Detection and Localization of Multiple Objects

Stefan Zickler and Manuela M. Veloso
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213-3890
Email: {szickler, veloso}@cs.cmu.edu

***Abstract*— Being able to identify and localize objects is an important requirement for various humanoid robot applications. In this paper we present a method which uses PCA-SIFT in combination with a clustered voting scheme to achieve detection and localization of multiple objects in real-time video data. Our approach provides robustness against constraints that are common for humanoid vision systems such as perspective changes, partial occlusion, and motion blurring. We analyze and evaluate the performance of our method in two concrete humanoid test-scenarios.**

I. Introduction

SIFT (Scale Invariant Feature Transform) has been shown to be an effective descriptor for traditional object recognition applications in static images. In this paper, we propose and evaluate a method that uses PCA-SIFT [1] in combination with a clustered voting scheme to achieve detection and localization of multiple objects in video footage as it is typically collected by a humanoid robot's vision system. A common constraint of humanoid vision is the occurrence of frequent perspective changes and motion blurring which are often caused by the robot's walking motions. Our approach attempts to minimize the effects of these problems while still being applicable as a real-time algorithm.

A flowchart of our approach is depicted in figure 1(a). First, a training video is recorded by observing the object from various perspectives. PCA-SIFT keypoints are then generated for each video frame. It should be emphasized that by training on an entire video sequence, we are able to capture impressions of the object from a continuous spectrum of poses, and under various configurations of lighting and video noise. This allows us to gain a more complete representation of the object in PCA-SIFT space compared to training from very few still images. The keypoint generation stage is followed by a manual annotation step which allows the user to approximate the boundary and position of the relevant object in the video. All keypoints that lie outside of the hand-annotated boundary are rejected. The annotation step is furthermore used to determine each keypoint's relative location toward the annotated object's center. All retained features build the initial training dataset. The dataset is then post-processed by a clustering algorithm to compress its overall size.

During the recognition stage, we again generate PCA-SIFT keypoints for each incoming video frame. A nearest neighbor search is performed on the training dataset for each PCA-SIFT keypoint while enforcing a maximum distance threshold. Using the relative location obtained during hand-annotation, each matching feature then votes for a 2D-coordinate predicting the center of the object. Clustering and filtering is applied to the voting space and the centroids of each remaining cluster are calculated. This becomes our final detection result. Additionally, we demonstrate how this voting space can be extended temporally to become a simple filter which is able to reduce the rate of false positives.

This paper is organized as follows. Section II introduces some of the related work in this area and provides a short review of the SIFT feature descriptor. The body of this paper is located in section III where we explain our algorithm in detail. Results from our experimental domains are presented and analyzed in section IV. Concluding remarks and ideas about future work are presented in section V.

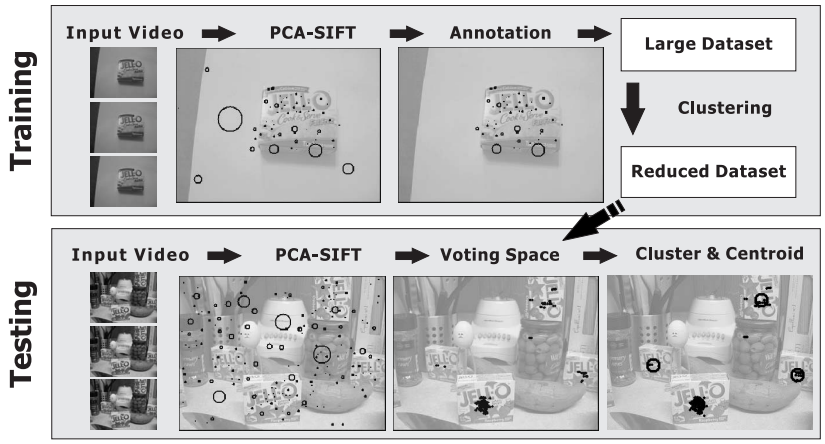
II. Related Work

It has been shown that SIFT descriptors can be used to achieve fairly robust object detection in still images [2]. SIFT has furthermore been used in several other related applications such as metric robot localization [3] and medical imaging [4]; and it was shown to be one of the best currently available descriptors in a comparative study [5].

