Motion and Object Detecion

Technical Report - FIT - VG20102015006 - 2013 - 04

ing. Štěpán Mráček

Faculty of Information Technology, Brno University of Technology

January 5, 2014

Abstract

This report brings an overview of motion and object detection techniques with respect to the security purposes - especially pedestrian tracking and gun detection. The brief description of motion detection as well as object detection techniques is given. The histogram of oriented gradient pedestrian detection and subsequent experimental detection of handguns based on template matching and optical flow tracking is also presented.

Contents

1 Introduction

Motion detection techniques as well as object detection algorithms are described in this report. The basic principles and techniques used in the area of motion detection are described in Section 2. Section 3 brings the overview of template-matching detection as well as the detection based on Haar-like features, descriptors and histogram of oriented gradients.

2 Motion Detection

The main objective of motion detection algorithms is to mark video frame regions that contains motion. They can also estimate the direction and magnitude of movement.

2.1 Background subtraction

Perhaps the most direct approach is the background subtraction technique. The detector compares the input frame with the average of previous n frames. The motion M_t at time t is detected if there is a difference greater than some defined threshold t on some coordinates (x, y) between the image I_t and average of previous frames:

$$M_t(x,y) = \begin{cases} 1 & \text{if} \quad \left| I_t(x,y) - \frac{\sum_{i=1}^{n} I_{t-i}(x,y)}{n} \right| > t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

This simple approach may fail if there is only a small pixel intensity difference between moving object and background. This may be solved with the employment of the edge detector, e.g., Canny edge detector [5] or Sobel operator [7]. The current frame I_t as well as the history $I_{t-1}, \ldots, I_{t-n}$ is processed with the edge detector first.

The more complex approach to the background subtraction is shown in [24]. Zivkovic presents adaptive algorithm using Gaussian mixture probability density. Recursive equations are used to constantly update the parameters and also to simultaneously select the appropriate number of components for each pixel.

2.2 Optical flow

The goal of the Pyramid Lukas-Kanade tracking [3] is to find for given point $\mathbf{u} = (u_x, u_y)$ in the image I corresponding point $\mathbf{v} = \mathbf{u} + \mathbf{d}$ in the consecutive image J. There is an assumption that the neighbourhood of the point $\mathbf{u}$ is similar to the neighbourhood of the point $\mathbf{v}$. The vector $\mathbf{d}$ is referred as the image velocity or the optical flow at $\mathbf{u}$. It is defined as the vector that minimizes the function:

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \\ \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I(x,y) - J(x+d_x, y+d_y)\right)^2 \tag{2}$$

The w_x and w_y define integration (neighbourhood) window of size $(2w_x + 1) \times (2w_y + 1)$. Their typical values for are 2,3,4,5,6,7 and usually $w = w_x = w_y$.

There is a trade-off between local accuracy and robustness when choosing the integration window size. Smaller window size is able to capture even tiny motion. On the other hand, the tracking is lost when the movement exceeds the search window. Contrary, the large window size lacks accuracy. Therefore, an involvement of the image pyramid representation has been proposed.

The image pyramid consists of the original image I^0 of size $n_x^0 \times n_y^0$ and L_m levels – usually 3. Each image I^L at level $L = 1, 2, \ldots, L_m$ is resized, such its size is recursively set to $\frac{1}{2}n_x^{L-1} \times \frac{1}{2}n_y^{L-1}$. In the image pyramid perspective, the point $\mathbf{u}^L$is computed from the point in base image $\mathbf{u} = \mathbf{u}^0$:

$$\mathbf{u}^L = \frac{\mathbf{u}}{2^L} \tag{3}$$

The pyramidal tracking starts at the highest level image L_m. This will produce the optical flow estimation guess $\mathbf{g}^{L_m - 1}$ that is propagated to to the next pyramid level L_{m-1} where the more correct flow estimation is computed. This is repeated until the base pyramid level is reached. The general equation for estimating the optical flow at level L is:

$$\begin{gathered}\epsilon(\mathbf{d}^L) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \\ \left(I^L(x,y) - J^L(x + g_x^L + d_x^L, y + g_y^L + d_y^L)\right)^2\end{gathered} \tag{4}$$

The initial guess $\mathbf{g}^{L_m} = (0, 0)$ and the propagation of the guess from level L to $L - 1$ is accomplished by $\mathbf{g}^{L-1} = 2(\mathbf{g}^L + \mathbf{d}^L)$. The final optical flow $\mathbf{d}$ is calculated from partial flow estimation at the base level and the corresponding guess:

$$\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0 \tag{5}$$

Although the optical flow is usually used for tracking selected point(s) of interest it may be also used for motion detection. The uniform grid of points is established in image I. Each point is tracked in the next frame J subsequently. If the flow $\mathbf{d}$ of some point exceeds defined threshold, motion is detected (see Figure 1).

