Rapid Detection of Many Object Instances

Suwan Tongphu, Naddao Thongsak, and Matthew N. Dailey

Computer Science and Information Management
Asian Institute of Technology
{Suwan.Tongphu,Naddao.Thongsak,mdailey}@ait.ac.th

Abstract. We describe an algorithm capable of detecting multiple object instances within a scene in the presence of changes in object viewpoint. Our approach consists of first calculating frequency vectors for discrete feature vector clusters (visual words) within a sliding window as a representation of the image patch. We then classify each patch using an AdaBoost classifier whose weak classifier simply applies a threshold to one visual word's frequency within the patch. Compared to previous work, our algorithm is simpler yet performs remarkably well on scenes containing many object instances. The method requires relatively few training examples and consumes 2.2 seconds on commodity hardware to process an image of size 640×480. In a test on a challenging car detection problem using a relatively small training set, our implementation dramatically outperforms the detection performance of a standard AdaBoost cascade using Haar-like features.

Keywords: Object detection, keypoint descriptors, clustering, visual words, bags of keypoints, AdaBoost.

1 Introduction

Many applications in areas as diverse as biomedical imaging, manufacturing, image retrieval, video surveillance, and autonomous search and rescue would benefit from robust single-image object detection. A large number of approaches have been proposed. Here we focus on methods capable of detecting and accurately localizing *multiple instances* of a given object category in a *single static image*. These methods are obviously applicable to problems such as cell counting in biomedical imaging, in which only a single image is available. However, the methods are also extremely useful for problems involving multiple target tracking with a fixed or moving video camera, since such trackers need to be initialized based on the first video frame. Depending on the tracking application, it may also be necessary to detect and track new instances as they appear in the scene.

When the goal is to detect and localize each instance of an object class in a static image, the typical approach is to run a sliding window over the image, independently classify each image patch, then try to combine multiple detections in the same region. *Template matching*, in which the image patch is normalized then compared to a bank of training examples, is perhaps the simplest method based on this approach. More sophisticated approaches that apply local filters and discriminative classifiers, e.g., the Viola and Jones Haar-like feature AdaBoost cascade [1], achieve impressive performance in applications such as face detection, in which the object has important features such as eyes in particular relative locations. Many improvements on the Viola and Jones approach have been proposed; for example, Huang et al. [2] use a more flexible yet efficient local filter set and a decision tree structure tailored to face detection under arbitrary in-plane and out-of-plane rotation. However, these approaches may not be ideal for detecting objects with more variable appearance, such as cars, people, buildings, and so on, unless a prohibitively enormous training set is used.

Methods that are more flexible, attempting to perform classification while explicitly allowing for changes in the precise spatial relationships between object parts, can be roughly grouped into those based on *image patch analysis* and *sparse feature analysis*.

Image patch analysis assumes that the target object is composed of a flexible configuration of many small image patches. Mohan et al. [3] propose a component-based method for detecting humans under clutter and occlusion. They train separate support vector machines (SVMs) to detect parts of the entire object (arms, legs, etc.) in restricted regions of the detection window, then train a higher-level SVM to impose geometric constraints among the constituent parts. Keysers et al. [4] first detect interest points, extract image patches around the interest points, scale each patch to a common size, then map each patch to a cluster ID. Finally, the authors apply Breuel's RAST (Recognition by Adaptive Subdivision of Transformation Space) algorithm [5] to find the best rotation, translation, and scale aligning the points on a test image with the matching points on a gallery image.

