

Minimal Scene Descriptions from Structure from Motion Models

Song Cao
Cornell University
caosong@cs.cornell.edu

Noah Snavely
Cornell University
snavely@cs.cornell.edu

Abstract

How much data do we need to describe a location? We explore this question in the context of 3D scene reconstructions created from running structure from motion on large Internet photo collections, where reconstructions can contain many millions of 3D points. We consider several methods for computing much more compact representations of such reconstructions for the task of location recognition, with the goal of maintaining good performance with very small models. In particular, we introduce a new method for computing compact models that takes into account both image-point relationships and feature distinctiveness, and we show that this method produces small models that yield better recognition performance than previous model reduction techniques.

1. Introduction

In recent years, the increasing availability of online tourist photos has stimulated a line of work that utilizes structure-from-motion techniques to construct large-scale databases of images and 3D point clouds [18, 1], for a variety of applications, including location recognition [21, 6, 9, 14, 10]. These location recognition methods often directly match features (such as SIFT [11]) in a query image to descriptors associated with 3D points. These databases of 3D points, however, can be very large—ranging in size from a few million points in a single location, to hundreds of millions when multiple places are considered together [10]. For purposes of modeling and visualization, the denser the 3D points the better. However, for other applications, such as recognition, there are advantages in having fewer points, such as reduced memory and computation requirements. This brings up an interesting question: how much data do we need to describe a location? What is a minimal description of a place?

One way to make this question concrete is to define it as a visibility covering problem [9, 12]: every possible image that one could take of the location should see some minimal number of 3D points stored in the reconstruction. Such a covering constraint makes it likely that a new image of the

scene will match a sufficient number of 3D points to enable pose estimation. Based on this idea, prior methods have used the visibility relationships between images and points in the database to compute reduced 3D point sets that cover the database images. However, another important factor is **distinctiveness**: in order to ensure accurate matching, one should select a subset of points that are distinct (rather than selecting points with very similar appearance). In this paper, we show that by computing a reduced scene description that takes into account both coverage and distinctiveness, one can compute very compact models that maintain good recognition performance.

We incorporate these considerations into a new point selection algorithm that predicts how well new images will be recognized using a probabilistic approach. We evaluate our algorithm on several standard location recognition benchmarks, and show that our computed scene representations consistently yield higher recognition performance compared to previous model reduction techniques.

2. Related Work

Our algorithm is inspired by the K -covering algorithms used in prior 2D-to-3D matching-and-pose-estimation systems [9], but has two key differences compared to this prior work. First, we consider **point appearance** in order to select visually distinctive points; second, our selection algorithm uses a **probabilistic model** of visibility as opposed to the strictly combinatorial methods used in prior work.

Park *et al.* select a subset of 3D points using mixed-integer quadratic programming [12]. A limitation of this approach is the computational hardness of the underlying optimization problem, making it difficult to scale up to large, world-wide datasets [10]. Choudhary *et al.* model point visibility probabilities to guide a 3D matching and pose estimation algorithm at runtime [4]. Our goal is different in that we aim to select a subset of database points in advance without knowledge of a specific query image. Although the probabilistic formulation in [4] works well for modeling inter-point and inter-image relationships given a query image, a direct adaptation of this approach, by modeling inter-point relationships, is computationally prohibitive due to the non-linear composition

of probabilities and combinatorial explosion of point sets. Irschara *et al.* use synthesized views from a point cloud to cover a 3D scene [6]. We take a different approach that directly models image-point relationships, but it is possible to extend our method by adding synthesized views as an additional set of “images” to cover in our algorithm.

Our use of distinctiveness as a factor in selecting points is related to prior work on identifying “confusing” features for recognition [17, 7]. For instance, Knopp *et al.* use image geotags to identify features that appear at multiple disparate locations, and incorporate this information into a bag-of-words recognition framework [7]. In our case, we select individual 3D points rather than visual words, and do not require GPS information. Hence, we have more fine-grained control over the set of features to avoid or select. Philbin *et al.* use descriptor learning to find a non-linear transformation of the descriptor space in order to better separate true feature matches from false ones [13]. Our work is orthogonal in that it seeks to find a subset of distinctive features in the standard descriptor space, but our algorithm could easily incorporate learned distance metrics.

Finally, our method is related to prior work on feature selection for identifying discriminative features [20]. Li and Kosecka propose a method for identifying highly discriminative individual features for a discrete set of locations [8]. Doersch *et al.* propose to use a discriminative clustering approach to find visual elements that are most distinctive for a city (such as Paris) [5]. Cao *et al.* use discriminative learning on clusters of images to define different distance metrics representing locations [3]. Turcot and Lowe use image matching process to select reliable visual words [19]. While we also favor distinctive image descriptors in our approach, we again do so at a much more fine-grained level than bag-of-words models, and also seek to maximize *coverage* of the dataset as well as discriminability.

