Detection of Multiple Deformable Objects using PCA-SIFT

Stefan Zickler and **Alexei Efros**
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, U.S.A.

Abstract

In this paper, we address the problem of identifying and localizing multiple instances of highly deformable objects in real-time video data. We present an approach which uses PCA-SIFT (Scale Invariant Feature Transform) in combination with a clustered voting scheme to achieve detection and localization of multiple objects while providing robustness against rapid shape deformation, partial occlusion, and perspective changes. We test our approach in two highly deformable robot domains and evaluate its performance using ROC (Receiver Operating Characteristic) statistics.

Introduction

SIFT (Scale Invariant Feature Transform) has been shown to be an effective descriptor for traditional object recognition applications in static images (Lowe 1999). In this paper, we propose and evaluate a method that uses PCA-SIFT (Ke & Sukthankar 2004) in combination with a clustered voting scheme to achieve detection and localization of multiple, highly deformable objects in real-time video footage.

The detection and localization of actively deforming objects, such as walking or dancing humanoid robots, is a significantly more challenging task than the well researched problem of detecting fixed shape objects in static images. One major reason for this is that highly deformable objects cannot be easily represented by a single global feature descriptor because an object's overall shape configuration can change fundamentally between observations. The same problem arises for approaches which focus on representing an object by the precise geometric relationship of its smaller local features, such as Hough Transforms used in combination with SIFT feature descriptors (Lowe 1999). Our presented approach aims to overcome this problem by representing an object using purely local features without enforcing a strict geometric relationship between them. Instead, we introduce a probabilistic voting scheme where each detected local feature will produce its own hypothesis of the object's most likely location. This proposed algorithm attempts to provide robustness against object deformation, object motion, and perspective changes.

This paper is organized as follows. We first introduce some of the related work in this area and provide a short review of the SIFT feature descriptor. The body of this paper explains the algorithm in detail, followed by presentation and analysis of experimental results from our sample domains. Concluding remarks and suggestions for future work are presented in the final section.

Related Work

It has been shown that SIFT descriptors can be used to achieve fairly robust object detection in still images (Lowe 1999). SIFT has furthermore been used in several other related applications such as metric robot localization (Se, Lowe, & Little 2001) and medical imaging (Moradi, Abolmaesoumi, & Mousavi 2006); and it was shown to be one of the best currently available descriptors in a comparative study (Mikolajczyk & Schmid 2003).

Various approaches exist on real-time recognition of deformable objects. A common one is silhouette matching using Distance Transforms (e.g. Chamfer Distance) which has been used for pedestrian recognition in smart vehicles (Gavrila & Philomin 1999). This approach contains several inherent problems, the most significant one being the fact that objects are only defined by their contours and not by any of their inner features. This not only increases the likelihood of finding false positives due to contour ambiguity, but it also results in combinatorial explosions when attempting to match all possible contours of highly complex and deformable objects. Also, scale and rotation are not inherently invariant and therefore need to be treated as additional deformation factors. While some of the issues such as contour ambiguity and occlusion handling can be improved using Oriented Edges and Hausdorff metrics (Olson & Huttenlocher 1997), most of the underlying problems of countour-matching do remain. We are aware of a hybrid model which attempts to combine this kind of counter-matching approach with the use of inner object features in order to achieve pedestrian recognition (Leibe, Seemann, & Schiele 2005). While results look promising, this approach still struggles with some of the inherent downsides mentioned above and is to our knowledge not yet able to perform in real-time.

Active Appearance Models (Cootes, Edwards, & Taylor 2001) are another recent advancement in the vision community. Unfortunately, these models require highly specific statistical models which need to be manually adapted to work for a particular object class (such as human faces). Active Appearance Models do not generalize well for random classes, and are furthermore unlikely to perform well for highly deformable classes where no simple statistical model exists.

