Object Detection

Overview of Algorithms and Approaches

Technical Report - FIT - VG20102015006 - 2012 - 02

Ing. Štěpán Mráček

Faculty of Information Technology, Brno University of Technology

December 6, 2012

Abstract

This report brings an overview of the techniques that are used in nowadays object detection algorithms. The feature based methods, such is Viola-Jones, are described. SIFT, SURF, and ORB descriptors are also mentioned. The evaluation of approaches is based on the computation complexity as well as their detection performance. The last part of the report is dedicated to the smoke and fire detection, where the searching object shape and size is a priori unknown.

Contents

1 Introduction

The object detection is the essential task in the area of computer vision and image understandings. The main objective is to answer whether some object of a certain class is present in digital image or video. In recent years, there has been proposed a lot of algorithms that deal with this task [24, 6, 21].

1.1 Template Matching

The simplest approach how to detect a certain object in the image is to pass a small sliding window across the image and calculate the sum of square differences (SSD) or sum of absolute differences (SAD) between the image patch and the reference object. However, this simple approach has many disadvantages, e.g., the intra-class variance of searched object, the lack of robustness to lightning condition, planar rotation of the object, or perspective deformation.

1.2 Real-time processing

Beside the detection accuracy, one of the most important property of the detection algorithm is its computation costs. This becomes essential when the computation has to be performed on-line on the continuous video stream or when the detection system has to be deployed on low-powered mobile device, such as smart-phones.

1.3 Text structure

Section 2 brings the overview of feature-based object detection methods. The common attribute of these methods is that the object is detected from the features extracted from the input digital image rather than directly from pixel intensity values. The Viola-Jones algorithm extracts features from every pixel within the current search region using specific Haar wavelets. Positive response to the filter is given if the patch contains searched object.

In section 3, SIFT, SURF, and ORB descriptors are mentioned and subsection 3.4 describes k-d trees that are usually used for fast searching of nearest neighbour in high-dimensional descriptors space. Descriptor-based approaches are trying to find distinctive points within the image (key-points, interest points). From each point, a multi-dimensional descriptor is extracted. Descriptor provide information about the vicinity of the key-point independently on orientation, lighting conditions, and perspective deformation.

However, in some applications, e.g., smoke detection [9, 18], when the shape of the detecting object is unknown, a slightly different approach has to be applied. Section 4 brings overview of some methods that deals with this task.

2 Feature-based Object detection

In order to detect objects that vary in size and lighting conditions, object detection algorithm relying on features have to be taken into account. A lot of research has been dedicated to this particular area in recent years. This chapter brings an overview of some of the most successful approaches.

2.1 Viola-Jones

Maybe the most frequent algorithm, especially in the area of face detection, is the Viola-Jones [22] and its variations [10, 11]. It involves several techniques that are also common in recent feature detections algorithms.

The first notable property of the Viola-Jones algorithm is the usage of integral representation of the input image i. In integral image I, the value at any point (x, y) is the sum of all the pixels above and to the left of (x, y), inclusive:

$$I(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{1}$$

Haar wavelets, or more precisely Haar-like features (see Figure 1), are used for feature detection. They are square shaped functions that are convolved with the input patch. At given point (x, y), the response to the Haar-like function is simply the sum of pixel values covered with the white region minus the sum of image patch pixels covered with the grey region of the Haar feature. Integral image is used for fast computation.

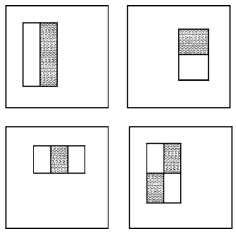

Figure 1: Haar-like features used in Viola-Jones algorithm [22]

Given that the base resolution of the detector patch is 24 by 24 pixels, the exhaustive set of rectangle Haar-like features is over 150 000. Therefore, a reasonable selection of the feature space subset has to be selected. Viola and Jones are applying the adapting boosting algorithm – AdaBoost. The basic idea of AdaBoost is that the complex (strong) classifier that decides whether the given image patch contains desired object or not can be made out of the weighted sum of weak classifiers.