3. Computing Minimal Point Sets

In this section we describe our algorithm for computing minimal scene representations, starting with background.

We begin by running structure from motion (SfM) to reconstruct one or more scenes that form a database for use in recognizing and posing new images [1]. The result of running SfM on an image set \mathcal{I} of size m is a 3D point set \mathcal{P} of size n , (typically $n \gg m$), as well as a *visibility matrix* M of size $m \times n$ defining the visibility relationships between images and points, where $M_{ij} = 1$ if point P_j is visible in image I_i in the reconstructed 3D model, and $M_{ij} = 0$ otherwise. This matrix can also be interpreted as a bipartite *visibility graph* \mathcal{G} on images and points, where an edge links each image to each point visible to that image. In addition to a 3D location, each point $P_j \in \mathcal{P}$ also has a feature descriptor, for instance the average SIFT descriptor of the features used to triangulate P_j [9]. Such a reconstruction

can be used to recognize the pose of new photos via 2D-to-3D matching and pose estimation techniques [15, 10]. To register a query photo, we extract features from the photo, match them to features in the database using approximate nearest neighbors [2], and robustly estimate the absolute camera pose using the matched points. Key to this process is to find a sufficient number of correct feature matches between the query image and the database.

Our goal is to compute a more compact database with a much smaller set of points $\mathcal{P}' \subset \mathcal{P}$, such that \mathcal{P}' captures as much of the information in the full model as possible. In particular, we wish to be able to correctly register as many new query images to the subset \mathcal{P}' as possible.

K -cover algorithm. The prior work of Li *et al.* [9] begins with the assumption that the distribution of query images is similar to the distribution of database images. Hence, they use coverage of the database images as a proxy for coverage of query images of interest. They formulate this as a K -cover (KC) *problem* on the visibility graph \mathcal{G} : select a minimum subset of points such that each database image sees at least K points in the subset. Finding such a minimum set is a combinatorially hard problem, and so they use a greedy algorithm that starts with the empty set, and incrementally adds the next point P_j that maximizes the *gain* in coverage achieved by adding P_j to the current set \mathcal{P}' :

$$G_{KC}(j, \mathcal{P}') = \sum_{I_i \in \mathcal{I} \setminus C} M_{ij} \quad (1)$$

where C is the set of images that are already “covered” at least K times by points in \mathcal{P}' , i.e. $C = \{I_i | \sum_{P_\ell \in \mathcal{P}'} M_{i\ell} \geq K\}$, and hence do not contribute to the gain of a point. This algorithm runs until no further point contributes a positive gain (or until a target percentage of images, e.g. 99%, are covered). We denote the gain function in Eq. 1 as G_{KC} in reference to the K -cover algorithm that it corresponds to. In short, the K -cover algorithm ensures good coverage of the database images by greedily selecting points that “cover” the most uncovered images. In what follows, we will define alternative gain functions that model additional aspects of the recognition problem.

In order to compute the pose of a query image, a minimum of three feature matches to 3D points are required (or four, in case the camera intrinsics are unknown, or six, in case a full projection matrix is desired). In practice, many more matches are desirable; hence, Li *et al.* use large values for K (e.g., $K = 100$).

Our approach. Like the K -cover algorithm, we use a greedy algorithm to incrementally create a compact subset of points. However, we define a different gain function using a probabilistic framework: we seek to select a subset of points that maximizes the *probability* of registering a new image, while minimizing the number of points selected. At a high level, our approach considers two aspects:

1. **coverage**, i.e., the subset covers the scene in that any new image has a high probability of seeing a large number of points, and
2. **distinctiveness**, i.e., the features we select are sufficiently distinct from one another in appearance.

We found that both of these aspects contribute to the goal of finding good feature matches between a query image and the database.

3.1. Maximizing expected coverage

We first describe our approach to ensuring coverage. As in the K -cover algorithm, we want to select points in such a way that any query image that matches the scene sees a certain minimum number of points. The K -cover algorithm treats covering the database in a strictly combinatorial way. However, because feature detection and matching are noisy processes, we instead view this problem from a probabilistic perspective, where the database images are treated as samples from some underlying distribution of images of a scene. Hence, we consider points being visible in images as random events with certain probabilities, rather than as simply binary variables. In particular, we define that each point P_j is visible in each database image I_i with probability p_{ij} . We experimented with different methods for defining these probabilities, such as forms of smoothing the bipartite visibility matrix M . We found that simply using a constant value p for $M_{ij} = 1$ and 0 for $M_{ij} = 0$ worked well, though we note that finding better ways to estimate p_{ij} is an interesting avenue for future work.

