Matching a photograph to satellite images

Roger Grosse and Matthew Johnson
MIT CSAIL and LIDS

Abstract

We describe a method for generating location estimates for a photograph by automatically matching it to satellite imagery. First, a user labels interest points in the photograph corresponding to stationary objects also identifiable from the air. We show how to compute robust invariants which can be matched against a large database of satellite views, and we evaluate several invariants on simulated data. We also explore the problem of automatic tree detection, and combine it with the invariants technique in a proof-of-concept demonstration of our image location method on a photograph from the Arnold Arboretum.

1. Introduction

A system to automatically estimate the spot where an photograph was taken based solely on matching the photograph to satellite imagery would have many exciting applications. For example, it could serve as a tool to locate terrorists or criminals from supposedly anonymous photographs or videos, or it could allow for the easy markup of personal photographs for high-granularity location-based organization without the use of GPS.

In our formulation of the problem, we allow for human markup on the ground-based photograph, since a photograph that we are interested in locating would likely warrant a few moments of a human's time. However, we require all satellite image processing and all comparisons between the ground-based photograph and the satellite imagery to be performed automatically, since the assumption is that the database of satellite images to compare against is large.

While low-level features such as SIFT have been successful at matching images up to small transformations (as in the PhotoSynth system [7]), for our problem the perspective change is so great that we would not expect these low-level features to be preserved. Indeed, experiments have shown SIFT is robust to some small levels of rotation, but certainly not to the transformation from satellite to ground. Instead, we use the geometry of large, stationary objects, since we can perform the matching using well-studied geometric mathematics. In particular, we focus on the relative locations of trees on the ground plane, because they are discrete stationary objects which can (in principle) be reliably detected in both images, and because we are left with a 2-D rather than a 3-D matching problem. However, the mathematical framework and basic techniques we develop should be extendable to other objects (including non-point objects such as roads).

In this paper we describe a method for identifying possible matches by performing nearest-neighbor queries on geometric invariants computed from the estimated tree positions. To make this possible, we define a geometric invariant on an ordered tuple of trees, as well as a way of assigning a canonical order to an unordered cluster of trees. We compare several alternatives for the invariant and canonical order on simulated data, and we show a demonstration of our first implementation of a complete system applied to arboretum photographs.

This paper proceeds as follows: first, we briefly outline related work in Section 2. Second, in Section 3 we discuss the possible formulations of the geometry of the problem along with some fundamental tradeoffs. Next, in Section 4, we describe our proposed system, including our approaches to the problems of tree detection and geometric invariant comparison. Finally, in Section 5 we both evaluate our geometric methods on simulated data and show a demonstration of a first implementation of the complete system.

2. Related work

The task of registering photographs with location and map data has been approached from several directions. The recent Viewfinder project [9] allows humans to align a photo with Google Earth's 3-D world models. Their method uses either user input or GPS data to find the geographic location of the image, and hand-labeled point correspondences to fix the other camera parameters. Because of its reliance on hand-labeled correspondences, their method does not seem directly practical for matching against large satellite databases, but it may be useful as a post-processing step to refine the matches our method retrieves.

Another approach is the IM2GPS project [3], in which image summary statistics such as color histograms, line features, and GIST features are compared to a large database of

GPS-tagged photographs to estimate the rough geographic location. The IM2GPS method is able to consider photos from practically anywhere on the planet and effectively return a distribution over geographic regions. However, while the IM2GPS method provides distributions over regions, countries, and continents of the entire planet, we aim to solve the slightly different problem of locating an image's exact position, which is likely only possible within much smaller search areas. Thus, these methods could be used in conjunction, with IM2GPS generating region estimates and our method refining them into exact locations.