The input for AdaBoost is the example pairs $(x_1, y_1), (x_1, y_1), \ldots, (x_n, y_n)$ where x_i is the input vector and y_i is its corresponding label ($y_i \in \{0, 1\}$). The initial point weights are set:

$$w_{1,i} = \frac{1}{m} \text{ for } y_i = 0 \tag{2}$$

$$w_{1,i} = \frac{1}{l} \text{ for } y_i = 1 \tag{3}$$

where m is the number of negatives and l is number of positives.

The following algorithm will create T weak classifiers h_t with corresponding weight α_t, such that the resulting strong classifier is:

$$\sum_{t=1}^{T} \alpha_t h(x) \tag{4}$$

For $t = 1, \ldots, T$:

1. Normalize the weights:

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{5}$$

2. Select the best weak classifier h_t with corresponding classification error ϵ_t

$$\epsilon_t = \sum_i w_i |h_t(x_i) - y_i| \tag{6}$$

3. Update the weights. This step will ensure that the misclassified points will obtain bigger weight for the next iteration:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \tag{7}$$

where $e_i = 0$ if the example is classified correctly, $e_i = 1$ otherwise, and

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t} \tag{8}$$

4. set $\alpha_t \leftarrow \log \frac{1}{\beta_t}$

In order to reduce the computation complexity, the cascade of classifiers is used instead. The sequence of classifiers is trained in such way that the initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation.

In recent years, various improvement to the original Viola-Jones algorithm. For example, in [10], a multi-view face detection algorithm is proposed. Instead of linear cascade of classifiers, a 3-level chain is proposed. Level 1 is a non-face rejecter, which could reject the non-face samples for all face views. Level 2 is view estimator and verifier. It has 2 sub levels to estimate the sample view from

coarse to fine. Level 3 is independent view verifier for each view. The sample can be determined as face area only if it can pass the verifier in level 3.

In [11], six different types of feature images rather than just one is used for feature extraction. Additionally, a key points based SVM predictor is implemented in the prediction phase to obtain the confidence of the detection result.

There are also proposed modification of boosting algorithm, like in [23, 7]

3 Descriptor-bases methods

3.1 Scale-invariant Feature Transform (SIFT)

The main disadvantage of Viola-Jones Algorithm is that the original algorithm was not able to detect objects rotated around axis perpendicular to the image plane. Lowe [12] proposed the algorithm that rely on key-points gained from the difference of Gaussian function applied in scale space to a series of smoothed and resampled images.

The difference of Gaussians (DoG) image of the original image I at (x, y) and scale σ is defined as:

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma), \tag{9}$$

where

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \tag{10}$$

is the convolution of image I with Gaussian function G.

The local extremes within the DoG image D are candidate key-points. From this candidate key-points, the points that are located on edges and within low-contrast areas are discarded.

After that, to each key-point, the orientation θ and magnitude m are assigned:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{11}$$

$$\theta(x, y) = \text{atan2}((L(x+1, y) - L(x-1, y), (L(x, y+1) - L(x, y-1)) \tag{12}$$

The resulting descriptor for each key-point is a feature vector created in the following way: The 4×4 sample regions rotated according to the key-point orientation θ is created around the key-point. Each subregion consist of 4×4 pixels. The histogram with 8 bins of intensity gradients is calculated within each subregion. This yields to the $4 \cdot 4 \cdot 8 = 128$ element feature vector. In order to decrease the effect of local lighting conditions, the feature vector is scaled to the unit length.