Given this probabilistic model of point visibility, our goal is to find a subset of points \mathcal{P}' that maximizes the probabilities of each image seeing at least K points in \mathcal{P}' . More formally, let $v_{i,\mathcal{P}'}$ denote the random variable representing the number of points in the selected set \mathcal{P}' that are visible in database image I_i . Our objective is to maximize

$$\sum_{i \in \mathcal{I}} \Pr(v_{i,\mathcal{P}'} \geq K) \quad (2)$$

i.e., the sum of probabilities that each image I_i sees at least K points in the selected set \mathcal{P}' . If we assume that each observation of a point is independent, then the distribution of each random variable $v_{i,\mathcal{P}'}$ is described by a binomial distribution if p_{ij} is a constant p , or a Poisson binomial distribution (a generalization of the binomial distribution) if p_{ij} varies for each point observation.

To balance the objective in Eq. 2 with the desire for a compact model, we set a target probability p_{\min} , and seek that $\Pr(v_{i,\mathcal{P}'} \geq K) \geq p_{\min}$ hold for all images I_i , with $|\mathcal{P}'|$ as small as possible. To achieve this goal, we adopt the greedy approach of repeatedly selecting the point P_j that

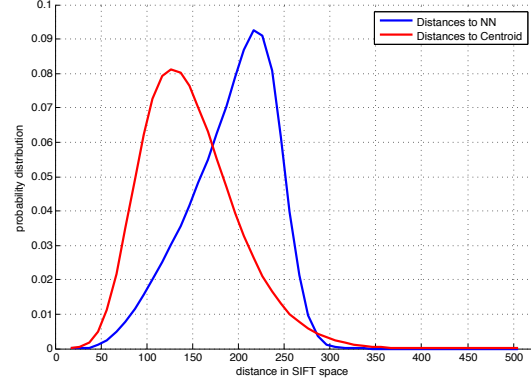


Figure 1. **Distributions of descriptor distances.** The curve in red shows the distribution of descriptor distances between image feature descriptors and their associated point descriptor centroids. The curve in blue shows the distribution of distances between a point descriptor centroid and its nearest neighbor in the full database. Note the significant overlap between these distributions. These distributions are generated from the Dubrovnik dataset by considering all database points and their associated image feature descriptors. (Figure best viewed in color.)

maximizes the following gain function:

$$G_{KCP}(j, \mathcal{P}') = \sum_{i \in \mathcal{I} \setminus C} \Pr(v_{i,\mathcal{P}' \cup \{P_j\}} \geq K) - \Pr(v_{i,\mathcal{P}'} \geq K) \quad (3)$$

Here, KCP refers to our probabilistic K -cover algorithm. This measures the gain in *expected coverage* of the database achieved by adding a point P_j to the selected set. As in Eq. 1, C is the set of already-covered images, but in a probabilistic sense: $C = \{I_i | \Pr(v_{i,\mathcal{P}'} \geq K) \geq p_{\min}\}$. In other words, our algorithm starts with $\mathcal{P}' = \emptyset$, and repeatedly adds the point P_j that maximizes the expected gain defined in Eq. 3. We describe this algorithm in more detail in Section 3.3.

However, there is a bootstrapping problem with this formulation. If, at some point in the algorithm (e.g., at the beginning), an image I_i sees fewer than $K - 1$ points in \mathcal{P}' (i.e., $\sum_{j \in \mathcal{P}'} M_{ij} < K - 1$), then the gain for adding any new point to \mathcal{P}' w.r.t. image I_i is zero. In this case, we cannot effectively compute the gain in coverage of image I_i by adding another point. To avoid this problem, we bootstrap by first choosing an initial set of points that “ K -covers” the images, i.e. a set \mathcal{P}' that satisfies $\sum_{P_j \in \mathcal{P}'} M_{ij} \geq K$ for each image I_i . We next describe how we select an initial set of distinctive points in Section 3.2, and then show how to select a final covering set by maximizing Eq. 3.

3.2. Appearance-aware initial point set selection

We now describe how we select the initial point set that “ K -covers” the images while considering distinctiveness of appearance. Consider a particular point P_j . P_j is associated

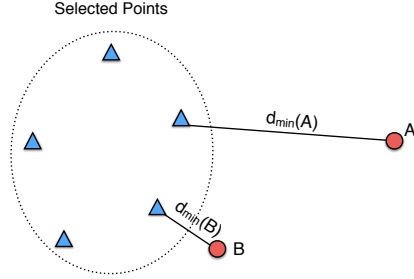


Figure 2. **An illustration of appearance-aware point selection.** The image above shows points in feature descriptor space (reduced to 2D for visualization purposes). Blue triangles represent point descriptors that are already selected by our algorithm, while red circles represent descriptors of candidate points to select next. Suppose that candidate points A and B cover exactly the same number of images in $\mathcal{I} \setminus C$, and thus would lead to equal gains in the K -cover algorithm. However, since the minimum distance of A to the selected point set $d_{\min}(A)$ is larger than that of B $d_{\min}(B)$, A is likely to result in fewer mismatches. during feature matching if selected. Hence A is preferred by our appearance-aware point selection algorithm.