Review of the SIFT feature descriptor

While it is beyond the scope of this paper to describe the SIFT algorithm in its entirety, we will quickly review its most significant properties and describe why it is suitable for our purpose. SIFT features are generated by finding interesting local keypoints in an image. A very common and efficient way of generating these keypoints is by locating the maxima and minima of Difference-of-Gaussians (DoG) in the image's scale-space pyramid. This is done by calculating different levels (octaves) of Gaussian blur on the input image and then computing the difference of neighboring octaves. A canonical orientation vector can then be computed for each keypoint, thus giving a complete keypoint coordinate of X, Y, scale, and orientation. A SIFT feature is a 128-dimensional vector, which is calculated by combining the orientation histograms of locations closely surrounding the keypoint in scale-space. The advantage of SIFT keypoints is that they are invariant to scale and rotation, and relatively robust to perspective changes (experiments have shown that SIFT is typically stable up to a perspective change of approximately 10-20 degrees). This makes it an ideal descriptor for object recognition tasks.

One common problem of SIFT is its relatively high dimensionality, which makes it less suitable for nearest neighbor lookups against a training dataset. PCA-SIFT is an extension to SIFT which aims to reduce SIFT's high dimensionality by applying principal component analysis (PCA). 20-dimensional PCA-SIFT was used as the feature descriptor in this paper.

Description of the Approach

In this section we will first give a concise algorithmic summary of our approach, and then follow up with a detailed description and discussion of each significant step. A simplified flowchart of the approach can be seen in figure 1. Our algorithm consists of two major components: the training stage in which we "learn" the representation of an object by collecting its PCA-SIFT features; and, the recognition stage in which we attempt to detect and localize the object in real-time video footage. The training stage of our algorithm is as follows:

1. We record a continuous training video V of the object.
2. For each frame v_i of our training video V:
   (a) We generate the set of all PCA-SIFT keypoints K_i containing keypoints $k_{ij} \in K_i$.
   (b) We manually create an annotation mask m_i approximating the boundary and center c_i of the training object.
   (c) We reject any keypoints from K_i which are lying outside of the annotation mask m_i.
   (d) We store the relative location $locrel_{ij}$ of each keypoint towards the annotated object's center c_i such that $locrel_{ij} = c_i - \mathrm{loc}(k_{ij})$.

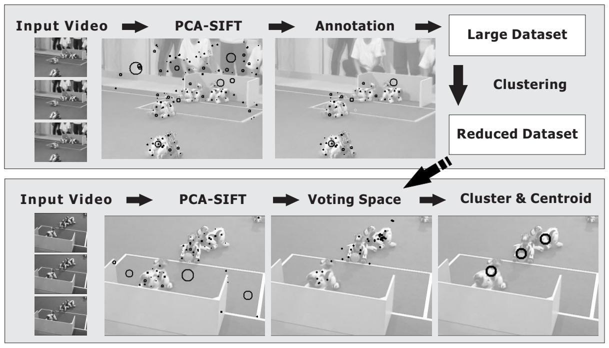


Figure 1: A simplified flowchart of the presented approach

3. We combine all retained keypoints k_{ij} into a single set T.
4. We further reduce the size of T using agglomerative clustering. This yields our final training dataset.

The detection and localization stage of our algorithm is as follows:
For each frame v_i of our incoming continuous video V:

1. We generate the set of all PCA-SIFT keypoints K_i containing keypoints $k_{ij} \in K_i$.
2. For each keypoint k_{ij} of K_i:
   (a) We perform a nearest neighbor lookup with all the elements t_l of the training set T. The main detection threshold θ is defined as the Euclidean distance in PCA-SIFT space between a keypoint k_{ij} and its nearest neighbor t_l from the training dataset. An observed PCA-SIFT feature k_{ij} is considered a match if
   $$\min_{t_l \in T} \mathrm{D}(k_{ij}, t_l) \leq \theta$$
   where D is the Euclidean distance function.
   (b) For any matching k_{ij} we calculate the hypothesized position of the object's center. We do so by scaling and rotating the previously recorded relative position $locrel_l$ of t_l to match the scale and orientation of k_{ij}. More specifically we calculate the hypothesized position p_l using the following equation:
   $$p_l = \mathrm{pos}(t_l) + \left(\frac{\mathrm{scale}(k_{ij})}{\mathrm{scale}(t_l)} \times \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \times locrel_l \right)$$
   where $\beta = \mathrm{orientation}(k_{ij}) - \mathrm{orientation}(t_l)$.
3. We run a clustering algorithm on the voting space using a clustering threshold δ and enforcing a minimum cluster size of s votes. We then calculate the center of mass for each cluster. This is our final localization result.