The Astrometry.net automated astrometry project [4] is also related. Their system compares relative geometric positions of stars in a photograph (via geometric invariants) to those in a large database, and is thus able to estimate the camera parameters of the photo and identify the stars within. The project has been very successful, and its excellent performance motivates similar geometric invariant techniques in our method. However, it should be noted that the possible transformations in the astrometry task are much more restricted than in our own task, since astronomical images are always taken from approximately the same cosmological viewpoint (Earth) and so the camera parameters only affect scale, rotation, and translation. In comparing satellite photos of the Earth to ground-based photos, there are several extra degrees of freedom in the perspective change.

As alluded to in the introduction, SIFT features have had much success in finding correspondences between photographs with much smaller transformations than the ones we are interested in. While SIFT features do not survive significant perspective changes, there has been work on extended affine-invariant low-level features such as ASIFT [5]. However, even these features assume a flat surface, and thus would not apply to complex 3-D objects such as trees. Such features may still be useful for capturing information about ground texture, although this would probably require finding satellite images from the same time of year as the photo was taken.

Finally, there has been a large amount of research into geometric invariants by the vision community, and the use of invariant indexing has been explored in applications such as object recognition [6]. Our method is able to leverage some of the geometric invariant framework that has been developed, and it can be viewed as a novel application of such techniques.

3. Geometric models

We assume we are given a set of hand-labeled trees in a photograph which we must match against a large database of processed satellite images. In particular, we assume the human labels each tree at the point where its trunk meets the ground plane. Because we can also interpret the trees in the satellite image as lying in a 2-D plane, we model the image ground plane as a 2-D projective transformation of the "true" world plane.

Depending how much work the user puts in, we can recover varying amounts of information from the photograph. In order from least to most specific, we can directly recover the locations of the points up to a projective, affine, similarity, or rigid transformation. Then, the problem becomes a matter of matching a set of query points against a large database in a way that is invariant to the corresponding transformation.

In particular, if the user simply labels the locations of the trees on the screen, the image points can be any projective transformation of the world points. If the user additionally provides a horizon line, we can recover the affine shape of the points. If the user labels a square on the ground plane, we can extract the locations up to a similarity transformation (and the problem becomes equivalent to the astrometry problem [4]). Finally, if we assume the photograph is taken from a known height, we have the locations up to a rigid transformation (translation and rotation). In all four cases, we may assume the transformation is orientation-preserving.

Each transformation has an associated number of degrees of freedom (dof) d. Since an (ordered) tuple of k points on the plane has $2k$ degrees of freedom, the set of points contains $2k - d$ degrees of freedom modulo the transformation. This quantity determines the maximum number of dof that an invariant may have, as well as the minimum number of points required to find an unambiguous match. The possible transformations and their properties are summarized in Table 1.

This choice of transformations gives us a tradeoff between the specificity of the representation and the labeling noise. On the one hand, the invariants for more general transformations are more prone to measurement noise. Also, they require more trees to have been correctly labeled by the tree detection algorithm. On the other hand, more specific transformations require more detailed input from the user, which provides an additional source of noise. We believe the optimal transformation model depends on the particular photograph. For instance, if the horizon line is clearly visible, the user might as well label it; however, we have found it difficult to accurately estimate an occluded horizon line, and in such cases, the projective model may be more accurate.

For the remainder of the paper, we will focus primarily on the affine transformation case, since it is the one we found most appropriate for the data used in our demo.

4. Procedure

The pipeline of our proposed method is summarized in Figure 1. First, both satellite and ground photographs are labeled with tree positions, the former being automatically

Transformation	dof	points needed	preserves	requires labeling
Projective	8	5	lines, cross-ratio, convex hull	points on screen
Affine	6	4	ratio of areas, parallel lines	horizon
Similarity	4	3	ratio of lengths, angles	square on ground
Rigid	3	2	length	height of camera

Table 1. Comparison of different transformation models: degrees of freedom of the transformation, minimum number of points to estimate an invariant, some quantities that are preserved, and additional information required from the user (each transformation requires the labels of those transformations listed above as well).

processed. Automatic tree detection methods are discussed in Section 4.1. The ground photograph may also be labeled with auxiliary information, particularly that required to recover affine shape as described in Section 4.2. Next, invariants are computed as "fingerprints" for the cluster geometry, as described in Sections 4.3 and 4.4, while compensating for anisotropic noise is discussed in Section 4.5. Finally, matching is established by comparing the invariants in the two images.