with a set of individual SIFT descriptors \mathcal{D}_j in two or more database images; in our work, for compactness, we represent P_j with the centroid of these descriptors, \bar{D}_j . Assuming that the database descriptors \mathcal{D}_j are representative of other query descriptors that will later match this point, we want these descriptors $D \in \mathcal{D}_j$ to be closer to \bar{D}_j in SIFT space than to the descriptor of any other selected point. To motivate this approach, consider Figure 1, which shows two distributions: (1) the distribution of distances between image features $D \in \mathcal{D}_j$ and their centroid \bar{D}_j (red), and (2) the distribution of distances between centroids \bar{D}_j and the nearest centroid of a different point (blue). Although the expected distance (red) from a given descriptor to its centroid (true match) is smaller than the expected distance (blue) between two nearby centroids (false match), there is significant overlap between these two distributions. While this simple analysis is not a comprehensive study of feature mismatches, it suggests that there is significant opportunity for query features to match to incorrect points (i.e., because a feature is closer to a nearby, incorrect, database point in descriptor space).

One way to increase the probability of query features matching to the correct database point is to select points that are far away from each other in descriptor space. Since our greedy selection algorithm adds points to \mathcal{P}' sequentially, when computing the gain of a point P_j under consideration, we implement this strategy by down-weighting a point’s gain according to its minimum distance to the current set of selected points \mathcal{P}' . Figure 2 illustrates this intuition.

We evaluated a range of options for this weighting approach. In the end, we found that a simple approach worked well: let $d_{\min}(j)$ be the minimum distance in descriptor

space between P_j and the already selected point set \mathcal{P}' . We down-weight the gain of P_j if $d_{\min}(j)$ is lower than a threshold d . Specifically, we define the gain of a point P_j as:

$$G_{KCD}(j, \mathcal{P}') = w_d(d_{\min}(j))G_{KC}(j, \mathcal{P}') \quad (4)$$

where $G_{KC}(j, \mathcal{P}')$ is defined in Eq. 1, and the weight $w_d(d_{\min}(j))$ is defined as

$$w_d(d_{\min}(j)) = \begin{cases} d_{\min}(j)/d, & d_{\min}(j) < d \\ 1, & d_{\min}(j) \geq d \end{cases} \quad (5)$$

This weight varies from 0 to 1 linearly in the range $[0, d]$. One simple interpretation of this weight is as a rough approximation of the chance of a correct feature match for query features—higher for points that are more distinct given the current set of descriptors, and lower for points that are less distinct—and thus the gain function above can be interpreted as an “expected gain” in image coverage, incorporating the possibility of a mismatch. Given this interpretation, the threshold d should be set so as to try and separate the two distance distributions in Figure 1. We choose $d = 180$ based on the empirical overlap of the two distributions.

We use this modified gain function in our greedy K -cover algorithm (the gain function G_{KCD} in Eq. 4 stands for “ K -cover with distinctiveness”). There is an order dependency in our greedy algorithm, but since the order in which points are added depends largely on their coverage, our approach can be seen as a trade-off between our two main objectives of coverage and distinctiveness.

Like the K -cover algorithm, our modified covering algorithm terminates if the gain for *every* point is zero (i.e., no unchosen points will cover any not-yet-fully-covered images). Figure 3 shows the effect of including descriptor distances into the selection method, by showing distributions of distances between nearest neighbors in selected point sets with and without considering distinctiveness. We see that the point set selected by our method has larger expected nearest neighbor distances in descriptor space, which will tend to decrease the rate of false matches in the feature matching phase of the recognition pipeline. We use this appearance-aware selection method to seed our probabilistic point selection algorithm, which we describe next.

3.3. Probabilistic K -cover algorithm

We now have an initial point set selected by our appearance-aware selection algorithm (KCD). This allows us to bootstrap our probabilistic point selection method. Recall that rather than treating the visibility matrix as binary, our probabilistic approach treats this matrix as a set of noisy observations of visibility, and selects a small number of additional points to add to \mathcal{P}' such that the number of images that satisfy $\Pr(v_{i, \mathcal{P}'} \geq K) \geq p_{\min}$ is as large as possible. That is, unlike the K -cover algorithm, which seeks to combinatorially “cover” the images at least K times, we set a

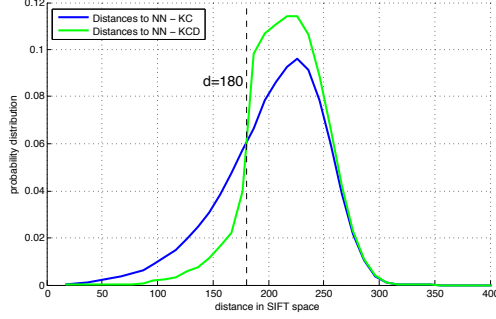


Figure 3. **Distributions of distances with and without appearance-aware selection.** This plot illustrates the “density” of two sets of point descriptors, by showing the distribution of distances between nearest neighbors in SIFT space between points within each set. The blue plot is for the set selected with the basic K -cover algorithm (KC), while the green plot is for the set selected with our appearance-aware selection algorithm (KCD) using the threshold $d = 180$. Both point sets contain 311,343 points selected from the Landmarks dataset [10]. Note that the KCD algorithm pushes points further away from one another on average.

minimum probability value p_{\min} and our goal is to achieve $\Pr(v_{i,\mathcal{P}'} \geq K) \geq p_{\min}$ for each image I_i .