Training

We start by recording a training video of the object, spanning a continuous spectrum of perspectives. Each incoming video frame v_i is treated as an independent observation of which we obtain a set of feature descriptors K_i. PCA-SIFT was chosen as the descriptor of choice, because it has a significantly reduced dimensionality compared to standard

SIFT while keeping a similar level of overall robustness. We adapted the PCA-SIFT implementation used by (Ke & Sukthankar 2004). Difference-of-Gaussians (DoG) is used as the basic interest point detector which is invariant to scale and rotation changes. The PCA-SIFT descriptors used in this paper are 20-dimensional. A typical frame in our testing environment generated somewhere between 50 and 300 keypoints.

Annotation

In order to build a training subset out of all the generated PCA-SIFT features, we first need to determine which ones belong to the training object and which ones do not. For this purpose, a graphical annotation tool was developed which allows the user to approximate each object's shape by drawing one or more geometric primitives (such as circles). This approach greatly simplifies the annotation task as it is possible to only annotate certain keyframes and then interpolate the movement and size of these geometric primitives for all frames in between two keys. The center c_i of each object is inherently annotated by calculating the center of mass of the geometric primitives used to approximate the object's shape. The relative location $locrel_{ij}$ of each PCA-SIFT feature toward the center of the object c_i is furthermore computed as previously described. All PCA-SIFT features lying outside of any annotated region are rejected from the training set.

Post-Processing

Even after annotation-based filtering we still have a very large dataset of relevant SIFT features. Considering that this data come from a continuous video sequence, it can be assumed that many of the collected features are highly similar or even identical. This is partly the case because a particular feature might only change slightly (or not even at all) between two video frames. Another reason for redundancy in the dataset is that a single object might contain several similar local features at different locations.

Since we are expecting to perform lookup on this dataset during recognition, it is desirable to reduce its size as much as possible without throwing away too much information. For this purpose, we run an agglomerative clustering algorithm (Gowda & Krishna 1978) which merges similar features in the dataset. When merging, we use the Mean to compute the location of the newly merged feature. Unlike k-Means, agglomerative clustering allows us to directly define a distance threshold in 20-dimensional PCA-SIFT space to determine whether two features should be merged. Euclidean distance is used as the distance metric.

Recognition

The goal of the recognition stage is to detect and to localize all instances of the trained object in an incoming video stream. We again start out by generating all PCA-SIFT keypoints for each observation received from the video stream. We then perform a nearest neighbor lookup for each of these features against every feature from the reduced training set T. In the current implementation, this is done by a simple linear nearest-neighbor search, mainly for simplicity reasons. Any other more sophisticated data-structure suited for higher dimensional data (such as KD-trees) could be used equally well if additional performance is required. The main detection threshold θ is defined as the Euclidean distance in PCA-SIFT space between a feature and its nearest neighbor from the training dataset as previously described in the formal algorithm definition. Choosing a good value of θ is critical. A smaller value of θ will deliver fewer, but more precise matches while a larger value of θ will deliver more matches while increasing the likelihood of false positives.

Centroid Voting Space

A voting scheme is used to determine the location of each object in the image. Any matched feature k_{ij} votes for where it predicts the center of its parental object to be. This is achieved by retrieving the nearest neighbor t_l and its relative location towards the annotated object's center from the training set T. We then rotate and scale t_l's relative location vector to match the scale and orientation of the newly detected keypoint k_{ij}. When adding this vector to the absolute location of k_{ij}, we get a hypothesis of the object's center. This point is counted as a vote for the object's center in the two-dimensional voting space.

