locally weighted average at the query point and take that as an estimate for $P(v_j|\theta)$ as follows:

$$P(v_j|\theta) \approx \frac{\sum_{i=1}^{N} k(\theta, \theta_i) \cdot v_j(\theta_i)}{\sum_{i=1}^{N} k(\theta, \theta_i)} \tag{1}$$

Now, we will explain the meaning of the functions $k(\cdot)$ and $v_j(\cdot)$ that are involved in the locally weighted average in Equation 1:

- The function $k(\theta, \theta_i)$ is a weighting function or kernel function that is used to calculate a weight, which emphasizes those camera poses that are similar to the query camera pose θ and deemphasizes very different camera poses. This makes sense, since if the camera at pose θ_i has already seen the point x_j, we may expect that the closer the query camera is to θ_i, the more likely it is to see point x_j.
- The function $v_j(\theta_i)$ just assigns a real value equal to 1 for those cases where a certain 3D point x_j is visible by a camera pose θ_i and 0 otherwise. Specifically, this function is a boolean random variable defined as follows:

$$v_j(\theta_i) = \begin{cases} 1 & \text{if } x_j \text{ is visible from camera } \theta_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

According to Equation 2, the final approximation of the locally weighted averaging for the visibility posterior can be derived as:

$$P(v_j = 1|\theta) \approx \frac{\sum_{i=1}^{N} k(\theta, \theta_i^{v_j=1})}{\sum_{i=1}^{N} k(\theta, \theta_i)} \tag{3}$$

where $\theta_i^{v_j=1}$ are the camera poses from the training data in which the 3D point x_j is visible.

IV. Learning Kernel Functions

In this section, we show how to learn a proper distance metric or kernel function between two camera poses. A good distance metric (i.e. the kernel function k) is crucial for the overall performance of the proposed approach. Similar to the framework proposed by Weinberger and Tesauro [16], our approach combines the Mahalanobis distance and the Gaussian kernel, and it applies to any distance-based kernel function with differentiable dependencies on parameters specifying the distance function. The Gaussian kernel is generally defined as follows:

$$G_{ij} = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left(-\frac{d(\vec{\theta}_i, \vec{\theta}_j)}{\sigma^2}\right) \tag{4}$$

where $d(\vec{\theta}_i, \vec{\theta}_j)$ is the squared distance between the vectors $\vec{\theta}_i$ and $\vec{\theta}_j$. These vectors encode different information related to the camera pose and its associated image view. We will explain in Section IV-A and IV-B how to define these input vectors. For simplification, we can drop the constant factor before the exponent in Equation 4 and absorb σ^2 in $d(\cdot)$, fixing $\sigma = 1$. A Mahalanobis distance is a generalization of the Euclidean metric that takes into account the correlations between variables and hence, it is scale-invariant. The Mahalanobis distance between two vectors $\vec{\theta}_i$ and $\vec{\theta}_j$ is defined as:

$$d(\vec{\theta}_i, \vec{\theta}_j) = (\vec{\theta}_i - \vec{\theta}_j)^T \mathbf{M} (\vec{\theta}_i - \vec{\theta}_j) \tag{5}$$

where $\mathbf{M}$ can be any symmetric positive semidefinite real matrix, i.e. a matrix whose eigenvalues are all non-negative. If $\mathbf{M}$ is equal to the identity matrix, the Mahalanobis distance reduces to the Euclidean distance. Unfortunately, learning the matrix $\mathbf{M}$ directly requires enforcing a positive semidefinite constraint in the optimization problem, which is highly non-linear and is also expensive to solve. One way to get rid of such an expensive constraint is to decompose $\mathbf{M}$ as:

$$\mathbf{M} = \mathbf{A}^T \mathbf{A} \tag{6}$$

where $\mathbf{A}$ is a full rank matrix. By substituting Equation 6 into 5, we can express the Mahalanobis distance as the Euclidean distance after the mapping $\vec{\theta} \to \mathbf{A}\vec{\theta}$:

$$d(\vec{\theta}_i, \vec{\theta}_j) = \left\|\mathbf{A}(\vec{\theta}_i - \vec{\theta}_j)\right\|_2 = \left\|\mathbf{A}\vec{\theta}_{ij}\right\|_2 \tag{7}$$