Like the K -cover algorithm, we use a greedy approach, but choosing the point P_{j^*} that maximizes *expected gain*, as defined in Eq. 3. This gain function is defined in terms of probabilistic coverage, $\Pr(v_{i,\mathcal{P}'} \geq K)$, which, in its simplest form, is a sum over a binomial distribution for $x \geq K$. In particular, given the initial set of points \mathcal{P}' , we first evaluate $\Pr(v_{i,\mathcal{P}'} \geq K) = 1 - \Pr(v_{i,\mathcal{P}'} < K)$ for each image I_i , i.e. the probability that an image I_i sees at least K points in the selected point set \mathcal{P}' . Suppose I_i is covered C_i times by the initial point set \mathcal{P}' , and every edge encodes a point visibility with probability p . Then from the binomial distribution we have

$$\Pr(v_{i,\mathcal{P}'} = K') = \binom{C_i}{K'} p^{K'} (1-p)^{C_i-K'}. \quad (6)$$

Hence, we can compute $\Pr(v_{i,\mathcal{P}'} \geq K)$ as

$$\Pr(v_{i,\mathcal{P}'} \geq K) = \sum_{K'=K}^{C_i} \Pr(v_{i,\mathcal{P}'} = K'). \quad (7)$$

These distributions for images with different levels of coverage are illustrated in Figure 4.

To choose the next point to add to \mathcal{P}' , we pick the point P_{j^*} that maximizes the sum of expected gains for all images, defined in Eq. 3. To compute this expected gain inside our greedy algorithm, the naive approach is to re-calculate (7) for all images. However, evaluating (7) can be expensive, as this must be done at each iteration of adding points, once for each point. In the simple case where p_{ij} is constant over the

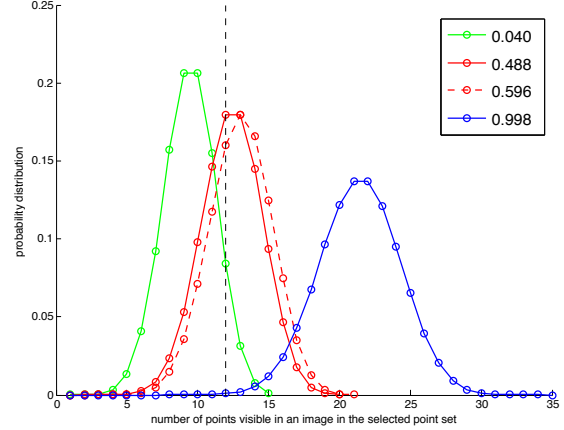


Figure 4. **Binomial distributions for images covered by different numbers of selected points.** The plots above show probability distributions for number of visible points, $v_{i,\mathcal{P}'}$, for three different images (solid lines). The images corresponding to the green, red solid, and blue curves are covered by 15, 20, and 35 points, respectively; in this example the probability of a positive point observation is set to $p = 0.6$. The legend shows the probability mass on the right side of the line $K = 12$ for each image, i.e. $\Pr(v_{i,\mathcal{P}'} \geq K)$, the probability that each image sees at least K points in \mathcal{P}' . As more points are added, the distributions will shift from left to right (green \rightarrow red \rightarrow blue). The dotted red distribution is the red distribution after adding a single new point that is visible in this image; the corresponding probability value $\Pr(v_{i,\mathcal{P}'} \geq K)$ has increased from 0.488 to 0.596 (a gain of 0.108). The objective of our probabilistic point selection algorithm is to select points that maximize the increase in expected gain $\Pr(v_{i,\mathcal{P}'} \geq K)$ for all uncovered images.

visibility graph (and hence we the probability distributions of interest are binomial), we can simply pre-compute all possible distributions, each with a different C_i , and use these as lookup tables. However, for better generality and extensibility to Poisson binomial distributions, our implementation uses a different approach to compute $\Pr(v_{i,\mathcal{P}'} \geq K)$.

