Object instance recognition using triplets of feature symbols

MSR-TR-2007-53

C. Lawrence Zitnick[2], Jie Sun[1], Richard Szeliski[2], and Simon Winder[2]

[1] Georgia Institute of Technology
{sun}@cc.gatech.edu
[2] Microsoft Research,
{larryz, szeliski, swinder}@microsoft.com

Abstract. Known object recognition is the task of recognizing specific objects, such as cereal boxes or soda cans. Millions of such objects exist, and finding a computationally feasible method for recognition can be difficult. Ideally, the computational costs should scale with the complexity of the testing image, and not the size of the object database. To accomplish this goal we propose a method for detection and recognition based on triplets of feature descriptors. Each feature is given a label based on a modified K-means clustering algorithm. Object matching is then done by inverse lookup within a table of possible triplets. The ambiguity of the matches is further reduced by having each triplet vote on its proposed object center. For planar objects, the proposed object centers should cluster at a single point. In general, assuming orthographic projection, the proposed centers will lie along a line. If enough triplets are in agreement on a specific object's center, the object is labeled as detected. Our algorithm has been evaluated on a new database with 118 training objects and various testing scenarios.

1 Introduction

Object recognition remains a challenging problem in computer vision. Within object recognition, two main classes of applications exist: object instance recognition and object class/category recognition. Object instance, or known object recognition, is the task of recognizing specific instances of objects, such as cereal boxes, soda cans or action figures [1–3]. Object class recognition deals with categories of objects such as cars, faces and airplanes [4–7]. In this paper we focus on the known object recognition task. More specifically, we attempt to recognize objects within a cluttered scene given several training images of each object.

Feature based methods are one of the most promising approaches to known object recognition [8, 9]. These methods extract local feature descriptors from salient points in the image. Recognition is done by matching the feature descriptors from the query image with those found from a set of training images (figure 2). Within a verification stage, ambiguous matches are eliminated by matching objects using a global affine transformation. One of the difficulties with this approach is matching found features with those in the database. The size of the feature database can be quite large, and scales linearly with the number of known objects. Approximate nearest neighbor (ANN) or hashing is commonly used to reduce the computational complexity of this search.

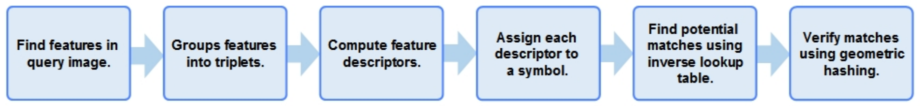
The limitations of this approach become apparent as the number of objects within the database increases. The size of the feature database can become quite large, for example, 40,000 features were used from 32 images in [8]. As the feature space becomes more crowded, it also becomes more difficult to find correct matches, as several good matches might exist for any feature within a query image.

In large feature databases, the ambiguity of the correctly matching feature is most likely unavoidable. Because of this, we assume the feature space will be densely populated, and assign each feature to a cluster [10] instead of finding its single closest match within the database. The set of clusters is created using a modified K-means clustering algorithm during training. The number of possible clusters can range from 1,000 to over 10,000.

This set of cluster means creates a vocabulary of features. The resulting symbols can be quite generic and are rarely object dependent. To resolve the ambiguity in matching symbols, we match feature triplets (groups of three symbols). For each triplet, we store the objects within which they exist using an inverse lookup table. This allows us to efficiently find all potentially matching objects.

Since the same triplet might appear in multiple objects, we add a final verification stage using geometric hashing [11]. For each pair of potential matching triplets, we use the positions of the three features to compute an affine transformation between the query and training images. The object center is then projected onto the query image. If enough triplets agree on the position of the object center, the object is labeled as detected.

One of the difficulties with the vocabulary of features approach is ensuring that corresponding features across images are assigned to the same symbol. If


Find features in query image.
Groups features into triplets.
Compute feature descriptors.
Assign each descriptor to a symbol.
Find potential matches using inverse lookup table.
Verify matches using geometric hashing.

(a)

Fig. 1. Outline of proposed algorithm.

the feature appearance varies due to image noise or mis-estimation of position, scale or rotation, differing symbols may be assigned. To increase the reliability of the feature descriptor, the local image appearance is warped using a affine transformation computed from the triplet's feature positions. Thus, a feature location will have a different descriptor for each triplet. As we discuss in section 3, this increases the repeatability of the features descriptors while also increasing the area in feature space in which the descriptors can lie. A final outline of the algorithm is shown in figure 1.