It should be noted that this is a heuristic approximation. There is no clear guarantee that the same feature cannot occur at a different relative location towards the object's center than in the training data. This is especially the case because the object is deformable. However, we can optimistically assume that many of the votes do in fact approximately match the object's center and that our voting scheme will take care of potential outliers.

An interesting side-effect of this kind of voting space approach is that it automatically solves the problem of occlusion. Even when an object is half occluded or moving off the visible screen, we are still able to correctly hypothesize its center. This would not be the case if we were to use a simple center of mass calculation over all matching features' absolute locations.

Clustering on the Voting Space

After populating the voting space, we attempt to localize its peaks by using a clustering algorithm. In our implementation we have experimented with both Mean-Shift (Cheng 1995) and agglomerative clustering. Both of them performed virtually identical in terms of quality. While Mean-Shift should be considered slightly less predictable due to its random initialization, the two algorithms did not produce any noticeable difference in detection results (less than 1% difference in the recognition rate). One significant difference however is speed: Mean-Shift runs significantly faster than agglomerative clustering when there is a large count of votes.

Both Mean-Shift and agglomerative clustering require the definition of a distance-threshold δ (this is known as the 'bandwidth' in Mean-Shift). The meaning of δ becomes clear when we take a simplified look at how agglomerative clustering works. Each point starts out as its own cluster. If the Euclidean distance between two clusters is smaller than δ then the two clusters are merged into one. This process is repeated until no more merges are possible. As our experiments will show, the clustering threshold δ is a crucial variable to gain decent detection results. Choosing a smaller value of δ makes the clustering of votes less likely, and can lead to unnecessary multiple detections of the same object. Choosing a larger value of δ increases the likelihood of clustering votes and increases the risk of wrongly grouping multiple object instances into a single detection point.

After completion of the clustering process we reject all clusters that contain less votes than a certain minimum cluster size threshold s. The center of mass is then computed for each remaining cluster. This becomes our final detection result.

Temporal Voting Space

So far we have only looked at single, independent video frames. One way to make use of the fact that the testing data are a continuous video-stream is to extend the previously described voting space into three dimensions, namely X, Y, and time. This can be achieved by combining the voting spaces of the last N frames with the third dimension representing the age of each frame multiplied by a scalar constant λ. We can then run the standard clustering mechanisms discussed in the previous section in this three-dimensional space.

The reason why one would consider this kind of temporal extension is to filter out noise in the video. Choosing a large N results in a stronger temporal filtering effect, which may reduce the rate of false positives, but may also decrease the object detection rate. Choosing a small N may increase the detection rate, but might concurrently increase the amount of false positives due to noise. Another problem of choosing a large value of N is that we create an effect of temporal lag. Especially when an object moves quickly along the image-plane, the centroid calculation will determine an average which will cover the past N frames, therefore misrepresenting the real current location of the object in the latest video frame.

Experimental Results

We test our approach in two different domains. The first domain originates from a set of randomly chosen RoboCup (Kitano *et al.* 1997) legged league games. The de facto standard robot used in this league is the SONY AIBO. It is a highly dynamic and deformable robot, featuring 20 degrees of freedom that allow almost any thinkable actuation of legs, feet, neck, and head. Participating RoboCup teams typically create their own unique robotic motions which range from basic walking patterns to rather complex and unorthodox bodily expressions. Video scenes are normally populated with several robots which can be depicted in various shape configurations, perspectives, and scales. Other frequent vision constraints are robot occlusion, highly cluttered backgrounds (for instance by a human audience), and motion blurring.

The second domain consists of two dancing humanoid robots in a laboratory environment. The robots used in this domain are SONY QRIOs, again featuring a great amount of deformability. The input used in both domains was in 8 bit greyscale at a 320x240 pixel resolution, captured at 30fps.