Various related approaches exist on Real-Time object recognition for robotic applications. Very simplistic models, such as color segmentation are fast, but normally require some type of colored object marker. Oftentimes, these approaches are also highly sensitive to lighting conditions [6].

Another common approach, which is used in pedestrian detection, is contour matching using Distance Transforms (e.g. Chamfer Distance) [7]. The inherent downside of this methodology is the fact that it operates only on an object's contour, and does not take any of an object's inner features into account. While this may be suitable for uniquely shaped objects with sparse textures, it tends to fail for objects with simple, ambiguous shapes. Oriented Edges [8] can reduce this ambiguity; however, they introduce the problem of relying on a unique gradient between the object and its background, which is very likely to change in real world scenes.

Parts-based object recognition is another successful recent approach. However, most parts-based recognition algorithms still suffer from very high computational costs, even when a sparse object representation is used [9] [10].



(a) A simplified flowchart of the presented approach

(b) QRIO, observing the desk domain

Fig. 1.

A. 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 [1] 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.

III. 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. 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 $\mathit{locrel}_{ij} = c_i - \mathrm{loc}(k_{ij})$.
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:

$$p_l = \text{pos}(t_l) + \left(\frac{\text{scale}(k_{ij})}{\text{scale}(t_l)} \times \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \times \textit{locrel}_l \right)$$

where $\beta = \text{orientation}(k_{ij}) - \text{orientation}(t_l)$.

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.

A. 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 [1]. 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.

B. 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 $\textit{locrel}_{ij}$ of each PCA-SIFT feature toward the center of the object c_i is furthermore computed as described in III. All PCA-SIFT features lying outside of any annotated region are rejected from the training set.

It should be noted that this manual annotation step is not necessarily required. One possible way to automate the annotation process could be to place the object in front of a bluescreen-like surface and then perform an automatic color-segmentation. While this would simplify the annotation process, it would also introduce new constraints on the training stage (such as the requirement of a monochromatic background) which is the main reason why we did not pursue this approach.

C. 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 [11] which merges similar features in the dataset. 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.

D. 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 described in III. 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.

E. 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} as described in III. 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. However, we can optimistically assume that most of the votes do in fact approximately match the object's center and that our voting scheme should 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.

F. 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 [12] and agglomerative clustering. Both of them performed virtually identical in terms of quality. While Mean-Shift should be considered a 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 does run 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 'band-width' 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 further below, 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 any clusters that contain less votes than a certain minimum cluster size threshold s. The center of mass is computed for each remaining cluster which becomes our final detection result.

G. 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 paragraph 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.

This temporal voting scheme is one of many possible ways to exploit the fact that the data are a continuous video-stream. Other possibilities not discussed in this paper could be Kalman or Particle Filters which would rather be layered on top of the native voting space.

IV. Experimental Results

We test our approach in two different domains. The first domain is a cluttered kitchen scene. The objective is to detect and locate all instances of a certain food product on a kitchen counter. The data of this domain originate from unedited grayscale video footage taken at 320x240 pixel resolution with a handheld camera. The second domain has the goal of detecting telephones in a highly cluttered desk scene. The data of this domain was gathered using the internal camera of a moving SONY QRIO humanoid robot as it is depicted in figure 1(b). The videos retrieved from the QRIO robot were at a 176x144 pixel resolution and contained clearly visible artifacts from a high JPEG compression factor which made this a very challenging domain.

A. 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. In our particular experiments we chose to place the object in an independent, uncluttered environment during the training stage. This was mainly done to ensure contextual independence between the training and the testing stage and is not a general requirement of our algorithm.

B. Testing Data

For testing, multiple instances of the object were placed in an extremely cluttered scene. The objects were placed in various positions and orientations, and some of them were partially occluded to increase the demand of robustness from our approach. Although the two scenes were recorded using different sources, both of them feature a similar "walk-by" with continuously changing perspectives parallel to the walking plane, containing additional vertical variations due to the walking motion of the observer. The desk domain contains slightly greater amounts of motion blurring due to the rather rapid and uncompensated walking motions of the QRIO robot.