We demonstrate the success of our algorithm on a database of over 100 objects, both planar and non-planar. Using both similar objects and ones that vary greatly, we attempt to simulate the performance of our algorithm on larger data sets.

2 Related work

Recognizing specific instances of objects has been explored using several techniques. One of the earliest methods is that of geometric hashing [11]. Murase and Nayar [12] proposed a method for recognizing 3D objects using a manifold created from object appearance. More recently, approaches that match local image features have been proposed using grey level invariants (Schmid and Mohr [1]), scale and rotation invariant features (Lowe [8, 3]) and affine invariant features (Rothganger et al. [9]).

The use of locally invariant features has been popular for both object instance and category recognition. A performance comparison of the various features can be found in [13]. Of these, two of the most commonly used are SIFT [2] and variants of the Harris corner detector [14, 15].

3 Feature triplets

In previous known object recognition algorithms [8, 3, 9], each object is represented as a set of training images such as those shown in figure 2. The images in turn are then represented as a bag of local feature descriptors, $f_i \in \{f_1, \ldots, f_n\}$. Each descriptor is based on local image statistics, such as edge magnitudes. The accuracy of these methods largely depends of the distinctiveness of each separate feature [13]. As a result, large databases of features are needed; for example 40,000 features were used from 32 images in [8].

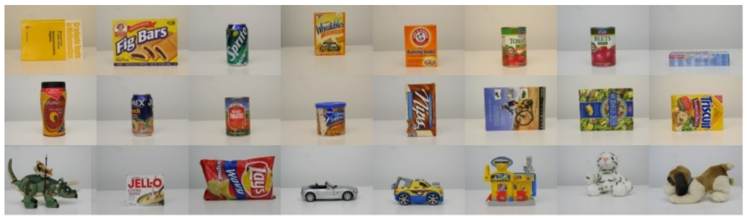


Fig. 2. Example training images.

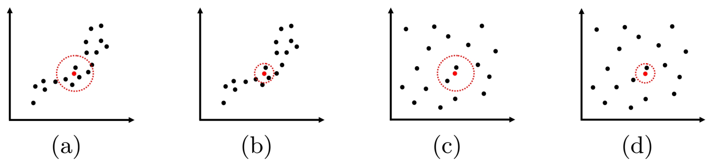


Fig. 3. Illustration of features as black dots in feature space. The query feature is shown as a red dot with the extent of the feature variance shown as the dotted red circle. (a) Standard features and variance, (b) reducing variance, (c) increasing the area that features can occupy, and (d) both.

As the feature database becomes more crowded, many erroneous matches might exist within a small neighborhood of the correct match. Let us define the variance of the correct corresponding feature descriptors across images as the feature variance, σ_F. The feature variance depends on the amount of image noise and the reliability of the computed position, scale and rotation of the features. As illustrated in figure 3(a), multiple features may lie within the feature variance (shown by the dashed red circle) of the query feature (shown as a red dot), resulting in ambiguous matches.

There are two approaches to reducing the chances of a false match. First, the repeatability of the features can be increased, reducing the feature variance (figure 3(b)). Second, we can increase the space in which features exist, so that fewer features lie close to the correct match (figure 3(c)). Standard rotation and scale invariant methods attempt to find locally similar image patches. As a result, the feature descriptors are more similar, as shown in figure 4(a), and only a fraction of the entire feature space will be occupied, figures 3(a) and 3(b).

To address this problem, we propose a method that improves the repeatability of the features while also increasing the space in which they exist (figure 3(d)). The method relies on a novel technique for finding feature footprints. The footprint of a feature is the area, usually a rectangular region, of the image used to compute the feature descriptor. Typically, the footprint is computed from the feature's position, scale and rotation. Instead, we compute the footprint using groups of three features, or triplets. Using the new footprints, standard descriptors, such as SIFT, are then used. The set of triplets, T, consist of all groups of three features that exist within a certain distance of each other in pixel and

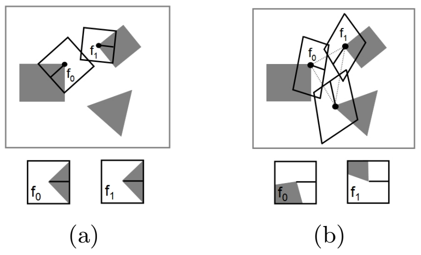


Fig. 4. Effects of invariant approaches: (a) Standard approaches using rotation and scale invariance attempt to find locally similar patterns, resulting in similar features that only occupy a subset of the feature space. (b) Approaches using triplets return features based on neighboring feature positions. As a result, the features occupy a larger set of the feature space.