Training Data

Continuous video footage is used in order to train for a new object. Since PCA-SIFT is robust against perspective changes up to approximately 10-20 degrees, it is recommendable that the training video contains views from various angles, preferably the ones which we might expect to encounter during testing. For the AIBO domain, three continuous videos were chosen as training data. Each of them contained between 300 and 1000 frames. There was no particular selection criteria for these videos, except that every one of them depicted several robots in various actions and angles that subjectively appeared to present a typical RoboCup game. For the QRIO domain, two 2000 frame training videos were used, each featuring a single robot, performing a dance in an uncluttered environment.

Testing Data

For the AIBO domain, special care was taken that the testing videos not only originated from different RoboCup games than the training data, but also that at least one of the two teams in these games did not occur in any of the training footage. It was furthermore ensured that the testing video was taped from a different perspective than all of the training videos. Both of these constraints were added to increase the demand of robustness from our approach. For the QRIO domain, a testing video with two dancing robots was created. The dance used in this video was significantly different from the one used in the training stage, containing many untrained shape configurations. This video furthermore covered various continuous perspective changes as well as robot occlusions.

Evaluation Criteria

We count a match if we correctly detect and localize the object's center in the image. The requirement is that the detected center lies close enough to the object's annotated center. We chose this threshold to be 50% of the object's annotated bounding box radius. Anything located outside of the circle with this radius will be counted as a false positive.

ROC curves (Receiver Operating Characteristic) are used as the main performance evaluator. Each curve shows a plot of the detection rate vs. the rate of false positives. A curve is generated by iterating through different detection thresholds θ. We compare the influences of different variables to our approach by comparing their ROC curves respectively.

The detection rate is defined as

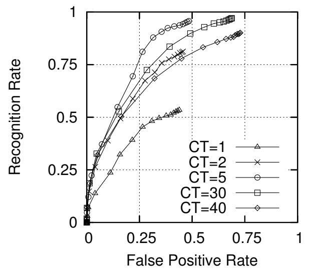
$$P_{\text{detection}} = \frac{\text{\# of correct detections}}{\text{\# of annotated objects}}.$$

It is 1.0 if no objects are annotated in the image.

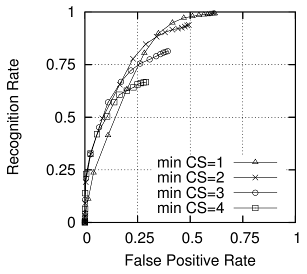
The false positive rate is defined as

$$P_{\text{false positives}} = \frac{\text{\# of incorrect detections}}{\text{\# of detections (both correct and incorrect)}}.$$

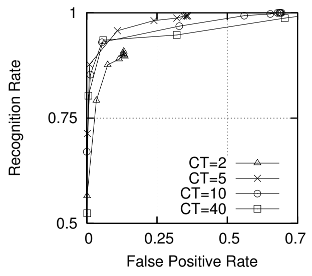
It is 0.0 if no objects have been detected.



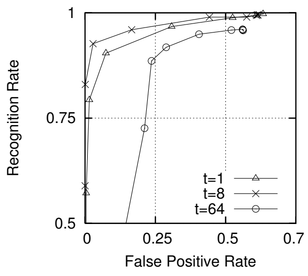
(a) Comparison of clustering thresholds in the AIBO domain



(b) Comparison of minimum cluster size per frame in the AIBO domain



(c) Comparison of clustering thresholds in the QRIO domain



(d) Comparison of temporal filtering sizes in the QRIO domain

Figure 2: ROC Curves analyzing the effects of various parameters. Each Curve was generated by modifying the nearest neighbor threshold θ. Each datapoint represents an average of running the algorithm on an entire video.

Performance

The following experimental results were created using the videos described in the previous section. After the annotation step, the training dataset of the AIBO domain contained a total of approximately 110,000 features. This dataset was then reduced using agglomerative clustering to have a total 15,000 features. The dataset of the QRIO domain was similarly reduced from about 132,000 to 4,575 features.