First, for each image I_i , we compute and store the full distribution of $v_{i,\mathcal{P}'}$ once, after the initial points are selected, using (6). Then we re-use and update the distribution of $v_{i,\mathcal{P}'}$ in later iterations. More specifically, from the independence assumption, $\Pr(v_{i,\mathcal{P}' \cup \{j\}} \geq K)$ can be written as

$$\Pr(v_{i,\mathcal{P}' \cup \{j\}} \geq K) = p_{ij} \Pr(v_{i,\mathcal{P}'} = K-1) + \Pr(v_{i,\mathcal{P}'} \geq K). \quad (8)$$

Hence, the gain of a point P_j for an image I_i is simply $p_{ij} \Pr(v_{i,\mathcal{P}'} = K-1)$. Hence from Eq. 3, the total gain $G_{KCP}(j, \mathcal{P}')$ of a point P_j can be written as

$$G_{KCP}(j, \mathcal{P}') = \sum_{i \in \mathcal{I} \setminus \mathcal{C}} p_{ij} \Pr(v_{i,\mathcal{P}'} = K-1). \quad (9)$$

We then choose the point P_{j^*} that maximizes $G_{KCP}(j, \mathcal{P}')$, add P_{j^*} to the set \mathcal{P}' , and update the distribution of $v_{i, \mathcal{P}' \cup \{P_{j^*}\}}$ for each image I_i that P_{j^*} is visible in, using

$$\begin{aligned} \Pr(v_{I_i, \mathcal{P}' \cup \{P_{j^*}\}} = K') = \\ p_{ij} \Pr(v_{i, \mathcal{P}'} = K' - 1) + (1 - p_{ij}) \Pr(v_{i, \mathcal{P}'} = K'). \end{aligned} \quad (10)$$

Discussion. We have considered several variants of our approach. For instance, it would be natural to add the appearance distinctiveness weighting into the probabilistic K -cover approach, and to that end, we have tried methods such as converting the minimum distance to the nearest neighbor to a probability using a global distribution of such distances. We found that our simple approach above worked as well, however, perhaps because the additional points we add are primarily improving coverage of a set of points that are already distinctive. However, this is an interesting topic for further exploration.

3.4. Full point set reduction algorithm

0. Initialize point set $\mathcal{P}' = \emptyset$.

Initial point set selection:

1. Given K and threshold d , select the point $P_{j^*} \in \mathcal{P}$ that maximizes $G_{KCD}(j, \mathcal{P}')$ (Eq. 4), and add P_{j^*} to \mathcal{P}' .
2. Repeat Step 1 until all images are covered by at least K points.

Probabilistic K -cover algorithm:

3. Given a parameter p_{\min} , and the point set \mathcal{P}' generated from Steps 1 and 2, evaluate $\Pr(v_{i, \mathcal{P}'} \geq K)$ for each image I_i using (7), and mark those images with $\Pr(v_{i, \mathcal{P}'} \geq K) \geq p_{\min}$ as covered.
4. Select the point P_{j^*} that maximizes the gain function $G_{KCP}(j, \mathcal{P}')$ defined in (9), and add P_{j^*} to \mathcal{P}' .
5. For each image I_i that sees point P_{j^*} : update I_i 's distribution using (10), re-evaluate $\Pr(v_{i, \mathcal{P}'} \geq K)$ using (7) and mark I_i as covered if $\Pr(v_{i, \mathcal{P}'} \geq K) \geq p_{\min}$.
6. Repeat from Step 4 until a specified percentage of images are covered.

4. Implementation

Efficient descriptor comparisons. Our initial point set selection method requires computing the descriptor distance between a candidate point and its nearest neighbor for each point in the selected set, and a naive approach would involve comparing all candidate descriptors to all selected descriptors.¹ However, we note that the expected gain $G_{KCD}(j, \mathcal{P})$ of a point P_j can only decrease across the iterations of the

¹One could use a kd -tree to speed up nearest neighbor computation, but in our case the tree would have to be dynamic since the selected point set grows over time.

selection process, because both terms in Eq. 4 are submodular set functions of \mathcal{P} , which will only increase in its size. Hence, our algorithm can maintain an upper bound on the expected gain $G_{KCD}(j, \mathcal{P})$ for each point P_j . At each iteration of searching for the best point to add, we can skip considering a point if its upper bound is less than or equal to the gain function value of the current best candidate point. The upper bound for each point is initialized by that point's degree, the maximum possible score for each point. This bound is updated on any selection iteration where that point is not skipped. Using this method, a large number of points can be skipped; we observe empirically that approximately $\mathcal{O}(\log n)$ points are evaluated per iteration, where n is the total number of points in the input set \mathcal{P} .

To further speed up our method, for each point P_j we also store the nearest neighbor and its corresponding distance in the selected set \mathcal{P} so far, as well as the size of \mathcal{P} during the last evaluation of P_j . This allows us to pick up where we left off when finding P_j 's nearest neighbor the next time we evaluate P_j . All in all, we found the running time of our appearance-aware initial point selection process to be acceptable. For example, for the Dubrovnik dataset with $K < 20$, the selection process runs in under a minute. For $K = 80$, the process takes about ten minutes. An upper bound on running time of our KCD algorithm is $\mathcal{O}(nc)$, where n is the number of points in \mathcal{P} and c is the number of selected points ($|\mathcal{P}'|$), since each selected point is compared to at most n other points.