4.1. Tree detection

The problem of tree detection in satellite and aerial imagery has been approached several times, particularly in research related to Geographic Information Systems (GIS). Methods vary in complexity from simple intensity maxima and template matching to incorporation of geometric information. We chose to implement two tree detection methods: template matching and Viola-Jones boosting. However, since we have not tuned and engineered these methods, it is likely that much better performance is possible, even with these same methods.

4.1.1 Template Matching

Template matching is the method of sliding a small prototypical tree image across a larger image and evaluating L2 distance between the template and the underlying image pixels, giving the "distance from template" as a function of position. Any local minima below some threshold are taken to be tree detections. Such a technique has been employed for tree detection with considerable success in the past, and it is included in the comparison paper [2]. An example output using a single template is shown in Figure 8, and the performance is discussed in Section 5.2.

An advantage of this method is that it does not require large amounts of labeled training data, but instead the selection of a few template trees. Such templates could be chosen on a per-region or even per-image basis without requiring significant human effort. However, it is not clear how well the method will scale with region complexity, e.g. to more varied forests or even urban scenes. (Choosing templates specific to a region may be sufficient, because a system such as IM2GPS [3] can identify the rough location of the image.)

4.1.2 Viola-Jones Boosting

Viola-Jones boosting has had much success in real-time face detection [8]. This boosting method trains a cascade of weak classifiers to achieve both speed and accuracy. Since the method is not specific to face detection, it has been employed for various other detection tasks, and here we evaluate its potential for tree detection.

One disadvantage of V-J boosting relative to template matching is that it requires much more hand-labeling work from the user. Also, though the classifier is computationally inexpensive to run, the training of the classifier can take a considerable amount of time, even on modern processors.[1] However, we believe it has the potential to generalize better than template matching to different regions of the world. A sample run of the our trained Viola-Jones tree detector can be seen in Figure 2.

4.2. Recovering affine shape from a photograph

Let us consider how to recover from an image the affine shape of a set of points on a ground plane, assuming the user has labeled the horizon. For simplicity, let us assume that the person taking the picture is facing directly forward, so that the image plane is orthogonal to the ground plane. (A generalization of this method where this orthogonality assumption does not hold is described in [1].) We may choose any orthogonal coordinate system we like for 3-space, so let's define the y axis as the direction normal to the ground plane and the z axis as normal to the image plane. In other words, the ground plane is defined by the equation $y = y_0$, for $y_0 < 0$, and the image plane is defined by $z = f$.

Finally, we assume the transformation from the image plane to pixels is given by a scaling and a translation; since the scaling factor can be factored into the focal length f, we can model this transformation as simply a translation. Therefore, the pixel locations (u, v) for a given point are

[1]In our evaluation we only trained 10 stages, which corresponds directly to a high false-positive rate (since false positives decrease exponentially with the number of stages, and most applications seem to use many more than 10 stages).

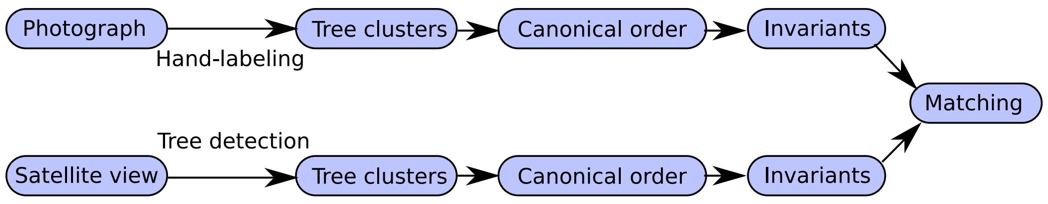


Figure 1. Flowchart of our system.