C. 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.

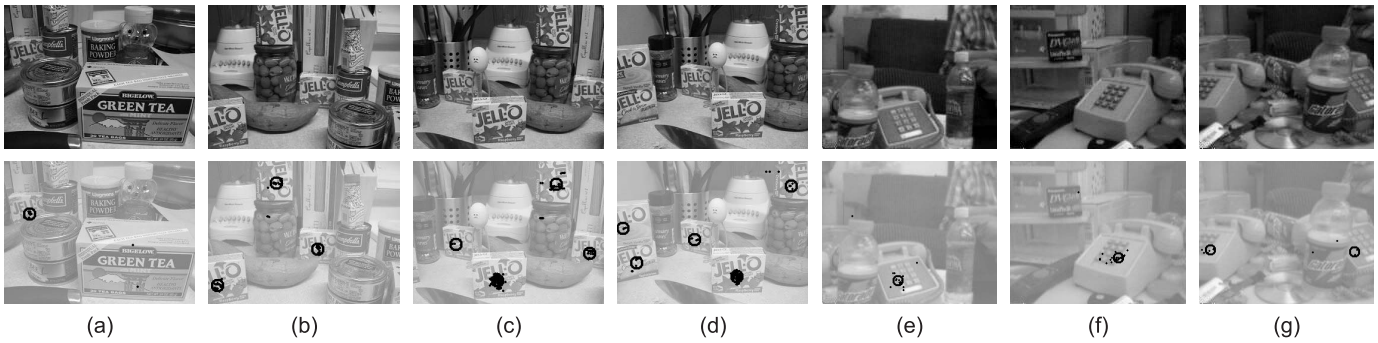
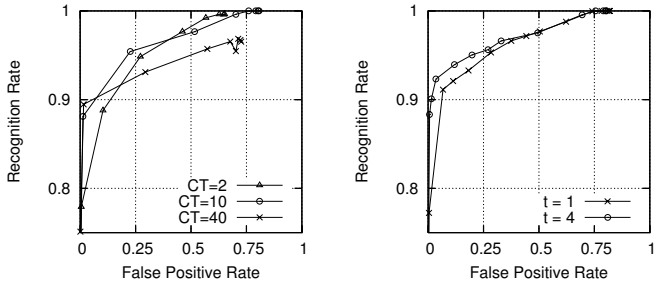


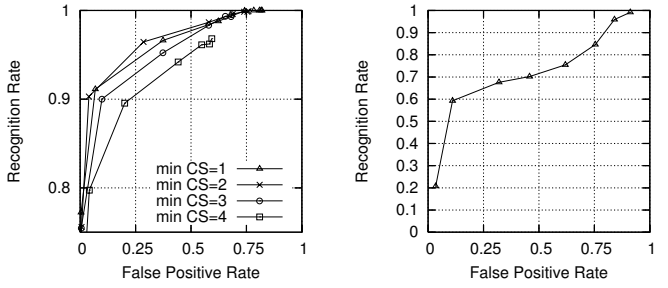
Fig. 2. A selection of interesting frames from the testing video of both the kitchen domain (a to d) and the desk domain (e to g). The top row shows the input frames. The bottom row shows the centroid voting space after running PCA-SIFT on the input frames. Each vote is annotated by a small black dot. The result of clustering and centroid-finding on this voting space is marked by larger black circles. These results were generated using a temporal filtering size of 4, a clustering threshold of 10, and a minimum cluster size of 4 (1 per temporal frame).

Fig. 3. A snapshot from the training sequence of the kitchen domain (left) and the locations of its annotation-filtered PCA-SIFT keypoints (right).



(a) Comparison of clustering thresholds

(b) Comparison of temporal filtering sizes



(c) Comparison of minimum cluster size per frame

(d) ROC of desk domain

Fig. 4. ROC Curves analyzing the effects of various parameters. Each Curve was generated by modifying the nearest neighbor threshold. Each datapoint represents the average over an entire testing video sequence.

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

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

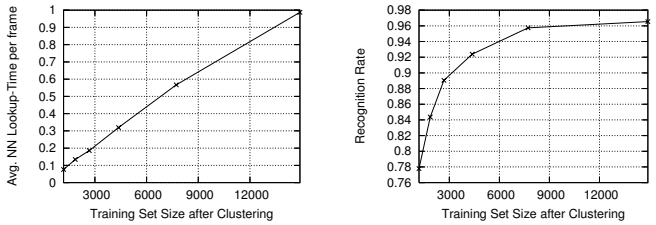
It is 0.0 if no objects have been detected.