Sparse feature analysis, on the other hand, uses interest point descriptors that are robust to changes of viewpoint, scale and orientation to increase detection performance. Kim and Dayhot [6] propose a component-based object detection method using SURF [7] and two types of SVMs. After SURF feature extraction, the first SVM filters out points unlikely to lie on objects of interest, and the second bank of SVMs classifies the feature descriptors into likely object components. Finally, the algorithm applies geometrical constraints between candidate components prior to final classification. Csurka et al. [8] introduced the idea of categorizing *bags of keypoints* in a way similar to the way text classification systems categorize bags of words. The authors extract keypoints using the Harris affine detector [9] and compute SIFT descriptors [10] for those keypoints. They then perform vector quantization on the descriptors using a k-means model, and, for purposes of classification, construct a feature summarizing the frequency of occurrence of each keypoint cluster across the image. Finally, they use naive Bayes or a SVM to classify frequency vectors. Das et al. [11] extend this idea to object detection. They use probabilistic latent semantic analysis (PLSA) with a bag-of-visual-words image patch model. SIFT features within a region of interest (ROI) are mapped to discrete visual words obtained during training, visual word occurrence vectors are mapped to a latent topic representation with PLSA. Poor candidate objects are filtered out using various appearance and contextual constraints, and an SVM-based classifier is used to map strong candidate regions to object classes. Although the algorithm performs well on images with multiple object instances, it does not run in real time. Zickler and colleagues [12,13] map observed PCA-SIFT features to cluster centers derived at training time then use each PCA-SIFT feature's scale and orientation to vote for an object center point. The votes from the different feature points are clustered to obtain a final set of object centers. One benefit of the voting approach is that it does not require sliding a detection window over every possible object location and scale. Virtually all sparse feature based techniques can benefit from low-dimensional representations of the descriptors, obtained through general unsupervised dimensionality reduction of standard descriptors [14,13] or discriminatory descriptor learning [15].

In both the image patch and sparse feature approaches, a variety of ways to model geometric constraints between object parts have been proposed. Fergus et al.'s generative model [14] designates one feature point as the "landmark" and models dependent features (which can be occluded) with respect to the landmark feature in a star configuration. Crandall et al. [16] propose an image patch based k-fan model that generalizes the bag-of-patches model ($k = 0$) that does not model spatial configuration and the star model ($k = 1$) that relates each dependent feature to one parent landmark. Higher order models relate dependent features to k parent nodes. All of the generative spatial models simplify computation relative to fully connected models by making various conditional independence assumptions about object parts.

In this paper, we introduce a new object detection algorithm based on sparse features. Our method is similar to other methods based on bags of keypoints [8,11]. We use the basic sliding window approach, mapping feature descriptors to discrete visual words then calculating frequency vectors for those visual words within each window. We then classify each patch using an AdaBoost classifier whose weak classifier simply applies a threshold to one visual word's frequency within the patch. Compared to the previous object detection work, the algorithm is simpler yet performs remarkably well on scenes containing many object instances. The method requires relatively few training examples and in our current implementation consumes 2.2 seconds on commodity hardware to process images of size 640×480. In a test on a challenging car detection problem using a relatively small training set, our implementation dramatically outperforms the detection performance of a standard AdaBoost cascade using Haar-like features.

2 Rapid Multiple-Instance Object Detection

2.1 Overview

Before providing the details of the proposed method, we give a rough overview of the architecture. Fig. 1 provides a schematic overview of our method, which consists of a training and a testing stage.

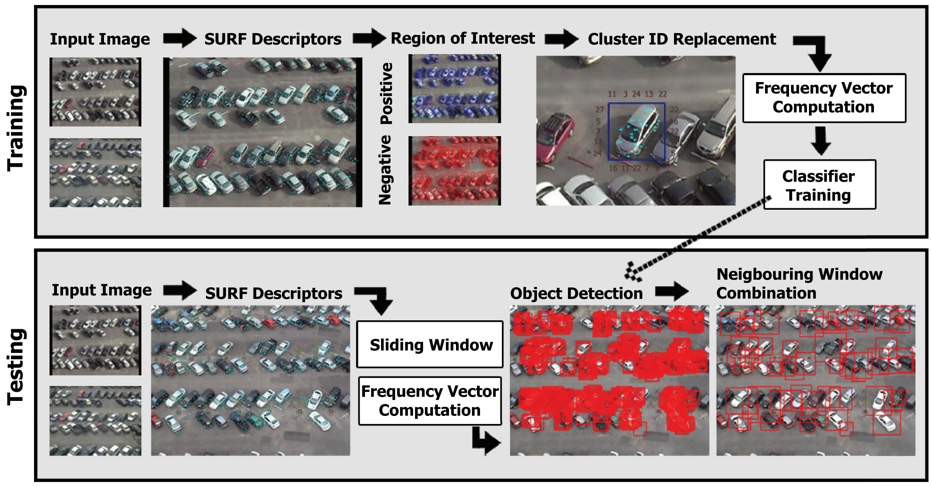

Fig. 1. System overview of the proposed method, showing training and testing flow