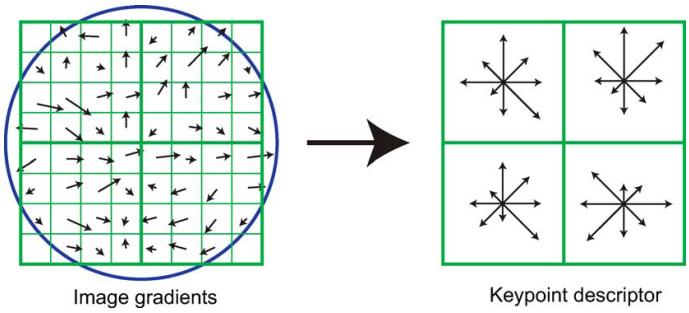

Figure 2: A key-point descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location (left). These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. [12]

The matching between descriptors obtained from the images from the training set and descriptors obtained from the current image is provided by approximate Best-Bin-First (BBF) algorithm, although other variations derived from the search within k-d tree may be also used [13].

SIFT imposes a large computational burden, especially for real-time systems such as visual odometry, or for low-power devices such as cellphones. There has also been research aimed at speeding up the computation of SIFT, most notably with GPU devices [17]. Another improvement has been reported in [8], where the dimensionality reduction of the resulting descriptor has been applied.

3.2 Speed-Up Robust Features (SURF)

Although the SIFT algorithm is able to detect descriptors in scale and rotation invariant manner, the computation complexity is much higher than with the Viola-Jones algorithm. The SURF algorithm [1] is able to locate high distinctive object descriptors with lower computation cost. The diagram of the computation of SURF descriptors is in Figure 3.

Figure 3: The flow diagram of descriptor computation with SURF algorithm.

Given a point $\mathbf{x} = (x, y)$ in an image I, the Hessian matrix $\mathcal{H}(\mathbf{x}, \sigma)$ in $\mathbf{x}$ at scale σ is defined as follows:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{13}$$

where $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second order derivative with the image I in point $\mathbf{x}$. In order to speed-up the calculation of the Hessian matrix, the approximation for second order Gaussian partial derivatives are used. See Figure 4.

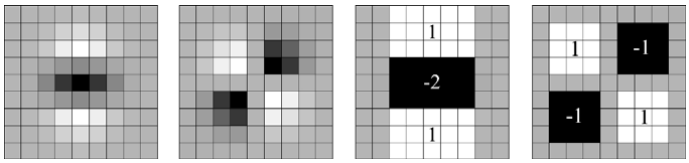

Figure 4: Left to right: The (discretised and cropped) Gaussian second order partial derivative in y (L_{yy}) and xy-direction (L_{xy}), respectively; The approximation for the second order Gaussian partial derivative in y (D_{yy}) and xy-direction (D_{xy}). The grey regions are equal to zero [1].

Additionally, rather than using a pyramid scale of images, the approximations of multiple-scaled Gaussian partial derivatives with single precomputed integral image is used. The points of interest are then located as the local maximums of the determinant of the Hessian matrix.

The assignment of orientation to the located points of interest is achieved by the dominant orientation of the Gaussian weighted Haar wavelet responses at every sample point within a circular neighbourhood. See Figure 5 for example of Haar wavelet filters and Figure 6 for the example of calculation of the dominant Haar wavelet response.

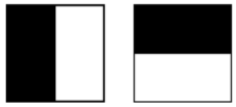

Figure 5: Haar wavelet filters that compute the responses in x (left) and y direction (right). The dark parts have weight -1 and the light parts +1 [1].

In some application, the calculation of the orientation is not necessary. The variation of the SURF algorithm without the orientation assignment of orientation is called upright SURF (U-SURF) [2].

In the following text, d_x is the Haar wavelet response in horizontal direction and d_y is the Haar wavelet response in vertical direction. The relatively rotated square region around key-point is divided into 4×4 square regions. In each subregion that consist of 5×5 pixels the Haar responses are calculated. The resulting descriptor is build from the sums of d_x and d_y. See Figure 7.