D. Performance

In our kitchen domain, the object that was trained for was a package of "Jello" as it can be seen in figure 3. After annotation filtering, the training dataset contained a total of approximately 124,000 features. This dataset was reduced using agglomerative clustering to have a total 4,372 features. The testing video which was used for the following performance analysis had a total of 539 frames and contained six instances of the 'Jello' object spread throughout the scene in various positions and under partial occlusion. Some interesting frames of the testing video can be seen in frames (a) to (d) of the top row of figure 2.

In the desk domain, the object that was trained for was a telephone as it can be seen in the top row of figure 2(e) to (g). The original training dataset of 50,000 features was reduced to a size of 6053 features. The testing video had 503 frames with two instances of the phone object, again under partial occlusion and in two different poses.

Figure 4(a) analyzes the performance of our approach in the kitchen domain under different voting space clustering thresholds δ (abbreviated CT in the figure). As predicted, choosing a threshold too small (such as one or two) as well as choosing a threshold too large (such as 40) will deliver slightly unoptimal results. A threshold in the range from five to 20 turned out to deliver the best results.



(a) Runtime (b) Recognition Rate

Fig. 5. Effects of the training dataset size on runtime and precision.

Figure 4(b) evaluates the impact of the proposed temporal clustering mechanism in the kitchen domain. We compare a temporal size of one frame (meaning no temporal clustering) with a temporal size of four frames. We can see that temporal filtering does in fact visibly reduce the rate of false positives.

In figure 4(c) we can see the impact of different minimum cluster size thresholds in the kitchen 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 4(d) shows a short performance evaluation of our approach in the desk domain. Here, our effective recognition rate lies in the range from 60% to 70% which is significantly less than the rates in the 90% to 95% range of the kitchen domain. One major difference between the two domains that could have caused this decrease, is the fact that the data of the desk domain had an extremely low resolution with a high JPEG compression factor which added additional noise to it.

All of the above experiments in the kitchen domain were performed with the same training dataset of size 4,372. However, a comparison of various different training set sizes was performed to determine the effects on precision and runtime. In figure 5(a) we can see that the nearest neighbor lookup time scales linearly with the size of the training set. Figure 5(b), however gives us an indication of the training set size's effect on the recognition rate. A training set size of approximately 10,000 (about 1/10 of the original dataset set) leaves us with a recognition rate of 95%. When reducing the training set to 4,372 features (the size used in the above experiments) we still retain slightly more than 92% recognition rate while splitting the runtime approximately in half (from 0.7 to 0.3 seconds).

E. Speed

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 performs typically at 1 to 10 frames per second (using a training dataset with approximately 2000 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.

V. Conclusion and Future Work

We have presented a simple approach to use PCA-SIFT for detecting and localizing multiple objects in video footage that is typical of humanoid vision systems. We have furthermore analyzed its performance using ROC statistics on real-world experiments.

A general limitation of our approach is that it requires feature-rich objects to perform well. There certainly exist geometrically and texturally simple object classes which will not generate many features. For these sparse kind of object classes, other methods such as contour matching might turn out more effectively than our approach.

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

References

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

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

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

[4] M. Moradi, P. Abolmaesoumi, and P. Mousavi, "Deformable Registration Using Scale Space Keypoints," *Proceedings of SPIE*, vol. 6144, p. 61442G, 2006.

[5] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 257–264, 2003.

[6] J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3, pp. 2061–2066, 2000.

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

[8] C. Olson and D. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *Image Processing, IEEE Transactions on*, vol. 6, no. 1, pp. 103–113, 1997.

[9] S. Agarwal and D. Roth, "Learning a sparse representation for object detection," *Proc. ECCV*, vol. 4, pp. 113–130, 2002.

[10] H. Schneiderman and T. Kanade, "Object Detection Using the Statistics of Parts," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 151–177, 2004.

[11] K. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighborhood," *Pattern Recognition*, vol. 10, no. 2, pp. 105–112, 1978.

[12] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.