We learn an appropriate kernel function between two camera poses by minimizing the loss function L defined as follows:

$$L = \sum_i \sum_{j \geq i} (y_{ij} - k_{ij})^2 \tag{8}$$

where y_{ij} is the target value or the similarity score between two camera poses, and k_{ij} is the estimate of the target value obtained by combining the Gaussian kernel and Mahalanobis distance. By means of Equation 6 we can rewrite the loss function L in Equation 8 in terms of the unconstrained matrix $\mathbf{A}$ (instead of $\mathbf{M}$). By minimizing Equation 8 we obtain a metric in an entirely non-parametric way. Finally, our kernel function measures the similarity between two camera poses as:

$$k_{ij} \equiv k(\vec{\theta}_i, \vec{\theta}_j) = \exp\left(-\left\|\mathbf{A}(\vec{\theta}_i - \vec{\theta}_j)\right\|_2\right) \tag{9}$$

Note that Equation 9 indeed describes the similarity between the two camera poses: when the output is equal to 1, the two cameras are identical, and conversely when the output is equal to 0, it means that they are extremely different with no visible 3D points shared between each other. In addition, Equation 9 shows how to compute the target estimates k_{ij} involved in the minimization problem described in Equation 8.

We define the target values y_{ij} as the mean of the ratios between the intersection of the common 3D points with respect to the number of 3D points visible to each of the two cameras:

$$y_{ij} = \frac{1}{2} \cdot \left| \frac{|X_i \cap X_j|}{|X_i|} + \frac{|X_j \cap X_i|}{|X_j|} \right| \tag{10}$$

Indeed, the above equation gives an estimate of the overlap between two views. Figure 2 depicts the similarity matrix for all pairs of camera poses in the dataset we used in our experiments.

Our algorithm learns its own scaling (encapsulated by matrix $\mathbf{A}$) and is therefore invariant to the scaling of the input vectors. This invariance to scaling is very important when dealing with complex functions such as visibility that involves cues with very different scales, such as the translation and the orientation. In our experiments we set the initial value of the matrix $\mathbf{A}$ to the identity matrix, and the algorithm converges in few iterations, satisfying the full-rank matrix $\mathbf{A}$ constraint. The Levenberg-Marquardt algorithm [17] has been used for all non-linear optimization in the metric learning procedure. Next, we will explain how to define the input vector $\vec{\theta_i}$ for each camera pose.

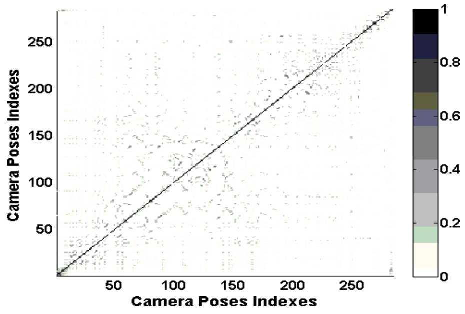


Fig. 2. Similarity score matrix for the St. Peter's Basilica dataset: Note that most of the camera poses have common visible 3D points with few camera poses. This means, that only the training data nearby a query pose θ_i are informative to correctly predict the visibility of a 3D point

A. Euclidean Distance and Viewing Direction Change

For each camera pose we define an input vector $\vec{\theta_i}$ that encodes geometrical information about the pose with respect to the global coordinate frame set of the 3D reconstruction. Each camera pose is parametrized by means of a vector $\vec{\theta_i} = \{T_i, R_i\}$ (3D vector for the translation and 4D unit quaternion for the rotation). In particular for our metric learning, we use two cues (difference in camera translation and viewing direction), $\vec{\theta_{ij}} = [d_R \ d_T]^T$, where d_T is the Euclidean distance between the translation of two cameras, and d_R defines the normalized inner product between the viewing directions of the two cameras (θ_i, θ_j).

B. Image Appearance Cue

Another important cue for a distance metric between two camera poses is their corresponding image appearance. The simplest way to utilize the image is by computing the sum of pixel-wise distances. However, the performance of this approach tends to drop considerably when dealing with wide-baseline images with high spatial variance. In this paper, we model the appearance by local histograms, which are more invariant to small camera motions and have been successfully applied to location recognition problems [14], [15] recently.