3.3 ORB

Rublee [16] proposed a very fast binary descriptor based on BRIEF [4], called ORB. For key-point detection, the FAST corner detector is used [15]. The BRIEF

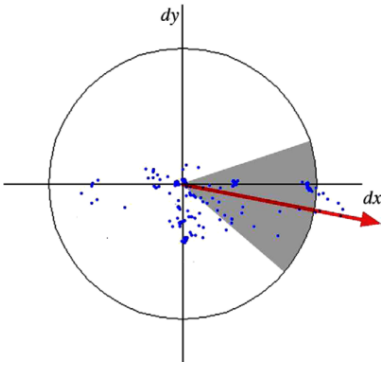

Figure 6: Orientation assignment: a sliding orientation window detects the dominant orientation of the Gaussian weighted Haar wavelet responses at every sample point within a circular neighbourhood around the interest point [1].

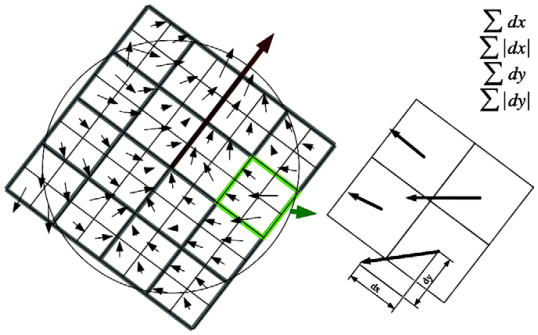

Figure 7: An oriented quadratic grid with 4×4 square sub-regions is laid over the interest point (left). For each square, the wavelet responses are computed from 5×5 samples (for illustrative purposes, only 2×2 sub-divisions are shown here). For each field, the sums dx, $|d_x|$; d_y, and $|d_y|$ are collected, computed relatively to the orientation of the grid (right) [1].

descriptor is a bit string description of an image patch constructed from a set of binary intensity tests τ of path $\mathbf{p}$ at point $\mathbf{x}$:

$$\tau(\mathbf{p}, \mathbf{x}, \mathbf{y}) = \begin{cases} 1 & : \quad \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & : \quad \mathbf{p}(\mathbf{x}) \geq \mathbf{p}(\mathbf{y}) \end{cases} \tag{14}$$

Choosing a set of n ($\mathbf{x}$,$\mathbf{y}$)-location pairs uniquely defines a set of binary tests. The n element feature vector is defined as a set of n binary tests:

$$f_n(\mathbf{p}) = \sum_{i=1}^{n} 2^{i-1} \tau(\mathbf{p}, \mathbf{x}_i, \mathbf{y}_i) \tag{15}$$

The set of binary tests defines $2 \times n$ matrix $\mathbf{S}$:

$$\mathbf{S} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & x_n \end{pmatrix} \tag{16}$$

The rotation θ and corresponding rotation matrix $\mathbf{R}_\theta$ of the corner detected by FAST is measured using the intensity centroid [14]. The steered version of matrix $\mathbf{S}$ is:

$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S} \tag{17}$$

and the steered BRIEF operator g_n becomes:

$$g_n(\mathbf{p}, \theta) = f_n(\mathbf{p}) | (\mathbf{x}_i, \mathbf{y}_i) \in \mathbf{S}_\theta \tag{18}$$

The Figure 8 shows the performance comparisons of ORB, SIFT, and SURF in synthetic test with added Gaussian noise and planar rotation. For each reference image, the FAST key-points and steered BRIEF features (rBRIEF), targeting 500 key-points per image, were calculated. The results are given in terms of the percentage of correct matches, against the angle of rotation.