Parameters. In all experiments, we define $p_{\min} = 0.99$ to be the minimum probability for an image to be ‘‘covered’’. We use 99% as the target coverage termination condition: that is, the K -cover algorithm terminates when 99% images are covered at least K times, and our algorithm terminates when 99% images are covered at least K times with probability greater than p_{\min} . We use a constant value $p = 0.6$ for all p_{ij} 's², and a descriptor distance threshold $d = 180$ for Eq. 5. We evaluated several values of d , and found that the results are fairly insensitive to its value.

5. Experiments

In this section, we evaluate the performance of our algorithm on several datasets, including the Dubrovnik dataset of Li *et al.* [9], the Aachen dataset of Sattler *et al.* [16], and the much larger Landmarks dataset [10]; these three datasets are summarized in Table 1.

We evaluate three approaches to computing minimal scene descriptions: the K -cover algorithm (KC) [9], our initial point set selection algorithm only (KCD), and our

²We arrived at $p=0.6$ by considering the empirical ratio between the number of inlier points when registering a query image and the number of points seen by that image in the original model; the results across a few datasets were in the range 0.5-0.6, and 0.6 worked well in practice.

Dataset	# DB Imgs	# 3D Points	# Queries
Dubrovnik [9]	6,044	1,886,884	800
Aachen [16]	4,479	1,980,036	369
Landmarks [10]	205,813	38,190,865	10,000

Table 1. Summary of datasets used in our experiments.

full approach including the probabilistic K -cover algorithm (KCP). All methods output a list of points to keep in the original 3D point cloud database. We use each subset of points to construct a reduced database, and use the algorithm of [10] to register the query images for each dataset. We record the percentage of successfully registered images and use it as a measure of how well the point set represents the original database. We are particularly interested in *very compact* scene descriptions (small K), and understanding how well we can represent scenes with a small fraction of points.

Numbers of points. In order to fairly compare different methods, it is easiest to compare the performance of scene descriptions with the same number of points. However, given a particular K , the number of points required to cover a database is generally smaller for the K -cover algorithm (KC) than with KCD, since KC selects points with maximal coverage without considering point appearance. Hence to compare performance, we run KCD until it selects the same number of points as the KC algorithm with the same K value. This could slightly favor the K -cover algorithm, as our initial point set selection algorithm is terminating early.

Since our full approach (KCP) consists of two stages, in which the initial selection KCD alone selects slightly more points than the KC algorithm, again, more points will be selected by KCP compared to KC using the same K value. To account for this, we use a lower value of K to select the initial point set (around pK), and continue running the KCP algorithm until it has selected the same number of points as KC. We show results on all datasets in Table 5.

For each dataset, we plot starting with the smallest K where we get close to 50% registration rate. Hence the K values vary for different datasets. Table 5 shows the results for KC, KCD, and KCP.

Initial point set selection. In all datasets, adding the descriptor distance-based weight $w(d_{\min}(j))$ in our gain function (4) for KCD improves the performance compared to the K -cover algorithm for nearly all values of K . The improvement is especially significant when K is low (and hence the number of selected points is small). However, as K is set higher, this advantage becomes less prominent, perhaps because images see more points and mismatches are less detrimental (i.e., coverage starts to win out).

Probabilistic K -cover. Table 5 also shows that our full approach (KCP) consistently outperforms the K -cover algorithm, and further improves on the gains achieved by our

Dubrovnik Dataset [9]				
# query images: 800, registered by full set: 99.50%				
K	12 (9)	20 (12)	30 (20)	50 (35)
# points	5,788	10,349	17,241	31,752
% points	0.31%	0.55%	0.91%	1.68%
KC	58.00%	77.06%	86.00%	91.81%
KCD	62.88%	78.88%	87.38%	92.50%
KCP	64.25%	79.13%	87.25%	93.38%

Aachen Dataset [16]				
# query images: 369, registered by full set: 88.08%				
K	30 (20)	50 (32)	80 (52)	100 (65)
# points	13,299	23,675	40,377	52,161
% points	0.67%	1.20%	2.04%	2.63%
KC	50.95%	62.06%	66.40%	71.27%
KCD	54.20%	63.14%	69.38%	72.36%
KCP	56.37%	64.23%	70.19%	73.98%

Landmarks Dataset [10]				
# query images: 10,000, registered by full set: 94.33%				
K	6 (4)	9 (6)	12 (9)	20 (12)
# points	140,306	222,161	311,035	571,864
% points	0.37%	0.58%	0.81%	1.50%
KC	44.84%	59.86%	69.56%	81.06%
KCD	45.45%	61.26%	70.59%	81.04%
KCP	45.90%	61.50%	71.87%	81.45%

Table 2. **Registration performance on Dubrovnik, Aachen, and Landmarks datasets.** KC stands for the K -cover algorithm, KCD stands for our appearance-aware point selection algorithm, and KCP stands for our full approach. Point sets of the same size are selected using the three algorithms, then used in the same registration algorithm [10] to evaluate the percentages of query images that are successfully registered to the database. Smaller K values (in brackets) are used to initialize our KCP method. For each experiment, we show the number of points in the reduced model, the percentage of total points this represents, and the performance of the three methods. For comparison, we also show the performance of [10] using the full set of input points.