The training stage is performed only once and aims to learn a representation of the target object. The following algorithm describes the steps in the training process:

1. For any image I_i, we extract a set D_i of N_i SURF descriptors, where $D_i = \{d_{i,j}\}, 1 \leq j \leq N_i$.
2. From a set of training images, we collect and cluster the SURF descriptors into C groups using the k-means algorithm.
3. From a separate set of training images, for each example object of interest, we create a mask from the object's square bounding box by hand. We find the descriptors $d_{i,j}$ in D_i lying within the mask region, and map each descriptor to the nearest cluster center

$$\operatorname*{argmin}_{c_k \in C} d(d_{i,j}, c_k)$$

   where $d(\cdot,\cdot)$ is the Euclidean distance between the input feature point and the input cluster center. We then construct a frequency vector summarizing the cluster centers or "visual words" found within the mask region.
4. From the same of training images, we label negative regions that do not contain the object of interest or only contain partial objects of interest, then construct negative frequency vectors in the same manner as for the positive locations.
5. We train an AdaBoost classifier using a threshold on the frequency of a single visual word as the weak classifier.

In the testing stage, for image I_i, we perform the following steps:

1. Extract a set D_i of N_i SURF descriptors, where $D_i = \{d_{i,j}\}, 1 \leq j \leq N_i$.

2. Move a sliding window over the input image, and for each window,
   (a) Find the nearest cluster center for each descriptor $d_{i,j}$ within the window.
   (b) Compute a frequency vector for the descriptors found within the window.
   (c) Pass the frequency vector to the AdaBoost classifier.
3. Group positive detections within a threshold distance θ of each other into a single detection.

2.2 Training Phase

Keypoint and Descriptor Extraction. We extract descriptors from the image and use them as the representative feature vectors instead of using image pixels directly. In our experiments, we actually consider two different feature descriptors: SIFT (Scale Invariance Feature Transform) [10] and SURF (Speeded-Up Robust Feature) [7]. These feature descriptors provide rich information about local texture that is tolerant to changes in scale and orientation. However, in our experiments, we find that although SIFT provides higher detection accuracy than SURF, because of its fine-scale processing which aims to achieve scale invariance, the multilevel image pyramid computation requires much more time. We therefore primarily use faster the SURF descriptor. Extracting SURF descriptors involves computing keypoints as local maxima of an approximation to the determinant of the local Hessian matrix, assigning scales to keypoints, computing a dominant orientation, then extracting Gaussian-weighted local gradient information in a square window around the keypoint.

Visual Word Representation. In this step, all descriptors extracted from training images are grouped together in order to construct word terms. More specifically, we use k-means to cluster descriptors into many groups which have similar characteristics. We determine the optimal number of classes k experimentally. Following previous research, we call each class a "visual word."

Regions of Interest. For training, we manually label square image regions as either containing a target object or background. A region is considered positive if and only if it covers most of the object's components. Partial objects are treated as negative examples. From the feature descriptors within each square ROI, we find the nearest cluster centers, which are used in the next step. The bounding box typically contains a small background region or parts of other objects along with the object of interest, so the extracted descriptors are typically somewhat noisy.

Frequency Vector. In this step, we count the frequency of occurrence of each visual word in each positive- or negative-labeled square ROI. That is, a k-bin histogram of the SURF descriptors in each labeled window is computed. The vectors are separated into positive and negative training examples, which are used as input to the classifier training procedure.

Classification Model. The positive and negative example frequency vectors are passed to the AdaBoost (Adaptive Boosting) [17] binary classifier training procedure. The strong classifier produced by AdaBoost is constructed from many weak classifiers. In each iteration of the training process, the best candidate weak classifier is selected according to its ability to classify the weighted training set, and on each iteration, the weights of the training examples are updated to give incorrectly classified examples higher weight. In this work, we use a standard monolithic AdaBoost classifier, but a cascade structure similar to that of Viola and Jones might help to speed up the detection process.