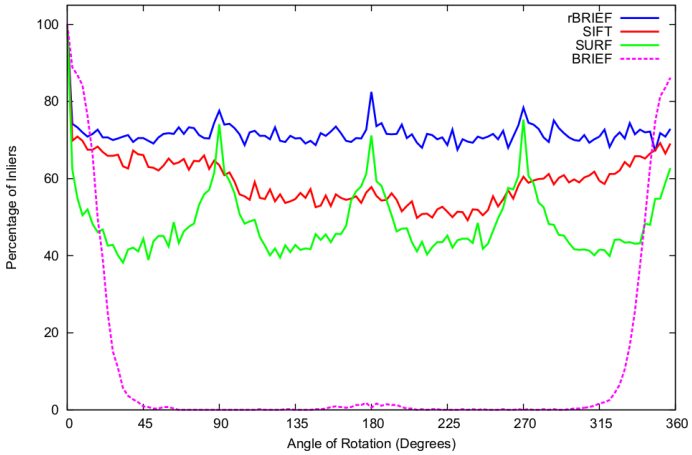

Figure 8: Matching performance of SIFT, SURF, BRIEF with FAST, and ORB under synthetic rotations with Gaussian noise of 10 [16].

3.4 Descriptors Matching

The final step in object detection, when the descriptors are extracted, is to match the descriptors from the input image to the descriptors stored in the database and decide if the image patch contains object of a certain class. Since this is very computation expensive, especially within high-dimensional space, k-d trees are used for this purpose. The k-d tree is a binary tree in which every node is a k-dimensional point. Every non-leaf node splits hyperplane that divides

the space into two parts. Every node in the tree is associated with one of the k-dimensions, with the hyperplane perpendicular to that dimension's axis. If, for example, for a particular split the x axis is chosen, all points in the left subtree will have a smaller x and contrary, all points in the right subtree will have a larger value of x [3].

The search within k-d tree has only the $O(\log N)$ complexity. Muja [13] pointed out that the search performance rapidly decreased in high-dimensional space. Original k-d tree algorithm splits the data in half at each level of the tree on the dimension for which the data exhibits the greatest variance. The randomized trees are built by choosing the split dimension randomly from the first D (usually $D = 5$) dimensions on which data has the greatest variance.

The second option – hierarchical k-means tree – constructs the tree by splitting the data points at each level into K distinct regions using a k-means clustering, and then applying the same method recursively to the points in each region. If the number of points in a region is smaller than K, recursion stops.

Based on the data structure and desired precision, Muja also proposed the fast approximate nearest neighbour search that selects optimal k-d tree modification and corresponding search algorithm.

The descriptor matching in [12] uses the nearest candidate match from the database and the second nearest point from the database that belongs to the different object. By comparing the ratio between these two points, we are able to predict the point discriminative ability. If the distance ratio between the nearest match and the second nearest point is greater than 0.8, the point is rejected. Another feature selection approach may be find in [20].

In [19] has been proposed that the usage of principal components of the SIFT descriptors may be suitable as the binning criteria. The PCA projection of the query input point is the key for specific part of the database (bin), where the search is performed.

4 Fire and Smoke Detection

In open space area, where the usual application of detectors searching for microscopic particles of smoke in the air is impossible. However, the usual application of feature-based object detection method can't be applied on smoke and fire detection, because we can't expect a specific shape with distinctive edges.

In [5], the fire detection algorithm relies on the classification of the values of the image that has been transformed to the YCbCr color model. Despite its simplicity, achieved detection rate is 99% with very low false alarm rate.

The detection of smoke of an uncontrolled fire can be easily observed by a camera even if the flames are not visible. The main idea of smoke detection algorithm proposed in [18] is that the smoke gradually smooth the edges in an image. Edges in an image correspond to local extrema in wavelet domain. Gradual decrease in their sharpness result in a decrease in the values of these extrema. However, these extrema values corresponding to edges, do not boil down to zero when there is smoke. In fact, they simply loose some of their energy

but they still stay in their original locations, occluded partially by the semi-transparent smoke. The detection algorithm is divided into following stages:

1. Recursive estimation of moving pixels:

$$B_{n+1}(\mathbf{x}) = \begin{cases} \alpha\ B_n(\mathbf{x}) + (1-\alpha) I_n(\mathbf{x}) & \text{when } \mathbf{x} \text{ is not moving} \\ B_n(\mathbf{x}) & \text{when } \mathbf{x} \text{ is moving} \end{cases} \tag{19}$$

$$B_0(\mathbf{x}) = I_0(\mathbf{x}) \tag{20}$$

   where $I_n(\mathbf{x})$ is the pixel intensity in n^{th} frame and α is a time constant that specifies how fast new information supplants old observations. Moving pixels are determined by subtracting the current image from the background image B and thresholding.