More specifically, the RGB features are accumulated over spatially localized 3×2 image grids. Within each grid, the RGB space is quantized into 50 bins. Hence, each image is represented as a 300-dimensional vector. Adding the image appearance cue, we have the following difference vector for any two cameras: $\vec{\theta_{ij}} = [d_R \ d_T \ d_H]^T$, where d_H is the Euclidian distance between two local histograms.

V. Speeding Up by Pre-pruning

In this section, we introduce a pre-pruning technique to decrease the complexity of computing the visibility for all the map elements, from the order of the map size $O(M)$ to $O(K)$ where K denotes the number of nearest neighbors of a given camera pose. A naive way of computing the visibility is to iterate over every one and apply Equation 3. The complexity of such an approach increases linearly with respect to the size of the map, M. However, the core observation is that the results of the visibility prediction using Equation 3 will be mostly zero, since most of the map elements in a large map would not be observed at all by the K Nearest Neighbors (KNNs) of the current query pose θ, effectively making the numerator in Equation 3 zero.

As a consequence, once we find the KNNs of the current query pose, we only need to predict the visibility for the subset of map elements seen at least once by these KNNs. Then, we can set the visibility to zero for the rest of the map elements without computing them at all. Finally, the locally weighted K nearest neighbor approximation for the visibility posterior is:

$$P(v_j = 1|\theta) \approx \frac{\sum_{i=1}^{K} k(\theta, \theta_i^{v_j=1})}{\sum_{i=1}^{K} k(\theta, \theta_i)} \tag{11}$$

where only the nearest K samples of the query pose $\Theta^K = \{\theta_1 \ldots \theta_k\}$ are considered.

VI. Experimental Results

We evaluate our algorithm using a real 3D database built from 285 photographs of St. Peter's Basilica in the Vatican City, as depicted in Figure 1. The SfM process is done as proposed in Snavely's paper [18] and will not be elaborated in this paper. The dataset comprises of 142,283 3D points, 466,222 image measurements, and 285 cameras. This difficult dataset is popular in the computer vision community and has been previously used to evaluate 3D reconstruction approaches such as [19].

A. KNNs Classification

We evaluate our Mahalanobis metric in terms of KNNs classification. For this purpose, we compare the predicted KNNs of our metric to the ground truth neighbors of all the poses from the training dataset. The ground-truth KNNs are obtained by means of the proposed similarity score between two camera poses as shown in Equation 10. For a query camera pose from the training set, we compute the similarity score for all the connected poses that share at least one visible feature in common. Then, we obtain the KNNs by sorting the camera poses in a decreasing order of the similarity scores. In this experiment, we used leave-one-out cross-validation, skipping the query camera pose from the training data.

First, we investigate the effectiveness of the cues used in our proposed distance metric. For KNNs classification, we compared the following metrics: Mahalanobis metric with two cues (translation, viewing direction), Mahalanobis metric with three cues (translation, viewing direction, local histograms) and the Euclidean distance of camera translation, viewing directions and image histograms separately. Figure 3(a) depicts the average KNNs classification error for all the camera poses of the dataset that see at least one 3D point. A predicted nearest neighbor will be classified as a false positive if that neighbor is not included in the ground truth neighbors set of a given query pose. Figure 3(b) depicts a more restrictive ratio, the average KNNs classification recall. This ratio is computed by considering only the first K ground truth nearest neighbors, i.e. even if a predicted neighbor shares some common 3D points with a query pose, we will classify that neighbor as a wrong one if it is not included in the first K ground truth nearest neighbors. For obtaining these graphs, we varied the number of neighbors K to consider in the prediction and compute the ratios according to this K.

We can observe in Figure 3(a) that the lowest error ratio is obtained with the proposed Mahalanobis metric with three cues (translation, viewing direction, histograms). The Mahalanobis metric with two cues also exhibits small error ratios similar to the ones obtained with three cues, especially when K is above 10.

The error slightly increases as long as we consider more NN in the prediction and a near constant error is obtained for a large number of neighbors. The reason for this, is that as long as we consider more NN, we are adding neighbors whose similarity score with respect to a query pose is low. For these neighbors, the differences in camera translations and local histograms can be very large, but viewing directions must be very similar. KNNs classification for high values of K is more challenging, and error ratios will slightly increase, since for these cases viewing directions play a more important role than camera translation and local histograms. Hence, for high values of K viewing directions, error ratios are similar to the ones obtained with the Mahalanobis metric with three and two cues. In the experiment in Figure 3(b) conclusions are similar. However, since we compute averaged recall ratios considering only the first K ground truth nearest neighbors, here we can observe a gain in adding the appearance cue into the Mahalanobis metric, and also a higher performance of the Mahalanobis metric with three and two cues with respect to viewing directions compared to Figure 3(a).