2.3 Testing Phase

Keypoint and Descriptor Extraction. As in the training phase, we extract SURF descriptors from the input image. We insert the descriptors into a kd-tree to speed up search time. We then move a sliding window over the entire scene, and for each window, we extract the descriptors within that window using the kd-tree.

Visual Word Representation. Using the visual word model created during training, we map the detected keypoints to visual words as previously described.

Visual Word Frequency Vector for ROI. In this step, the frequency of occurrence of each visual word term within a particular detection window is counted. As during training, we generate a k-bin feature frequency vector. This k-dimensional vector is used to represent an image patch and is used as input to the classifier. As is the case for training ROIs, the frequency vector will typically contain some additive noise in the form of keypoints from the background region within the ROI.

Classification. At this point, we use the AdaBoost classifier from the training stage. We pass the feature frequency vector for each detection window location through the classifier. Clearly, if one detection window position contains an object of interest, nearby detection window locations will also contain the object of interest. Also, since detection window locations close to a positive object location will often contain the same or a very similar set of keypoints, we will typically obtain several positive detections in the neighborhood of a target object. We thus combine neighboring windows into a single detection using a distance threshold.

3 Evaluation

3.1 Experimental Design

In this section, we describe an experimental evaluation of our method on the task of car detection in a surveillance video acquired from a large parking lot. We mounted the camera on a building above the parking lot and captured 9 hours of data in three sessions on 3 different days. An example frame from the test video is shown in Fig. 2.

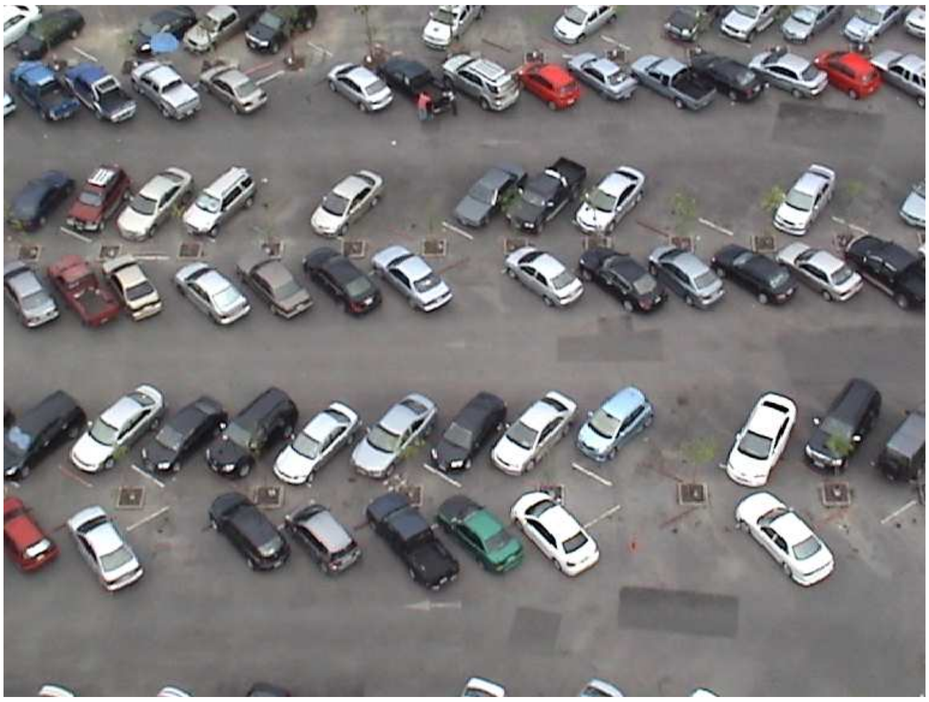

Fig. 2. Test image extracted from surveillance video for day 3

From the three sessions, we selected the first two days' video data as training data and the third day's video data as test data. In the training data, we manually labeled the location of each car as a positive example and also labeled a large number of background and partial car patches as negative examples. We obtained about 3,000 positive and 7,000 negative training windows through this process. As test data, we simply selected one frame from the third day's surveillance video and labeled the positive locations as ground truth. Fig. 2 shows this frame.