2. Decrease in high frequency content corresponding to edges in moving regions are checked using spatial wavelet transform.

3. The decrease in chromatic channels of YCbCr color model is detected for areas from the previous step.

4. The flicker in smoke is also used as an additional information. The candidate regions are checked whether they continuously appear and disappear over time.

5. At the last step, the convexity in the shape of the smoke regions is checked. Smoke of an uncontrolled fire expands in time which results in regions with convex boundaries.

5 Conclusion

In this text, several object detection approaches were proposed. Huge amount of research interest have been dedicated to this area in last years. Nowadays algorithms are able to deal with partial object covering, multiple views, and scale. However, each algorithm is usually suitable for a specific purpose only. Specific requirements and conditions, under which the object detection system will be employed, should be always considered.

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[2] Herbert Bay, Beat Fasel, and Luc Van Gool. Interactive Museum Guide : Fast and Robust Recognition of Museum Objects. In *First International Workshop on Mobile Vision (IMV 2006)*, page 15, 2006.

[3] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.

[4] Michael Calonder, Vincent Lepetit, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. *Lecture Notes in Computer Science*, 6314:778–792, 2010.

[5] Turgay Çelik and Hasan Demirel. Fire detection in video sequences using a generic color model. *Fire Safety Journal*, 44(2):147–158, February 2009.

[6] Vidit Jain and Erik Learned-miller. FDDB: A Benchmark for Face Detection in Unconstrained Settings. Technical report, 2010.

[7] Roman Juránek, Pavel Zemčík, and Michal Hradiš. Real-Time Algorithms of Object Detection Using Classifiers. In *Real-Time Systems, Architecture, Scheduling, and Application*, pages 227–248. 2009.

[8] Yan Ke and Rahul Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 506–513, 2004.

[9] DongKeun Kim and Yuan-Fang Wang. Smoke Detection in Video. In *WRI World Congress on Computer Science and Information Engineering*, pages 759–763. Ieee, 2009.

[10] Jung-Bae Kim, Haibing Ren, and SeongDeok Lee. Multi-view Face Detection Using Multi-layer Chained Structure. In Frederic Truchetet and Olivier Laligant, editors, *Intelligent Robots and Computer Vision XXVI: Algorithms and Techniques*, volume 7252, page 8, January 2009.

[11] Qian Li, Usman Niaz, Bernard Merialdo, and Sophia Antipolis. An Improved Algorithm on Viola-Jones Object Detector. In *10th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, 2012.

[12] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[13] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP International Conference on Computer Vision Theory and Applications*, pages 331—-340, 2009.

[14] P. L. Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291–307, 1999.

[15] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: a machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010.

[16] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision*, pages 2564–2571. Ieee, November 2011.

[17] Sudipta N. Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc. GPU-based Video Feature Tracking And Matching. Technical Report May, 2006.

[18] B. Ugur Töreyin and A. Enis Çetin. Wavelet Based Real-time Smoke Detection in Video. In *Proceedings of the 2005 European Signal Processings Conference*, page 4, 2005.

[19] Geoffrey Treen and Anthony Whitehead. A PCA-Based Binning Approach for Matching to Large SIFT Database. In *Canadian Conference on Computer and Robot Vision*, pages 9–16. Ieee, 2010.

[20] Panu Turcot and David G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2109–2116. Ieee, September 2009.

[21] Tinne Tuytelaars and Krystian Mikolajczyk. Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2007.

[22] Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[23] Jan Šochman and Jiří Matas. WaldBoost - Learning for Time Constrained Sequential Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 150–156. Ieee, 2005.

[24] Cha Zhang, Zhengyou Zhang, and Technical Report. A Survey of Recent Advances in Face Detection. Technical Report June, 2010.