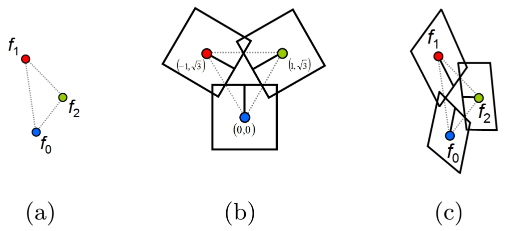


Fig. 5. Finding feature footprints using feature triplets; (a) original triplet, (b) triplet warped to canonical frame with feature footprints , (c) feature footprints warped back to original frame.

scale space. The details of creating triplets can be found in section 6. Resulting triplets, in which the feature positions form an overly narrow triangle, are discarded. More specifically, triplets in which the ratio of the longest distance between features to the sum of the two shortest distances is greater than a threshold $\tau = 0.75$ are removed.

We use the feature positions, which are typically computed more accurately than the scale and orientation, to compute an affine transformation A_i from image space into a canonical frame. Within the canonical frame, the three points are warped to positions $(0,0), (-1, \sqrt{3}), (1, \sqrt{3})$. If the position of feature $f_{i,j} \in t_i$ is $p_{i,j} = (x_{i,j}, y_{i,j})$ for $j \in \{0, 1, 2\}$ we have:

$$A_i \begin{bmatrix} x_{i,0} \\ y_{i,0} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, A_i \begin{bmatrix} x_{i,1} \\ y_{i,1} \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ \sqrt{3} \\ 1 \end{bmatrix}, A_i \begin{bmatrix} x_{i,2} \\ y_{i,2} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ \sqrt{3} \\ 1 \end{bmatrix} \quad (1)$$

These specific points are chosen for the canonical frame so that they form an equilateral triangle and feature $f_{i,0}$ is mapped to (0,0), figure 5(b). We can compute A_i by:

$$A_i = \begin{bmatrix} 0 & -1 & 1 \\ 0 & \sqrt{3} & \sqrt{3} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_{i,0} & x_{i,1} & x_{i,2} \\ y_{i,0} & y_{i,1} & y_{i,2} \\ 1 & 1 & 1 \end{bmatrix}^{-1} \tag{2}$$

Within this warped frame, shown in figure 5(b), the footprints of the three descriptors are placed at the corners of the equilateral triangle with their orientations pointing towards the triangle's center. Within the canonical frame, the size of a footprint's side is 2, the same as the distance between feature locations.

Using A_i, the feature descriptors within the triplet are invariant to affine transformations between images. As a result, the feature variance of correctly corresponding descriptors across images should decrease over simpler scale and rotation models. Affine invariant features have also been proposed for Harris corner detectors [15] using image gradients. However, if there is not enough local gradient information, the computed affine transformations may not be robust. A similar approach to ours, which uses neighboring feature positions for finding affine invariant features, was proposed for panoramic stitching in [16].

Even though the triplet features are invariant to more general affine transformations, and not just scale and rotation, the space which the descriptors can occupy is larger. This is due to the affine transformations being computed using neighboring feature positions and not local image properties. For example, in figure 4(a), both features will have the same descriptor if their scales and rotations are computed from local image gradients. With the triplet approach, figure 4(b), the descriptors will vary based on the location of the neighboring features. As a result, the possible area in feature space that can be occupied by the descriptors is larger, figure 3(d).

4 Matching feature triplets

In the previous section, we described how we create triplets and use their feature positions to find affine invariant descriptor footprints. The final result is a set of three SIFT feature descriptors for each triplet. This section, we describe a method for efficiently matching triplets across images.

The number of triplets is typically much larger than the number of features. For an average image of 300 features, there exists $\binom{300}{3}/3 = 1,485,033$ possible triplets. Since we limit the set of possible triplets based on the relative positions and scales of features, the number of triplets is reduced to about 10,000. Given a large training set, a database of descriptors would be quite crowded. If we attempt to find correct corresponding features, mismatches are inevitable. This will be the case even though the space in which the triplet features can exist is larger.