In a first experiment, as a baseline for comparison, we evaluated a standard Viola and Jones AdaBoost cascade using Haar-like features [1] on our car detection dataset. In a second experiment, we compared two different versions of our detection algorithm, one using SURF descriptors and one using SIFT descriptors. In all cases, we used the same training set and the same (separate) test image. We predicted that SURF would outperform SIFT in terms of run time performance and that our sparse feature-based method would outperform the Haar-like feature-based method in terms of accuracy, given the relatively small training set.

3.2 Experiment 1 Results

We built the baseline AdaBoost cascade using the `haartraining` utility from OpenCV [18]. We used a minimum hit rate threshold of 0.995 and a maximum false positive rate of 0.5 for each stage. The trained cascade contained 10 stages

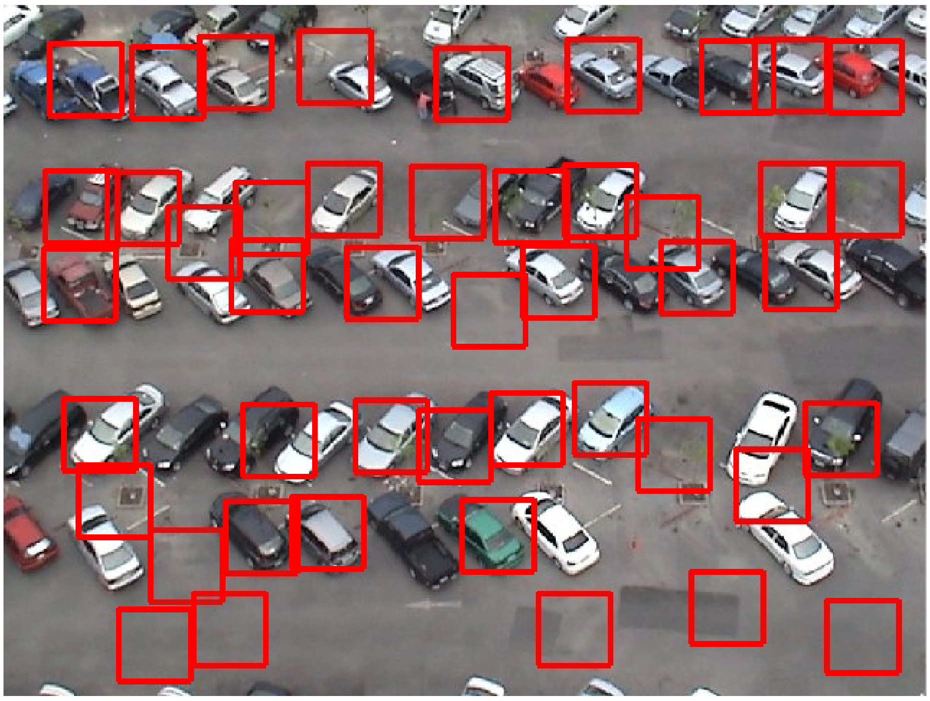

Fig. 3. Detection results for the baseline AdaBoost cascade using Haar-like features

comprised of a total of 234 weak classifiers and had an overall hit rate and false positive rate of 0.996 and 0.249, respectively.

After training, we tested the final classifier on the test image (Fig. 2). We adapted the OpenCV `facedetect` demo to use the single known detection scale for the test data. The detection results are shown in Fig. 3 and the first row of Table 1.

3.3 Experiment 2 Results

We used the Hess implementation [19] of SIFT and the ETH implementation [20] of SURF. We extracted 570,000 128-dimensional SIFT feature descriptors and 320,000 SURF feature descriptors at both 64 and 128 dimensions from the 24 images in the clustering training set, and for each of the three feature sets, we built a model consisting of $k = 30$ clusters. We then computed feature frequency vectors for 3,000 positive and 7,000 negative windows in the training set images. Using these data, we built AdaBoost frequency vector classifiers using OpenCV's [18] generic AdaBoost implementation.