Some interesting frames of the AIBO domain's testing video and their detection results can be seen in figure 4. This video contained various perspectives and configurations that have never been encountered during the training stage. One such configuration can be seen in figure 4(f) where two robots are performing a special "celebration" move which mainly exposes the robots' underside to the camera. Detection fails in this instance as none of the visible local features were ever observed during training. Two frames of the QRIO domain and their detection results can be seen in figure 3.

PCA-SIFT is certainly not immune to finding false positives. It is possible that a completely different object generates features that coincide with the ones from the training object due to local similarities. Although voting and temporal filtering are able to reduce these coincidental outliers, we can still end up with the generation of false positives. Figure 4(g) contains such a scene where some features in the cluttered background were falsely detected to belong to the trained object.

Figures 2(a) and (c) analyze the performance of our approach under different voting space clustering thresholds δ (abbreviated CT in the figures). As predicted, choosing a threshold too small as well as choosing a threshold too large will deliver unoptimal results. A threshold in the range from 5 to 20 turned out to deliver the best results in both of our testing domains.

Figure 2(d) evaluates the impact of the proposed temporal clustering mechanism in the QRIO domain. We compare a temporal size of one frame (meaning no temporal clustering) with a temporal size of 8 and 64 frames respectively. We can clearly see that a short temporal filtering range does in fact significantly reduce the rate of false positives, while an excessively long temporal filtering range actually worsens the results.

In figure 2(b) we can see the impact of different minimum cluster size thresholds in the AIBO domain. While a minimum cluster size of one tends to be too optimistic, a minimum size of two seems to significantly reduce the rate of false positives while only slightly decreasing in the maximum recognition rate. A minimum size of anything greater than two starts to significantly hurt the recognition rate.

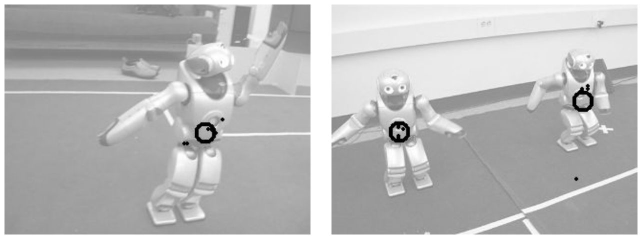


Figure 3: Two frames from the QRIO domain (visualized as bright background) and both of their centroid voting spaces. Each vote is annotated by a small black dot. The result of clustering and centroid-finding on this voting space is marked by larger black circles.

Real-time performance was a significant goal of this approach as it is intended to be used in a real robotic setting. On an Intel Core 1.8GHz processor at a 320x240 resolution, the algorithm performed typically between 1 and 10 frames per second (using a training dataset with approximately 4000 features). At a resolution of 176x144, the algorithm performs at approximately 5 to 15 frames per second on the same machine. For the above mentioned training data, about 50% of processing time was spent on building the PCA-SIFT vectors. Another 40% was spent on the nearest neighbor search through the compressed training dataset. Only 10% or less were spent on the actual clustering of the voting space when using Mean-Shift clustering.

Conclusion and Future Work

We have presented a simple approach to use PCA-SIFT for detecting multiple deformable objects in real-time. We have furthermore analyzed its performance using ROC statistics.

It should be noted that although the robots used in our experimental setup are highly deformable, they generally