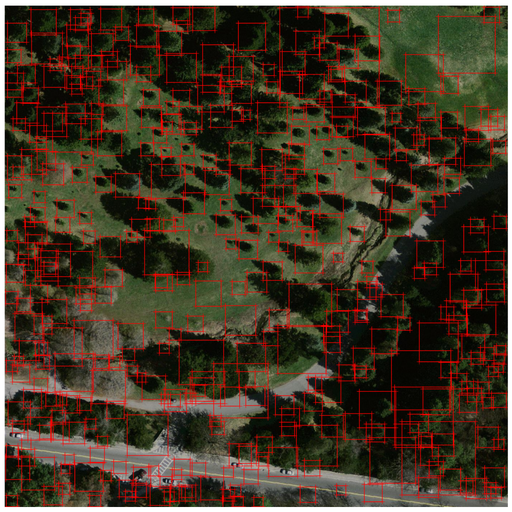


Figure 2. Output of the Viola-Jones boosting tree detection algorithm on a satellite view of the arboretum. The red boxes indicate detected trees. Note that the erroneous detections are primarily false positives.

given by:

$$u = \frac{fx}{z} + u_0 \qquad v = \frac{fy}{z} + v_0.$$

By letting z approach infinity, we find that $v = v_0$ is the equation of the horizon line. For a point (x, y_0, z) on the ground plane, we get that

$$z = \frac{fy_0}{v - v_0} \qquad x = \frac{z(u - u_0)}{f} = \frac{y_0(u - u_0)}{v - v_0}.$$

Now, we still do not know the values of f, y_0, or u_0, but changing these values only causes an affine transformation to our set of points. Therefore, we may choose any nontrivial values for these constants to retrieve the true locations of the points up to an affine transformation.

4.3. Invariants

Suppose we are given an ordered tuple of k points $(p_1, \ldots, p_k)$ on the ground plane. We can define an affine invariant as follows: we consider p_1, p_2, and p_3 to be "special." For any three pairs of non-collinear points, there is a unique affine transformation which maps the first point of each pair to the second one. We compute the affine transformation T which maps p_1, p_2, and p_3 to an equilateral triangle. Then, we concatenate the coordinates of the transformed $k - 3$ non-special points $T(p_4), \ldots, T(p_k)$ to get the invariant. Since each point has two degrees of freedom, this invariant has $2(k - 3)$ degrees of freedom. We saw in Section 3 that this is the most degrees of freedom we can hope for in an affine invariant.

The same procedure yields invariants for similarity or projective transformations, except that the similarity invariant uses two special points instead of three, while the projective invariant uses four special points. The similarity case is the same invariant used by Astrometry.net [4].

4.4. Canonical ordering for affine transformations

Recall that, to build the tree database, we index all clusters of k trees such that all the trees are sufficiently close to each other. For each k-cluster in the hand-labeled photograph, we wish to find its best matches in the database. However, since our invariants are defined over *ordered* tuples, this would seemingly require one of two unappealing strategies: (a) index all $k!$ permutations of each cluster, or (b) store only one permutation but run $k!$ separate queries. Since neither of these approaches is practical, we instead define a *canonical ordering* of the trees in a cluster, where the ordering is invariant to affine transformations. This way, for each (unordered) cluster, we simply need to store the invariant for its canonical ordering. When we search for a cluster in our database, we only need to search for its canonical ordering.

We define the canonical ordering as follows. Because affine transformations preserve ratios of determinants, the three "special" points will be the triplet with the largest determinant. (Conveniently, the transformation mapping these points to the equilateral triangle is likely to be the most stable, and therefore give the most reliable estimates of the invariant.) There are still 6 possibilities for the order of these three points. We define the first point p_1 to be the one with the minimum distance to any other point in the transformed space, and the second and third points p_2 and p_3 to be in counterclockwise order. Finally, we put all of the non-special points in order by their angle from p_1, going counterclockwise, starting from p_2. (Because we know the camera is above the ground plane, we can choose the affine shape in a way that preserves orientation, and therefore these last steps are invariant to affine transformation.)