Instead of storing all feature descriptors within the training data set, we group features of similar appearance into clusters. The cluster centers $S = s_1, \ldots, s_N$ form a vocabulary of symbols over the set of descriptors. Each descriptor $f_{i,j}$ in a query image is then mapped $m(f_{i,j}) = s_k$ to the closet symbol. Clustering of features for generalization can be useful for object categorization [7, 10]. However, the quantization of the feature space reduces the distinctiveness of features, and as a result, its usefulness for object instance recognition.

While the descriptiveness of each individual feature is reduced, each triplet, t_i, will have three symbols, $s_{i,0}, s_{i,1}, s_{i,2}$ associated with it. Given the large number of possible triplets, $\approx 1,000^3$, the uniqueness of each triplet is increased over single features. As a result, the number of training images that contain a specific triplet will be much smaller than those that contain any single feature. Using an inverse look-up table that we describe below, we can efficiently find all training images that contain the same triplet. Before we do this, we first describe our method for clustering feature descriptors.

4.1 Clustering feature descriptors

There are two goals to clustering feature descriptors. First, we want as many feature clusters as possible. The sparseness and efficiency of our inverse lookup table increases as the number of symbols or clusters increases. Second, given two corresponding features across images, we want them both to be assigned to the same symbol or cluster. These two goals must be balanced, since the more clusters that exist, the more likely corresponding features will be assigned to different clusters.

Ideally, the feature descriptors would group together into easily identified clusters. Unfortunately, feature descriptors are more evenly distributed as shown in figure 3(d). Thus, finding appropriate cluster boundaries is difficult.

Given the relatively uniform distribution of feature descriptors, the spacing between clusters is as important as their exact position. To minimize the chances of corresponding features being assigned to different symbols, we enforce that the variance of feature positions within a cluster be greater than some multiple of the variance between corresponding features across images. Thus, we modify the standard K-means algorithm that iteratively groups descriptors into clusters, by enforcing a certain separation between clusters. After each iteration, any clusters whose centers are less than $\beta\sigma_F$ apart, are merged together. To maintain a constant number of clusters, any merged cluster is assigned a new center chosen randomly from the descriptor set. For our experiments, we set $\beta\sigma_F = 0.03$. Figure 6(b) shows the cluster centers for the 30 largest clusters. As seen in figure 6(c), the features are well distributed among the clusters.

4.2 Inverse look-up table

Given a triplet t_i with symbols $s_{i,0}, s_{i,1}, s_{i,2}$, we would like to efficiently find all training images with similar triplets. To accomplish this, we use an inverse look-up table. For each possible combination of three symbols, we store the index of every training image that contains a triplet with the same symbols. Thus, if we have N symbols, the lookup table will have N^3 entries. Given a large size of N, over 1000, most entries within the lookup table will be empty. A sparse matrix representation is used for the table to reduce memory requirements. To

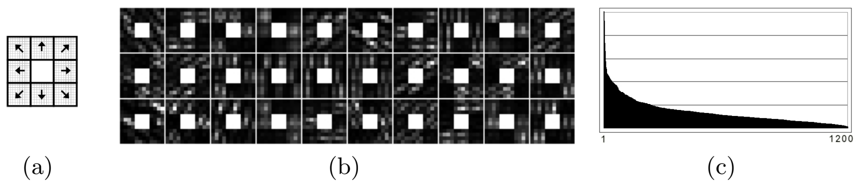


Fig. 6. Feature properties: The SIFT features are displayed by grouping the 16 positions in eight orientations as shown in (a). (b) Top 30 feature cluster means. (c) The frequency histogram of 1,200 feature symbols within the training set.

ensure all rotationally symmetric triplets are grouped together, the symbols are reordered in ascending order, $s_{i,0} < s_{i,1} < s_{i,2}$, before lookup. If more than one feature is assigned the same symbol, the mapping between triplet features is ambiguous. Since a unique mapping is needed in the verification stage of our algorithm, described in the next section, these triplets are discarded.

One possible method for object detection is to store the probability of each triplet t_i, given the object O_j, $P(t_i \mid O_j)$. Assuming the triplets are independent, we could then compute the likelihood of each object as:

$$P(O_j \mid T) \propto P(O_j) \prod_{t_i \in T} \frac{P(t_i \mid O_j)}{P(t_i)} \tag{3}$$

where $P(t_i \mid O_j)$ is equal to some small value ϵ when the triplet hasn't been observed for the object.