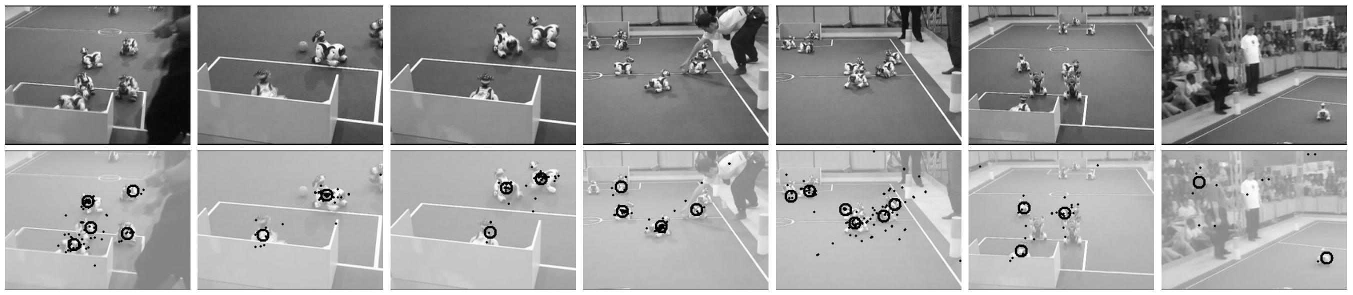


Figure 4: A selection of interesting frames from the testing video of the AIBO domain. The top row shows the input frames. The bottom row shows the centroid voting space after running PCA-SIFT on the input frames. Each vote is annotated by a small black dot. The result of clustering and centroid-finding on this voting space is marked by larger black circles.

all share the same hardware characteristics. This means that besides dynamic joint configurations and minor surface differences, there exists only small inter-class variation in terms of texture. We are aware of the fact that some other object-classes such as pedestrians in human scenes or certain types of animals are not only deformable, but also have an extremely high inter-class variation of shape, texture, and color. Although we did not analyze the performance of our algorithm for these types of classes, we can make the educated guess that our approach will lose some of its effectiveness when trained on only a small subset of instances from this type of classes. A precise evaluation for these classes will be an interesting topic for future work.

A more general limitation of our approach is that it requires feature-rich objects to perform well. This is luckily the case for legged robots which generate a sufficient amount of interest points. There certainly exist geometrically and texturally simple object classes (such as the single-colored ball in a RoboCup game) which will not generate many features. For these sparse kind of object classes, other methods such as contour matching might turn out to be more effective than our approach.

One significant feature which we have not discussed in this paper, is detecting the orientation of objects. Theoretically, it should be possible to retrieve orientation from objects using the same annotation and clustering methods that we used for centroid finding. A detailed test and analysis should be part of future research.

Furthermore, the current approach presented in this paper does not make any use of color. For future work, it would be interesting to investigate how to best integrate color-information into the presented voting scheme, and to analyze what impact this would have on the algorithm's performance.

References

Cheng, Y. 1995. Mean Shift, Mode Seeking, and Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(8):790–799.

Cootes, T.; Edwards, G.; and Taylor, C. 2001. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(6):681–685.

Gavrila, D., and Philomin, V. 1999. Real-time object detection for smart vehicles. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* 1.

Gowda, K., and Krishna, G. 1978. Agglomerative clustering using the concept of mutual nearest neighborhood. *Pattern Recognition* 10(2):105–112.

Ke, Y., and Sukthankar, R. 2004. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. *Proc. CVPR* 2:506–513.

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; and Osawa, E. 1997. RoboCup: The Robot World Cup Initiative. *Proceedings of the first international conference on Autonomous agents* 340–347.

Leibe, B.; Seemann, E.; and Schiele, B. 2005. Pedestrian detection in crowded scenes. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* 1.

Lowe, D. 1999. Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* 2:1150–1157.

Mikolajczyk, K., and Schmid, C. 2003. A performance evaluation of local descriptors. *IEEE Conference on Computer Vision and Pattern Recognition* 2:257–264.

Moradi, M.; Abolmaesoumi, P.; and Mousavi, P. 2006. Deformable Registration Using Scale Space Keypoints. *Proceedings of SPIE* 6144:61442G.

Olson, C., and Huttenlocher, D. 1997. Automatic target recognition by matching oriented edge pixels. *Image Processing, IEEE Transactions on* 6(1):103–113.

Se, S.; Lowe, D.; and Little, J. 2001. Vision-based mobile robot localization and mapping using scale-invariant features. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* 2.