KCD algorithm. For instance, KCD improves recognition performance on the Dubrovnik dataset by nearly 5% (58% to 62.9%) when $K = 12$, and KCP improves performance further to 64.2%. To check whether KCD is indeed helping, we tried initializing KCP with KC instead of KCD, but found that this performs worse than KCP initialized with KCD.

How much performance are we losing with our compactness? Table 5 also shows the performance of state-of-the-art methods that utilize the full point set [10, 15], which use full models and have a minimum registration rate of 88% on these datasets. It is worth noting that although compared to them, our method has lower raw registration performance (Table 5), our resulting models are much more compact

(< 3% of the size of the full model). With the limited portion of database we use, our algorithm still performs surprisingly well in registering new images. For instance, we can recognize over 70% of the query images in the Aachen dataset with only 2% of the 3D points in the full model ($K = 80$).

How much benefits are we getting through compactness? As well as dramatically improving memory use, we have also observed reduced registration time. For instance, the smaller reduced models ($K \leq 30$) process queries in about half the time compared to the full model. This improvements result from the compactness of data structures and efficiency in rejecting false images, both of which stem from the compactness of the 3D point set.

6. Conclusions and Discussions

We have proposed a new method for computing compact point models from structure from motion datasets of 3D points, exploring how a little data can often go a long way. Our method can be used to reduce the memory and computational cost of a location recognition system. Our main contribution is to combine two key considerations—coverage and distinctiveness—in an algorithm for computing compact models of places. Based on our experiments, we conclude that both coverage and distinctness are important considerations, and that our probabilistic approach also aids in computing representative models. We believe these ideas could also be used in other recognition settings where compact models are sought.

One limitation of our approach is that we require more pre-computation time than the K -cover algorithm, even with the optimizations in Section 4, since we compare descriptor distances between points. However, this can be done as a batch process once for a dataset. Another limitation is that we use greedy algorithms to optimize for both coverage and distinctness; in the future, we hope to investigate more sophisticated global selection algorithms. Our probabilistic model also makes the simplifying assumption that each edge in the visibility graph for a scene represents a random event with equal probability p . We have tried some simple variants that estimate different probability values p_{ij} per edge with similar performance, but we believe that further exploration of probabilistic models of point visibility can likely improve performance further. For instance, analyzing negative information in a visibility graph—i.e., when a point should be visible but was not detected in an image—may assist in modeling probabilities. Further, understanding the distribution of appearance for each individual 3D point could also yield further improvements.

Acknowledgements. This work was funded in part by grants from the National Science Foundation (IIS-0964027, IIS-1149393, and IIS-1111534), and by support from the Intel Science and Technology Center for Visual Computing.

References

- [1] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building Rome in a day. In *ICCV*, 2009. 1, 2
- [2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. of the ACM*, 45(6):891–923, 1998. 2
- [3] S. Cao and N. Snavely. Graph-based discriminative learning for location recognition. *CVPR*, 2013. 2
- [4] S. Choudhary and P. Narayanan. Visibility probability structure from SfM datasets and applications. In *ECCV*. 2012. 1
- [5] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? *SIGGRAPH*, 2012. 2
- [6] A. Irschara, C. Zach, J. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009. 1, 2
- [7] J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. In *ECCV*, 2010. 2
- [8] F. Li and J. Kosecka. Probabilistic location recognition using reduced feature set. In *ICRA*, 2006. 2
- [9] Y. Li, N. Snavely, and D. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010. 1, 2, 6, 7
- [10] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012. 1, 2, 5, 6, 7
- [11] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1
- [12] H. S. Park, Y. Wang, E. Nurvitadhi, J. C. Hoe, Y. Sheikh, and M. Chen. 3d point cloud reduction using mixed-integer quadratic programming. In *CVPR Workshops*, 2013. 1
- [13] J. Philbin, M. Isard, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *ECCV*, 2010. 2
- [14] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2D-to-3D matching. In *ICCV*, 2011. 1
- [15] T. Sattler, B. Leibe, and L. Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012. 2, 7
- [16] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt. Image retrieval for image-based localization revisited. In *BMVC*, 2012. 6, 7
- [17] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007. 2
- [18] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *SIGGRAPH*, 2006. 1
- [19] P. Turcot and D. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *Workshop on Emergent Issues in Large Amounts of Visual Data, ICCV*, 2009. 2
- [20] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *ICCV*, 2003. 2
- [21] W. Zhang and J. Kosecka. Image based localization in urban environments. In *Int. Symp. on 3DPVT*, 2006. 1