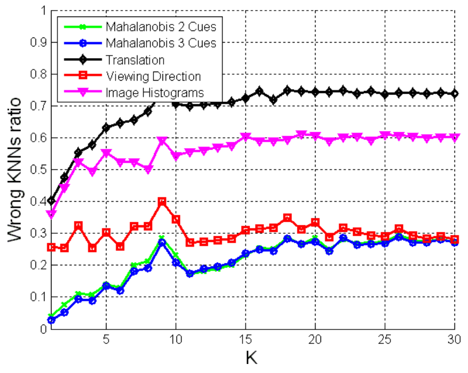
Viewing directions are a much more robust evidence than camera translation distances, due to the fact that people tend to shoot the same interesting object from multiple views which have large baselines between each other. In this case, even if the cameras can move wildly, the viewing direction remains approximately the same. Image histograms are also very useful cues, especially as viewing directions become less reliable as the cameras get close to the interested object. Imagine the scenario in which people pass by a certain facade of St Peter's basilica while taking pictures. Both translations and rotations of the cameras may change dramatically, yet the appearance of the images will be very similar, and so will the histograms of the images.

Figure 4 depicts two examples of KNNs classification, showing the different NN rankings that were obtained by means of the proposed Mahalanobis metric with three cues and the ground truth information. That is, a neighbor which is ranked as 1 should be a neighbor that shares the highest number of visible 3D points with respect to the query pose among all the connected poses. Figure 4(a) shows classification results considering only $K = 4$ neighbors of one of the images of the St. Peter's Basilica that was taken from St. Angel's Bridge at a distance of approximately 1 km from the basilica. Figure 4(b) depicts a more challenging classification scenario, since the photo was taken in the middle of St. Peter's Square, where appearance details are very similar between slightly different view points due to the similar sculptures surrounding the square. We can observe, that even if rankings are not exactly the same, even in a very difficult scenario such as Figure 4(b), all the first predicted neighbors share visible 3D points with respect to the query pose.

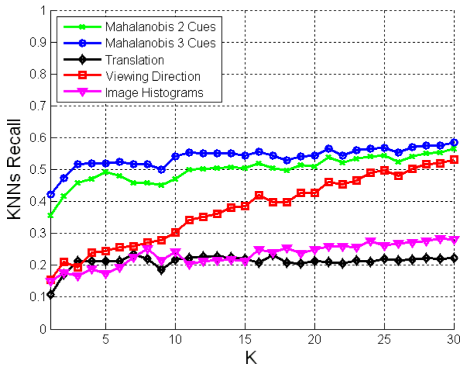
B. Visibility Prediction: Sensitivity to Noise and Number of Nearest Neighbors

Now we evaluate the performance of our visibility prediction with respect to different levels of noise, and study the influence of the number of nearest neighbors K in the resulting prediction. We evaluate the performance of our approach by means of *recall* versus *1-precision* graphs [20]. For visibility prediction, we define *recall* and *1-precision* as:

$$\begin{aligned} recall &= \tfrac{\#predicted\ visible}{\#real\ visible} \\ 1 - precision &= \tfrac{\#false\ predicted\ visible}{\#all\ predicted\ visible} \end{aligned} \tag{12}$$



(a)



(b)

Fig. 3. Evaluation of Metric Learning for KNNs classification. (a) Depicts the average KNNs classification error for all the camera poses of the dataset that see at least one feature whereas (b) shows the average correct KNNs classification recall only considering the K ground truth nearest neighbors.