Using the same approach to sliding a constant-scale detection window over the test image as described for Experiment 1, we obtained the results shown in Figure 4 and the last three rows of Table 1.

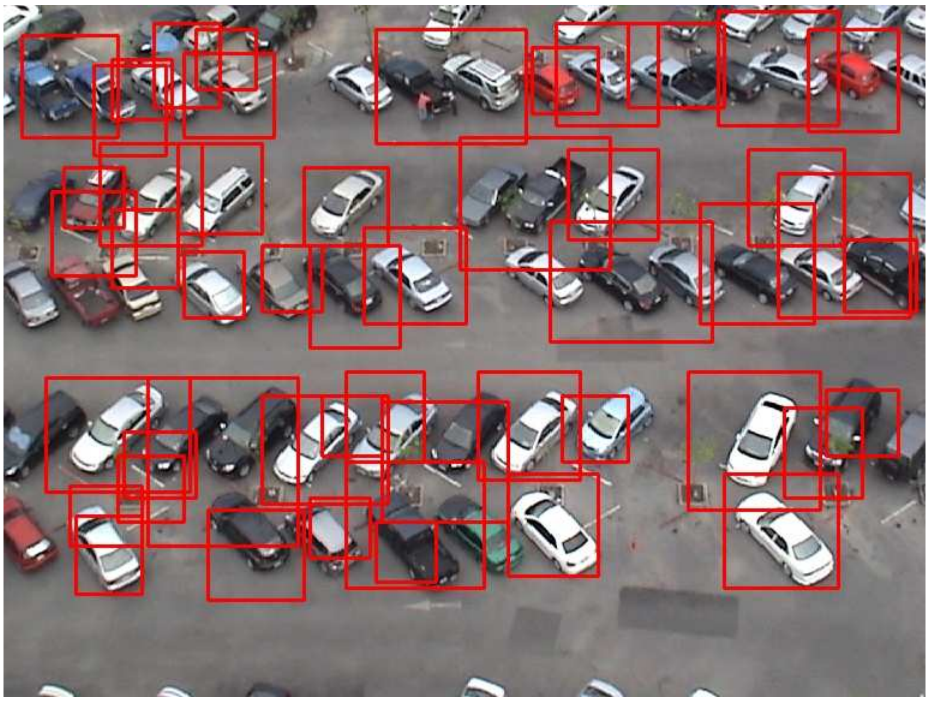

Fig. 4. Results from proposed object detection algorithm using 128-dimensional SURF descriptors

Table 1. Car detection results. The test scene (Fig. 2) contains 49 target objects. The feature extraction, classification, and overall times are measured in seconds. For bag-of-keypoint methods, feature extraction time includes keypoint extraction and cluster center mapping; classification time includes detection window scanning, frequency vector computation, and AdaBoost-based classification.

Detector	Dims	Hits	Misses	FPs	Time (features)	Time (classification)	Time (overall)
Haar cascade	n/a	29	20	20	n/a	n/a	0.25
Bag of SIFT	128	40	9	4	8.78	0.89	9.66
Bag of SURF	128	37	12	6	1.65	0.54	2.19
Bag of SURF	64	24	25	16	1.40	0.70	2.10

4 Discussion and Conclusion

Examining the results of experiments 1 and 2, we find that the bag-of-keypoints approaches using SIFT or 128-dimensional SURF descriptors perform quite well in terms of detection rates and false positive rates. We also observe that extracting 128-dimensional SURF descriptors takes much less compute time than extracting SIFT descriptors. Although the detection accuracy of SURF is not quite as high as that of SIFT, we find it acceptable.

In comparison to the Haar cascade, our method is much more robust at the cost of a ninefold increase in compute time (0.25 seconds for the Haar cascade compared to 2.2 seconds for the 128-dimensional SURF keypoint based method). We can attribute the increased accuracy of the bag-of-keypoints approach to its rotation invariance, its robustness to non-rigid deformations of the object model, and its insensitivity to lighting conditions. Although it may be possible to dramatically increase the performance of the Haar cascade, the required training set would be prohibitively large, and the resulting detector would necessarily be more complex and most likely slower.