If we detected objects based on the thresholded values of (3), our precision and recall results would be quite poor. While the use of triplets increases the descriptiveness over the use of single symbols, this "bag of words" approach will result in many ambiguous matches. For this reason, we do not use this approach. Instead, as we describe in the next section, we use relative feature positions as well as feature symbols to find reliable matches.

5 Object verification

Since the symbols within a triplet are not unique to a specific object, additional constraints are needed to guarantee a correct match. One possibility is to use the spatial relationship between triplets. As in [8], we could assume a global affine model for our objects. We could then find sets of triplets that vote for the same object and have similar affine transformations using a Hough transform or RANSAC. This approach works well for objects with a single dominant plane, such as cereal boxes or books. However, the global affine model is violated for non-planar objects or when there is significant projective distortion.

Instead of assuming a single global affine model, we propose a geometric hashing [11] approach that computes an affine transformation between each pair of matching triplets. Given a reference point, p_R within a training image I_j

(usually assumed to be the object center,) we project the point into the query image using the computed affine transformation (figure 7). If an object is present, the set of projected points from each matching triplet pair will obey certain relationships, as we discuss in the following sections.

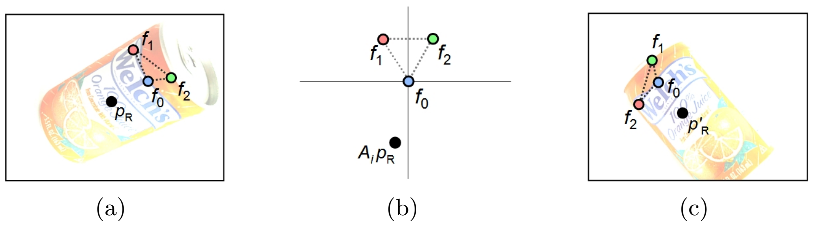


Fig. 7. Projecting reference points: (a) triplet with reference point p_R in training image, (b) reference point projected to canonical frame $A_i p_R$ and (c) reference point $p'_R = A_i'^{-1} A_i p_R$ projected into the query image.

5.1 Projecting reference points

As we discussed in section 3, we can compute an affine transformation, A_i that projects our triplet t_i into a canonical frame. Using the same affine transformation, we can project a reference point p_R into the same canonical frame (figure 7(b)). Given a matching triplet t'_i in another image with corresponding affine transformation A'_i, we can project the reference point p_R from one image into another, as shown in figure 7, using:

$$p'_R = A_i'^{-1} A_i p_R \tag{4}$$

The reference point p_R is usually assumed to be the object center in training image I_j. If the object is roughly centered in the image, the image center may be used.

To verify the detection of objects in a query image, we need to project the reference point using each triplet. Instead of storing the entire matrix A_i in our inverse lookup table for each triplet in the training data set, we can just store the location of the reference point in the canonical frame, $A_i p_R$.

5.2 Planar objects

For a planar object with an orthographic plus scale camera model, the transformation from a training image to an observed image can be modeled using a global affine transformation. In many cases, the more general projective camera model can also be well approximated locally using this affine model.

Assuming our object can be modeled using a global affine transformation A^*, all correctly matching triplets have an affine transformation such that $A_i'^{-1} A_i =$

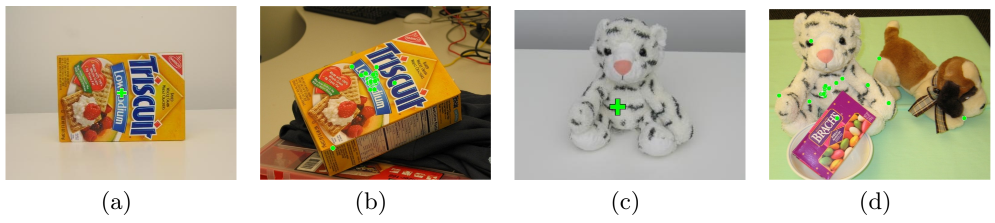


Fig. 8. Example of projected reference points. The reference points are shown as a green crosses in the training images (a) and (c). The projected reference points are shown as green dots in the query images (b) and (d).

A^*. As a result, all of the projected reference points p'_R lie at a single point (figure 8).

Object detection for planar objects can then be accomplished using a simple 2D histogram binning technique called geometric hashing [11]. An entry in the histogram is created for each 8x8 block of pixels within the query image. Every projected reference point is then added to the closest 2D bin and its 8 connected neighbors. The quality of the match is measured by the number of points within the largest bin of the histogram. Even if many incorrectly matched triplets exist, it is unlikely they will vote on the same bin. Thus, we can easily eliminate most false matches.