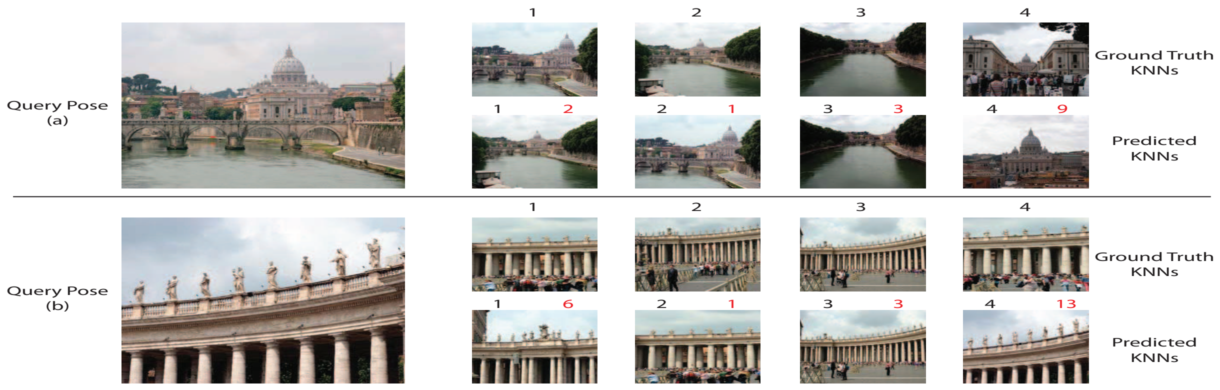


Fig. 4. KNNs ranking: (a) St. Angel's Bridge (b) St. Peter's Square. We show on the left the KNNs ranking results obtained from ground truth, and at the right, the ranking results obtained with our Mahalanobis metric (three cues). We show in red the ranking results obtained with the proposed metric, whereas for ground truth, we display results in bold. For this experiment, we only considered the first 4 nearest neighbors.

where we know the number of *real visible* 3D points for each of the poses from the dataset. To simulate the noisy signals of a GPS sensor in the city, we added noise with normal distribution to the ground-truth camera pose and then predict the visible 3D points for this noisy pose. We consider random Gaussian noise (mean μ, standard deviation σ) for all the camera translation and viewing direction components and then normalize the noisy viewing direction vector to unit length. In addition, we also added noise to the image histograms (the scale of image histograms is normalized between 0 and 1). Table I shows the two Gaussian noise settings that we used in our experiments, distinguishing two different levels of noise.

According to Equation 11 we decide if a feature is visible if its probability of being visible is higher than a fixed threshold. By varying the value of this threshold, we can obtain *recall* versus *1-precision* graphs. In this experiment we include all the camera poses as possible KNNs of the noisy camera pose.

Figure 5(a) depicts averaged *recall* versus *1-precision* graphs for all camera poses in the dataset. We added random Gaussian noise ($N1$ and $N2$ as defined in Table I) and considered K in the visibility prediction, comparing Mahalanobis metrics with 2 and 3 cues (denoted as M2 and M3 respectively). The average number of *real visible* 3D points per camera pose in this dataset is 1666.

Cue	**Noise Level** $N1$	**Noise Level** $N2$
Translation	$\mu = 15m, \sigma = 30m$	$\mu = 5m, \sigma = 10m$
Orientation	$\mu = 0.1$, $\sigma = 0.2$	$\mu = 0.05, \sigma = 0.1$
Histograms	$\mu = 0.0, \sigma = 0.3$	$\mu = 0.05, \sigma = 0.1$

TABLE I
RANDOM GAUSSIAN NOISE SETTINGS IN OUR EXPERIMENTS.

An important conclusion from Figure 5(a) is that a higher *recall* is obtained when considering a small number of nearest neighbors in the prediction. The reason for this is that as long as we increase the number of neighbors, we are also adding more 3D points in the prediction of Equation 11. Some of these 3D points may not be truly visible from the query noisy pose, but they may be visible from a local neighborhood of the query noisy pose. Typically, for large baseline 3D reconstructions, one camera view shares most of its visible 3D points with few camera poses, i.e. only the training data nearby a query pose is informative enough to correctly predict the visibility of a feature, as shown in Figure 2. In addition, an approximate 20% gain in *recall* can be obtained when the appearance cue is used. This cue becomes less important, however, when the number of neighbors increases. Furthermore, considering that we can have noisy position measurements from a GPS or another hand-held device, the image appearance is a very useful cue, since in general, it is the less affected by noise. Also, the reduction in *recall* when increasing the number of neighbors is more significant for the Mahalanobis 3 cues case. This is because some neighbors may look similar in terms of appearance but images can be taken from different viewpoints, as for example happens inside St. Peter's Square, where image appearance is similar and repetitive, but viewpoints are different.