4.5. Modeling uncertainty

As we mentioned previously, the schema for invariants described in Section 4.3 preserves all available information in the noiseless case. However, in reality, both the automatic tree detection and the hand-labeling are noisy, and sometimes the invariant is unstable and therefore amplifies the noise. Figure 3 shows a scatterplot of estimated invariants for ten different randomly generated 4-tuples, each with different amounts of random noise generated. Clearly, different sets of random points show different error covariances in the invariants.

Because the tree detection is an automated procedure, we may assume that the noise it contributes is much larger than the noise in the hand-labeling. Assuming that a particular tree was correctly detected, the estimated location may not be at the exact center of the tree. Let's suppose the offset of each detection from the center is given by a Gaussian with (unknown) variance $\sigma^2 I$. Assuming σ^2 is small, we can approximate the error in the invariant as a linear function of the detection offset. Thus, the error in the invariant will be a gaussian with some variance Σ. For a given k-tuple, we can estimate Σ by randomly adding small white gaussian noise to each of the points, computing the invariants, and using the empirical covariance matrix of these invariants. This gives us the noise covariance up to a scale factor, which is sufficient to rank the matches. We can compute Σ for each of the tree clusters as we add it to the database.

Finally, given a query invariant a (which we take to be exact), we rank the tree clusters b in our database according to our noise model:

$$p(b|a) = N(b|a, \Sigma_b) = \frac{1}{|\Sigma_b|^{1/2}} \exp\left((b-a)^T \Sigma_b^{-1} (b-a)\right).$$

5. Experiments

5.1. Simulations

In order to understand the tradeoffs in the design of our system, we have run simulations on randomly generated data. Suppose we are trying to match a set of hand-labeled trees against our database. There are several questions we must answer:

1. Should we use a projective, affine, or similarity transformation model?

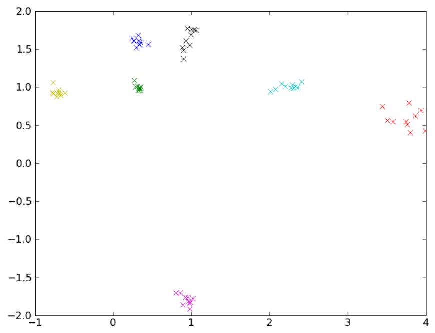


Figure 3. Ten random 4-tuples were generated from a uniform distribution, and white gaussian noise was added to each. The invariants of the noisy observations are plotted. Note that the noise covariance is non-isotropic and varies greatly from one 4-tuple to another.

2. What algorithm should we use for matching?

3. How many trees should be in the cluster we try to match?

Naturally, we would like to maximize our probability of finding a correct match. There are three ways we can fail to do so:

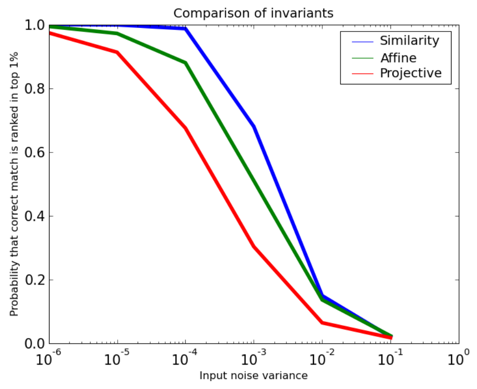
1. The tree detection algorithm misses at least one of the trees.

2. The canonical order of the query or the correct match is wrong.

3. The estimated invariant of the query or the correct match is noisy.

The first source of error, missed detection, only depends on the number of trees in a cluster. The more we try to match, the greater the probability of a missed detection. The other two sources of error are more subtle, and we present simulations illustrating the tradeoffs.