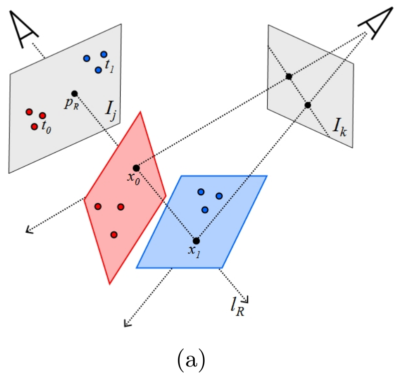


Fig. 9. The projections of the reference point p_R lie along a line in the query image I_k.

5.3 Non-planar objects

Unlike planar objects, the projection of a non-planar object cannot be modeled using a single affine transformation. To understand how this affects the projection of the reference points, let us re-examine how this projection is accomplished.

Each feature f_k within a triplet t_i has a corresponding unknown 3D location x_k on the object. Similarly, the three features within a triplet t_i correspond to some 3D plane, q_i (figure 9). Since we project the reference point p_R relative to the triplet's positions, we are assuming the reference point also lies on this unknown plane q_i. The hypothesized 3D position x_i of the reference point p_R is then determined by the intersection of the plane q_i with the line l_R projecting from the camera center of the training image I_j through the true 3D point x^* of the reference point. The position of the projected reference point p'_R is then just the projection of the 3D point x_i onto the query image (figure 9).

For planar objects, the 3D planes q_i corresponding to each triplet are the same. As a result, the hypothesized 3D locations x_i of the reference points, as well as their projections into the query image, are the same. For non-planar objects, the planes q_i will vary between triplets. Thus, the hypothesized locations x_i of the reference points will lie at different points along the line l_R and project to different locations within the query image. However, since the points x_i all lie along a single 3D line l_R, their projections onto the query image will also lie on a line (commonly referred to as the epipolar line for p_R).

Assuming an orthographic plus scale model, the object centers should lie exactly on a line. However, projective distortions and errors in computing the feature centers will introduce noise. Several methods are available for robustly finding 2D lines in point clouds, such as RANSAC, Hough transform, etc.

Despite the generality of the line fitting approach, we have found experimentally that it is not necessary. There are two reasons for this: First, the spread of the points on the line is related to the difference in rotation of the object in the training and query image (similar to the disparity range in stereo vision). If the training images are taken close together (less than 25 degrees apart), then a simple point cluster detector as proposed in section 5.2 is adequate. Second, the position of the feature detectors become increasingly unreliable as the rotation between images increase, reducing our ability to match highly separated views.

5.4 Final verification

Each feature in an image should only correspond to a single object. If we do not impose a one-to-one mapping constraint between features and training images, it is possible for the same feature in a query image to be matched to different training images. We enforce this constraint by assigning features to training images using a greedy approach. We first find the training image with the highest number of matching triplets and assign all of the features in those triplets to that training image. To prevent the features from matching to multiple training images, we remove *all* of the triplets from the query image that use these features. We then find the best matching training image using the remaining triplets and iteratively repeat this process until no training image has more than κ matching triplets.

6 Implementation Details

In this section we describe some the implementation details of the algorithm related to finding feature triplets and efficiently matching them across images.

Our features are detected using the Difference of Gaussian (DoG) approach proposed in [3]. Each feature has a pixel location p_k, scale σ_k equal to the standard deviation used to compute the DoG, and rotation θ_k. The triplets are found by grouping all sets of three features that lie within a certain distance in pixel and scale space. In scale space, the ratio of feature scales must lie between 0.5 and 2.0. The distance between feature locations must be less than $\xi\sigma_k$, where $\xi = 8$. To further reduce the number of triplets and to increase the accuracy of the projected reference points, the average distance between features must be greater than 20 pixels. Once the feature footprints are found, we use the SIFT feature descriptor described in [3].

When clustering features detectors, we found 1,200 features with the highest occurrence in the training set and differed in appearance greater than $\beta\sigma_F$. To reduce the computational cost of assigning symbols to features, we first found the difference between the average within the 8 orientations of the SIFT descriptor for the feature and the cluster mean. If the difference was less than a threshold (0.45) a full comparison was done.