One limitation of our method is that it requires sufficient texture; it does not perform well on objects with color similar to the background. In future work, we will investigate directions to solve this issue. First, we aim to find alternative image features which are robust to the uncluttered object's color. Secondly, we will investigate spatial modeling of the relationships among object components in order to increase detection performance.

The other main limitation is that the run time performance is not as fast as it should be: practical use in real-time systems would in most cases require speedup by a factor of at least 4. We will further optimize our implementation in future work. Through algorithmic improvements and processor architecture-specific optimizations, a significant speedup should be possible. Cascading the AdaBoost classifier would improve classification time (the current classifier is monolithic). In some applications, it is possible to detect objects of interest only once then apply a tracking algorithm like the one proposed in [21]. Instead of detecting objects in every incoming frame, we would perform full detection only on key frames and then use a simpler appearance-based tracker for the intermediate frames. Another possible optimization is dimensionality reduction for the SURF feature descriptor using PCA or a discriminative dimensionality reduction method. This would speed up the nearest neighbor computation required for mapping features to visual words.

We are currently exploring applications of the proposed method in video surveillance systems and in unmanned aerial vehicle (UAV) missions.

Acknowledgments

S. Tongphu and N. Thongsak were supported by graduate fellowships from the Royal Thai Government.

References

1. Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
2. Huang, C., Ai, H., Li, Y., Lao, S.: High-performance rotation invariant multiview face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(4), 671–686 (2007)

3. Mohan, A., Papageorgiou, C., Poggio, T.: Example-based object detection in images by components. IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 349–361 (2001)
4. Keysers, D., Deselaers, T., Breuel, T.: Optimal geometric matching for patch-based object detection. Electronic Letters on Computer Vision and Image Analysis 6, 44–54 (2007)
5. Breuel, T.M.: Implementation techniques for geometric branch-and-bound matching methods. Computer Vision and Image Understanding 90, 294 (2003)
6. Kim, D., Dahyot, R.: Face components detection using SURF descriptors and SVMs. In: International Machine Vision and Image Processing Conference, pp. 51–56 (2008)
7. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (SURF). Computer Vision and Image Understanding 110(3), 346–359 (2008)
8. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Proceedings of the ECCV International Workshop on Statistical Learning in Computer Vision (2004)
9. Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 128–142. Springer, Heidelberg (2002)
10. Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision, ICCV (1999)
11. Das, D., Masur, A., Kobayashi, Y., Kuno, Y.: An integrated method for multiple object detection and localization. In: Proceedings of the 4th International Symposium on Advances in Visual Computing (ISVC), pp. 133–144 (2008)
12. Zickler, S., Veloso, M.: Detection and localization of multiple objects. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pp. 20–25 (2006)
13. Zickler, S., Efros, A.: Detection of multiple deformable objects using PCA-SIFT. In: Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 1127–1133 (2007)
14. Fergus, R., Perona, P., Zisserman, A.: A sparse object category model for efficient learning and exhaustive recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 1, 380–387 (2005)
15. Hua, G., Brown, M., Winder, S.A.J.: Discriminant embedding for local image descriptors. International Journal of Computer Vision, 1–8 (2007)
16. Crandall, D., Felzenszwalb, P., Huttenlocher, D.: Spatial priors for part-based recognition using statistical models, pp. 10–17 (2005)
17. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, P.M.B. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 23–37. Springer, Heidelberg (1995)
18. OpenCV Community: Open source computer vision library version 1.1, C source code (2008), `http://sourceforge.net/projects/opencvlibrary/`
19. Hess, R.: SIFT feature detector, C source code (2006), `http://web.engr.oregonstate.edu/~hess/index.html`
20. Bay, H., van Gool, L., Tuytelaars, T.: SURF version 1.0.9 (2006), C source code, `http://www.vision.ee.ethz.ch/~surf/`
21. Okuma, K., Taleghani, A., Freitas, N.D., Freitas, O.D., Little, J.J., Lowe, D.G.: A boosted particle filter: Multitarget detection and tracking. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 28–39. Springer, Heidelberg (2004)