First, we compared the three kinds of invariants. To estimate the probability of getting the canonical order correct, we randomly generated k-tuples from the unit box and added small amounts of white gaussian noise. If the canonical ordering was the same with and without noise, we counted it as correct. Similarly, to estimate the reliability of the invariant, we repeatedly generated 1000 k-tuples. We selected one of the tuples, added noise to it, and then ranked all 1000 tuples according to their invariants. If the correct match was in the top 10, we counted it as a success. The results of these experiments are shown in Figure 4. As we would expect, the more general transformation models

(a) 

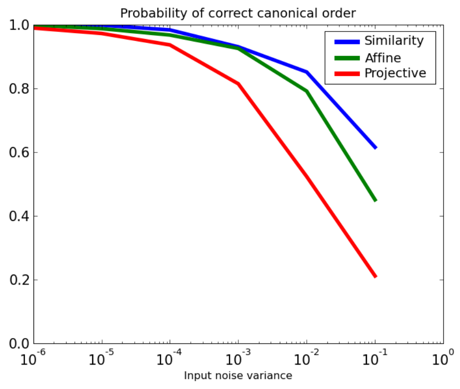
(b) 

Figure 4. Comparison of different invariants. (a) Probability that the correct match is in the top 1% returned by the algorithm, for randomly generated k-tuples. (b) Probability of correct canonical order. In both cases, we used the minimum number of points which can define the invariant (as given in Table 1).

are less robust to noise.[2]

Next, we compared four different matching algorithms. First, the most basic was simply to compute the invariants of each of the k-tuples and rank them by squared Euclidean distance. Second, we used the probabilistic approach described in Section 4.5. Third, we tried exhaustively fitting the transformations from each of the indexed tuples to the query and ordering them by the closeness of the fit. This exhaustive procedure is far too expensive to apply in practice, but it provides an upper bound on the performance we can expect from an invariant-based method. Finally, we tried a cascade approach where all of the tuples were ranked according to the probabilistic approach, and then the top 5% of matches were re-ranked using the exhaustive procedure.

As shown in Figure 5 (a), for affine and projective invariants, there is significant room for improvement relative to exhaustive search. Figure 5 (b) compares the four different matching procedures. The probabilistic approach does not appear to do any better than the basic least-squares approach.[3] The cascade approach closes much of the gap between the probabilistic and exhaustive approaches, suggesting that it is both fast and efficient in practice.

Finally, we looked at the tradeoff between different numbers of points in the tuple. As shown in Figure 6, using more points increases the specificity of the invariant, leading to a larger probability of the correct match being in the top 1%. The drawback is that the canonical form is less likely to be correct. In our experiments with real data, we used 5 points in each cluster.

5.2. Example

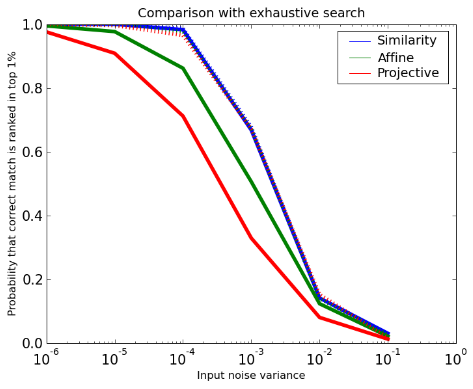
Finally, we tried our system out on a small-scale experiment with real images. We photographed a scene at the Arnold Arboretum and attempted to register it against satellite images from Google Maps. The photograph is shown in Figure 7 (a), along with the hand-labeled horizon line and trees. Using the procedure from Section 4.2, we retrieved the affine shape of this tree cluster. The affine shape is shown, in its canonical order, in Figure 7 (b).

Next, we applied our template-matching approach from Section 4.1 to detect the trees, and the result is shown in Figure 8. (This figure represents the full set of trees indexed.) The tree detector successfully labeled most of the trees which were similar in size to the template, but failed to detect trees which were significantly larger or smaller than the template. We believe this problem can be solved by searching with multiple templates. We indexed all of the clusters of five trees, such that the trees were all within 60 pixels of one another. This gave a total of 16,006 clusters.