C. Comparison with Heuristic Visibility Prediction

We compare our approach to a *length and angle visibility heuristic* that has been widely used in the literature [6]. This heuristic is very easy to compute and provides good results in non-cluttered and small environments. Feature visibility is calculated considering the difference between the viewpoint from which the feature was initially seen and a new viewpoint. This difference in viewpoint has to be below some length and angle ratio to predict the feature as visible. Usually a feature is expected to be visible if the length ratio $|h_i|/|h_{orig}|$ is close enough to 1 (in practice between 5/7 and 7/5 and the angle difference $\beta = \cos^{-1}((h_i \cdot h_{orig})/(|h_i||h_{orig}|))$ is close to 0 (less than 45° in magnitude). However, the main drawbacks of this criterion is that it can not deal with occlusions since it assumes a *transparent* world and that it has to predict its visibility for every feature, which can be computationally expensive for very large 3D reconstructions.

Usually for very large city-scale reconstructions such as the ones presented in [21], the number of camera poses is more than a hundred times smaller than the number of 3D points (e.g. in the Dubrovnik reconstruction we have $4,585$ poses and $2,662,981$ 3D points and in the St. Mark's Square we have $13,699$ poses and $4,515,157$ 3D points), which means that we can obtain a substantial speed up if we predict visibility by obtaining the KNNs of a query camera pose, instead of predicting the visibility for each feature individually.

We compare our approach with the heuristic visibility prediction described above, but instead of checking the visibility of each feature individually, which will incur in a high computational demand and low recall, we introduce some modifications. We first bound the regions for possible camera neighbors, using geo-spatial constraints (3D camera pose location) placing a sphere centered on the query pose. Then we predict the visibility according to the mentioned heuristic, just for those 3D points that are seen by the camera pose neighbors that lie inside the sphere. Finally, by varying the radius of the sphere, we can generate *recall* versus *1-precision* graphs.

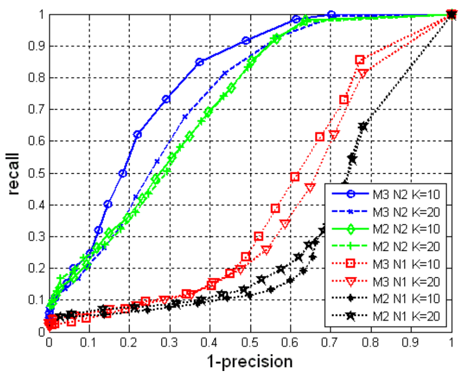
As can be seen in Figure 5(b), we obtain results with our visibility prediction which are superior to other common geo-spatial constraints based on translation and angle heuristics. Moreover, we can obtain a very high recall for very high precision values, i.e. we get rid of most of non-visible 3D points yielding a better descriptor matching and a faster localization. We obtained the graphs shown in Figure 5(b), considering noise level $N2$, and a number of neighbors equal to $K = 5$ for our visibility prediction. For the heuristic prediction, we used the default values: $5/7 < |h_i|/|h_{orig}| < 7/5$ and $\beta < 45^\circ$. We also performed another experiment considering a more restrictive length ratio keeping the same angle ratio: $3/4 < |h_i|/|h_{orig}| < 4/3$. In addition, we can also observe that there is a gain in performance comparing the graphs with $K = 5$ and the graphs in Figure 5(a), where K was set to 10 and 20.

A timing evaluation revealed that our MATLAB implementation runs faster than 30 Hz (or 33.3 ms per frame). Just predicting the visible 3D points for a query camera pose takes 12.20 ms for $K = 5$. Visibility heuristic in combination with geo-spatial pruning for a radius of 10 m takes 139.49 ms, whereas predicting the visibility of all the 3D points individually with the heuristic takes 4.44 s. All timing results were obtained on a Core 2 Duo 2.2GHz computer. As shown in 5(b), we can see that our algorithm obtained a more robust and faster visibility prediction, yielding better results than conventional approaches.