Even though the geometric hashing technique using 2D binning proposed in section 5.2 is relatively efficient, we still do not want to create a set of 2D bins for every possible training image. In practice, we first find the set of training images from which a minimum number of reference points (e.g. 20) project within the query image's boundary. Even if several training images contain a certain triplet, as stored in the inverse look-up table, only a fraction of the projected reference points will also lie within the boundary of the image. Using this approach, the number of training images that need to be considered is greatly reduced.

7 Experiments

To test our algorithm we have collected a database of 118 objects. Approximately 100 of the objects are box or can shaped grocery store items. The rest of the objects are toys, such as cars and stuffed animals. Each object has 48 training images sampled at different orientations. Three rows were taken vertically with 25 degree spacing and 16 columns horizontally with 22.5 degree spacing. A sample of the training images is shown in figure 2. To simulate larger datasets, the objects were chosen so that some were very similar in appearance, such as the juice cans, while others varied greatly, such as the toys.

We collected two testing datasets: non-occluded single objects and occluded multiple objects. The non-occluded single object data set contains images of a single object with little or no occlusion. The multiple object data set contains images with up to 6 objects and possible occlusions. The entire database is available at [removed for review purposes].

For each of the two categories, we created an ROC curve with values of κ, the number of matching triplets needed for detection, ranging from 3 to 100 (figure

10). With $\kappa = 15$, we obtain an 78.8% detection rate with 14.4% false positives for the single object database, and an 81.1% detection rate with 8% false positives for the multiple object database. Since each object only occurred at most once in each image, if the same object was detected more than once, it was labeled as a false positive. In general, the algorithm was most successful with objects that are highly textured with some planar surfaces. Some example matches can be found in figure 11. The objects within the database were deliberately chosen to have varying degrees of difficulty for recognition. Some of the most difficult objects are shown in figure 12, these include specular objects, objects with little texture, non-planar objects and deformable objects. Several of the objects were also taken at angles greater than 30 degrees from any training image. It was our hope that this database will remain challenging for comparison against future algorithms.

The computation time for our algorithm is between 5 and 20 seconds per query image of size 800x600, depending on the number of features found. Approximately, 20% of the time is spent on finding triplets and their descriptors, and 80% of the time on matching feature descriptors with their symbols. Once the feature symbols are known, it takes on average 0.3 seconds to search the database of 5664 training images with 14,145,274 triplets for possible matches. Given the efficiency of our search, we believe our algorithm could scale to large databases. Our experiments were run on a Intel Xeon 2.79 GHz machine with 2 GB of memory.

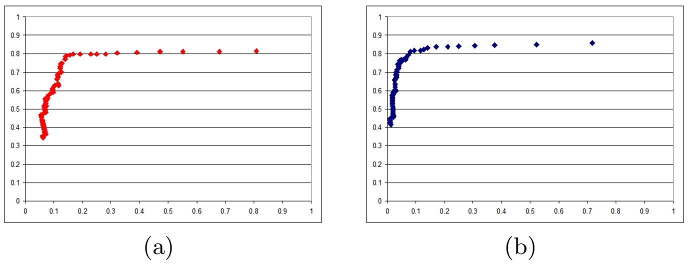


Fig. 10. ROC curves for (a) single object and (b) multiple object database.

8 Conclusions and discussions

Billions of unique individual objects occupy our world. Recognizing these objects, in an accurate and efficient manner is a difficult, but important task for many applications.

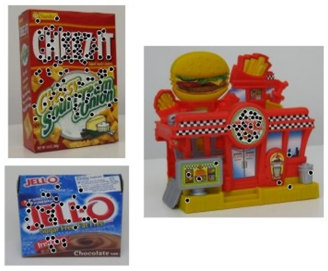
Within this paper, we describe a method for object instance recognition using triplets of features. The use of feature triplets serves two purposes. First, the space of possible feature descriptors and their repeatability are increased.

Second, by assigning features to symbols and using an inverse lookup table, possible matches can be efficiently found. The computational cost of finding features is increased over standard single feature methods [8], but the cost is independent of the training database size. Once the feature symbols are found, matching training images to the query image is very efficient, and should scale to large databases of objects.