We computed the affine invariants of the query and of the indexed clusters, and ranked all of the clusters according to the least-squares criterion. (We have not had time to try the other criteria yet.) Out of these 16,006 clusters, the correct match was ranked 10th. This suggests that, even with the noise introduced by hand-labeling and automatic tree detection, our procedure provides enough information to find good matches. There were about 20 tree clusters which looked very similar to the query shape, suggesting that it would be difficult to improve this result simply by

[2]This experiment underestimates the performance of the affine invariant in practice, because the tuples were randomly generated without respect to the canonical order. As described in Section 4.4, our canonical order is chosen such that the invariant is likely to be stable.

[3]In our original simulations, we measured the expected rank of the correct match, rather than the probability of the rank being in the top 1%. Using this other statistic, the probabilistic approach gave a large improvement over least-squares. We believe this is because the expected rank is dominated by the most unstable invariants, where the probabilistic approach is most likely to help. On the other hand, if the correct match is in the top 1%, the invariant was probably stable anway, perhaps explaining why the probabilistic approach doesn't help in these cases. We report the newer statistic because it more directly reflects the overall success rate of the system.

(a) 

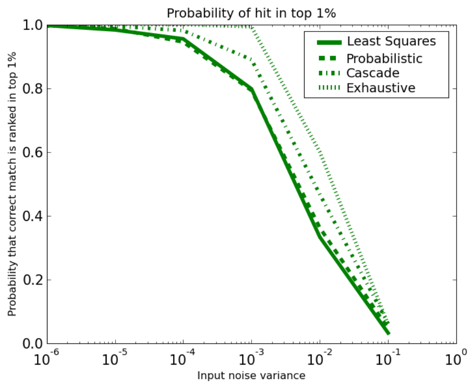
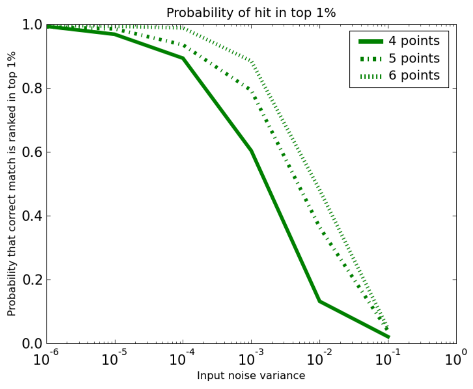
(b) 

Figure 5. Comparison of different matching algorithms. Both figures show the probability that the correct match is in the top 1%. (a) For all three transformation models, matching by invariant is shown in solid, and matching by exhaustively fitting transformations is shown in dashed lines. (All three curves for the exhaustive approach line up with the curve for similarity invariants.) Clearly, using invariants loses a lot of information, especially for the projective model. (b) Comparison of four matching algorithms for the affine model: least-squares matching of invariants, probabilistic matching of invariants (Section 4.5), the cascade of ranking algorithms, and exhaustive search (both defined in Section 5.1).

(a) 

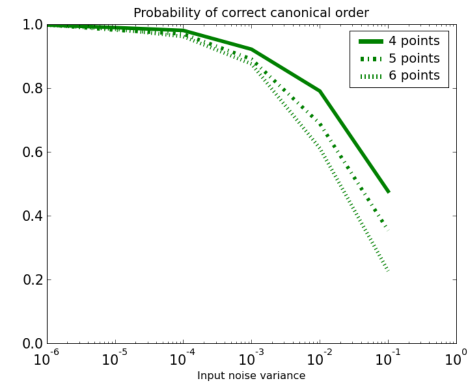
(b) 

Figure 6. Comparison of different numbers of points for affine transformations, matching invariants with the probabilistic approach. (a) Probability of correct match in top 1%. (b) Probability of correct canonical order. Using more points gives a more specific invariant, at the cost of a less reliable canonical order.

improving the invariants. Instead, to scale up our system, we believe it is necessary to match multiple clusters from the photograph and apply a voting procedure to select a single best match.

References

[1] A. Criminisi. Single view metrology. *International Journal of Computer Vision*, pages 123–148, 2000.