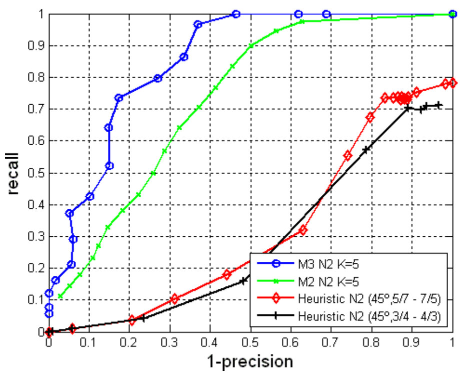
VII. Conclusions and Future Work

In this paper we proposed a novel method for predicting the visibility of 3D points in large-scale urban environments. In our approach, every map feature models its visibility with respect to the camera poses via non-parametric distributions. By means of a combination of Mahalanobis distance and a Gaussian kernel, we showed how to learn a similarity metric between two camera poses in an entirely non-parametric way that is invariant to the scale of the input vectors. We have discussed the importance of the cues used in our metric and shown the benefits of adding local image histograms instead of using only camera translation and viewing direction.

For very large 3D reconstructions, the number of map elements is more than a hundred times higher than the number of camera poses. Based on this observation, we think that for fast and robust vision-based localization over large 3D reconstructions, our algorithm can dramatically reduce the computation time and improve robustness, contrary to other common approaches that check the visibility of each of the 3D points from the dataset individually or pruning-out information based in geo-spatial constraints.



(a)



(b)

Fig. 5. Recall versus 1-precision graphs: (a) Evaluations under different noise settings and the number of neighbors K (b) Comparisons with respect to the heuristic visibility. The number of neighbors K is set to 5.

As future work, we are interested in applying our idea to incremental SfM reconstruction, by incrementally learning the Mahalanobis distance metric between camera poses, and then showing the results of our method in vision-based localization experiments under very large city-scale environments. Moreover, the addition of new cues to the proposed metric will be studied.

VIII. Acknowledgments

This work was supported in part by the Community of Madrid under grant CM: S-2009/DPI/1559(RoboCity2030 Project). Frank Dellaert and Kai Ni gratefully acknowledge support from the National Science Foundation through award IIS - 0448111 "CAREER: Markov Chain Monte Carlo Methods for Large Scale Correspondence Problems in Computer Vision and Robotics". The authors would like to thank Jon Cohen for his support in the audio recording of the supplementary video and other colleagues at University of Alcalá and Georgia Institute of Technology for many useful discussions. We would also like to thank Photosynth from Microsoft for the St. Peter's Basilica data.

References

[1] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography," *Commun. Assoc. Comp. Mach.*, vol. 24, pp. 381–395, 1981.

[2] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, , and F. Dellaert, "Detecting and matching repeated patterns for automatic Geo-tagging in urban environments," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[3] P. Alcantarilla, S. M. Oh, G. Mariottini, L. Bergasa, and F. Dellaert, "Learning visibility of landmarks for vision-based localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Anchorage, AK, 2010.

[4] H. Wuest, A. Pagani, and D. Stricker, "Feature management for efficient camera tracking," in *Asian Conf. on Computer Vision (ACCV)*, 2007.

[5] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Eur. Conf. on Computer Vision (ECCV)*, Marseille, France, 2008.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 6, 2007.

[7] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[9] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[10] Z. Zhu, T. Oskiper, S. Samarasekera, R. Kumar, and H. Sawhney, "Real-time global localization with a pre-built visual landmark database," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[11] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.

[13] C. Atkeson, A. Moore, and S. Schaal, "Locally weighted learning," *AI Review*, vol. 11, pp. 11–73, April 1997.

[14] A. Torralba, K. Murphy, W. Freeman, and M. Rubin, "Context-based division system for place and object recognition," in *Intl. Conf. on Computer Vision (ICCV)*, 2003, pp. 273–280.

[15] K. Ni, A. Kannan, A. Criminisi, and J. Winn, "Epitomic location recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[16] K. Q. Weinberger and G. Tesauro, "Metric learning for kernel regression," in *In Proceedings of the Eleventh International Workshop on Artificial Intelligence and Statistics (AISTATS)*, Puerto Rico, 2007.

[17] M. Lourakis, "levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++," [web page] `http://www.ics.forth.gr/~lourakis/levmar/`, 2004.

[18] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo Tourism: Exploring image collections in 3D," in *SIGGRAPH*, 2006.

[19] K. Ni, D. Steedly, and F. Dellaert, "Out-of-Core Bundle Adjustment for Large-Scale 3D Reconstruction," in *Intl. Conf. on Computer Vision (ICCV)*, 2007.

[20] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.

[21] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building Rome in a Day," in *Intl. Conf. on Computer Vision (ICCV)*, 2009.