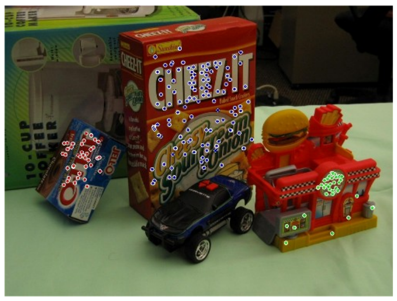
The algorithm has shown good recognition results on non-deformable objects with some planar surfaces. The quality of the results is limited by the ability of the feature detector to find repeatable features in complex non-planar objects, or objects with little texture. To handle these more difficult cases, other types of feature detectors and descriptors need to be used and developed.

References

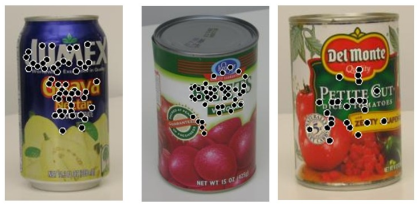
1. Schmid, C., Mohr, R.: Local gray value invariants for image retrieval. IEEE Trans. on Pattern Recognition and Machine Intelligence **19** (1997) 530–534
2. Lowe, D.G.: Object recognition from local scale-invariant features. In: IEEE Proc. of Int'l Conf. on Computer Vision. (1999) 1150–1157
3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int'l J. of Computer Vision **60** (2004) 91–110
4. Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: IEEE Proc. of European Conf. on Computer Vision. (2000) 18–32
5. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proc. CVPR. (2001) 511–518
6. Fergus, R., Perona, P., Zisserman, A.: Object class recogntion by unsupervised scale-invariant learning. In: IEEE Proc. of Computer Vision and Pattern Recognition. (2003) 264–271
7. Dorko, G., Schimid, C.: Object class recognition using discriminative local features. IEEE Trans. on Pattern Recognition and Machine Intelligence (2004)
8. Lowe, D.G.: Local feature view clustering for 3D object recognition. In: IEEE Proc. of Computer Vision and Pattern Recognition. (2001) 682–688
9. Rothganger, F., Lazebnik, S., Schmid, C., Ponce, J.: Object modeling and recognition using local affine-invariant image descriptors and multi-view spatial contraints. Int'l J. of Computer Vision (2004)
10. Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering objects and their location in images. In: IEEE Proc. of Int'l Conf. on Computer Vision. (2005)
11. Lamdan, Y., Wolfson, H.J.: Geometric hashing: A general and efficient model-based recognition scheme. In: IEEE Proc. of ICCV. (1988) 218–249
12. Murase, H., Nayar, S.: Visual learning and recognition of 3-d objects from appearance. Int'l J. of Computer Vision **14** (1995) 5–24
13. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. on Pattern Recognition and Machine Intelligence (1997)
14. Harris, C., Stephens, M.: A combined edge and corner detector. In: Alvey Vision Conference. (1998) 189–192
15. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: IEEE Proc. of European Conf. on Computer Vision. (2002) 128–142
16. Brown, M., Lowe, D.G.: Invariant features from interest point groups. In: British Machine and Vision COnference. (2002) 656–665



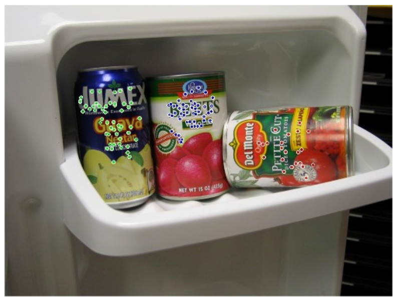
(a)



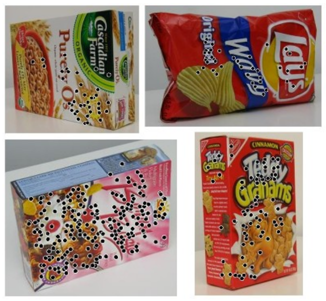
(b)



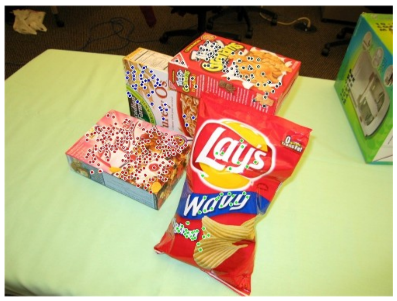
(c)



(d)



(e)



(f)

Fig. 11. Example matching results with $\kappa = 15$: The training images (a), (c) and (e) with corresponding feature locations. The query images (b), (d) and (f) with found matches.

(a)

(b)

(c)

(d)

Fig. 12. Difficult types of objects for recognition: (a) specular objects, (b) non-textured objects, (c) non-planar objects and (d) deformable objects.