[2] M. Erikson and K. Olofsson. Comparison of three individual tree crown detection methods. *Mach. Vis. Appl.*, 16(4):258–265, 2005.

[3] J. Hays and A. A. Efros. im2gps: estimating geographic information from a single image. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[4] D. W. Hogg, M. Blanton, D. Lang, K. Mierle, and S. Roweis. Automated Astrometry (Invited). In R. W. Argyle, P. S. Bunclark, and J. R. Lewis, editors, *Astronomical Data Analysis Software and Systems XVII*, volume 394 of *Astronomical Society of the Pacific Conference Series*, pages 27–+, Aug. 2008.

[5] J. Morel and G. Yu. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[6] C. A. Rothwell. *Object Recognition through Invariant Indexing*. Oxford Science Publications, 1995.

[7] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[8] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

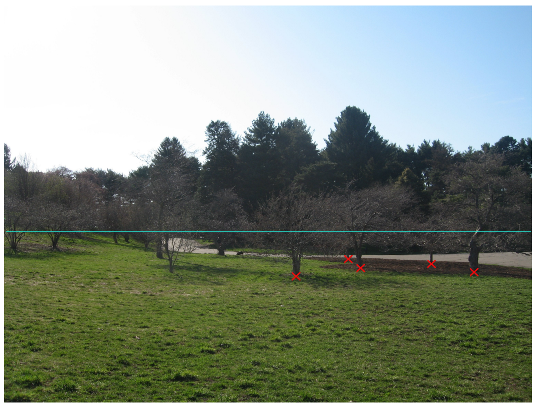
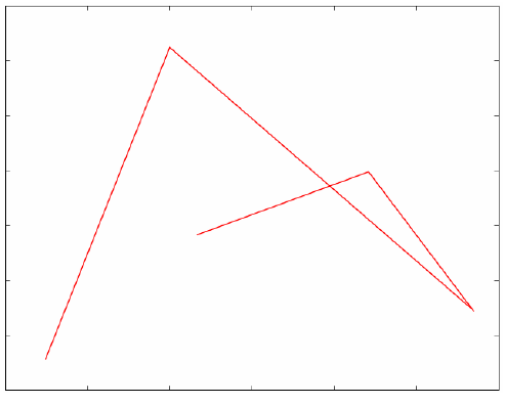
(a)  (b) 

Figure 7. (a) The query photo, with trees and the horizon labeled. (b) The affine shape extracted from the labels, in its canonical order.

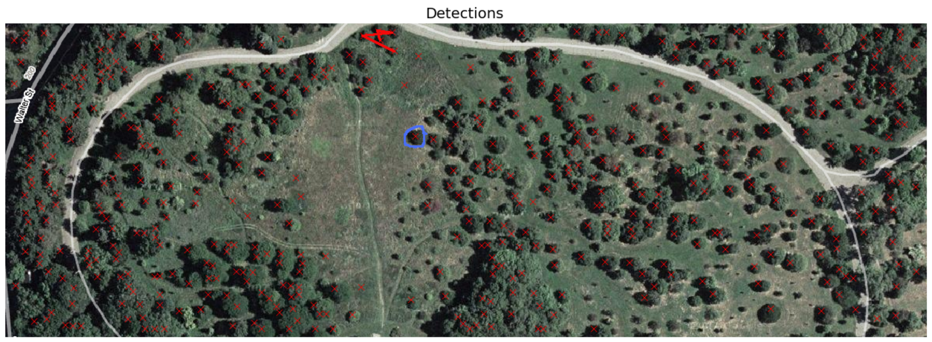


Figure 8. Output of the template-based tree detection algorithm on the satellite view of the arboretum. The template is circled in blue, and the correct match is shown in red. Notice that most of the trees which are similar in size to the template are correctly detected, while the detector fails for trees which are much larger or smaller.

[9] W. Carter et al. Viewfinder, May 2009. `interactive.usc.edu/viewfinder/index.html`.