Figure 1: Optical flow

3 Object Detection

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 [23, 8, 19].

3.1 Template Matching

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

There are several possibilities how to measure distance R between the input image I and the template T:

- Square difference:

$$R(x,y) = \sum_{x',y'} \left(T(x',y') - I(x+x',y+y')\right)^2 \tag{6}$$

- Normalized square difference ensuring that the values are from range $\langle 0,1 \rangle$:

$$R(x,y) = \frac{\sum_{x',y'} \left(T(x',y') - I(x+x',y+y')\right)^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \tag{7}$$

- Correlation:

$$R(x,y) = \sum_{x',y'} \left(T(x',y') \cdot I(x+x',y+y')\right)^2 \tag{8}$$

- Normalized correlation:

$$R(x,y) = \frac{\sum_{x',y'} \left(T(x',y') \cdot I(x+x',y+y')\right)^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}} \tag{9}$$

- Correlation coefficient:

$$R(x,y) = \sum_{x',y'} \left(T'(x',y') \cdot I'(x+x',y+y')\right)^2 \tag{10}$$

  where $T'(x',y') = T(x',y') - 1/(T_w \cdot T_h) \cdot \sum_{x'',y''} T(x'',y'')$

- Normalized correlation coefficient:

$$R(x,y) = \frac{\sum_{x',y'} \left(T'(x',y') \cdot I'(x+x',y+y')\right)^2}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \tag{11}$$

3.2 Haar-based detection

Maybe the most frequent algorithm, especially in the area of face detection, is the Viola-Jones [21] and its variations [11, 12]. 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:

$$I(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{12}$$

Haar wavelets, or more precisely Haar-like features (see Figure 2), 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.

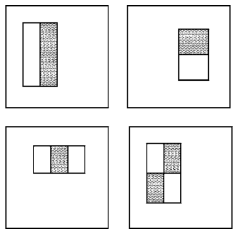

Figure 2: Some Haar-like features used in Viola-Jones algorithm [21]

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:

$$w_{1,i} = \frac{1}{m} \text{ for } y_i = 0 \tag{13}$$

$$w_{1,i} = \frac{1}{l} \text{ for } y_i = 1 \tag{14}$$

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:

$$\sum_{t=1}^{T} \alpha_t h(x) \tag{15}$$

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

1. Normalize the weights:

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{16}$$

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

$$\epsilon_t = \sum_i w_i |h_t(x_i) - y_i| \tag{17}$$

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

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \tag{18}$$

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

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t} \tag{19}$$

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 [11], 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 [12], 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 [22, 9]

3.2.1 Detecting objects in video sequence

The original algorithm for detection objects of interest in static image is extended to the video sequence in [20]. Viola-Jones features operate on the differences between pairs of images I_t and I_{t+1} in time. Motion filters operate on 5 images:

$$\begin{aligned} \Delta &= |I_t - I_{t+1}| \\ U &= |I_t - I_{t+1} \uparrow| \\ D &= |I_t - I_{t+1} \downarrow| \\ L &= |I_t - I_{t+1} \leftarrow| \\ R &= |I_t - I_{t+1} \rightarrow| \end{aligned} \tag{20}$$

where $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ are image shift operators, e.g., $I_{t+1} \uparrow$ is I_{t+1} shifted up by one pixel. See Figure 3.

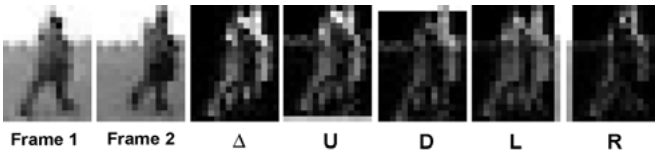

Figure 3: Various shifted difference images used in [20]. The first two images are two frames with a low resolution pedestrian pictured. The following images show the Δ, U, D, L, and R images described in the text.

The input of AdaBoost is the set F of the following filters:

- First type of filters compares sums of absolute differences between Δ and one of $\{U, D, L, R\}$:

  $$f_i = r_i(\Delta) - r_i(S) \tag{21}$$

  where $r_i()$ is a sum of intensities within detection window and S is one of $\{U, D, L, R\}$.

- Second type of filters compares sums within the same image:

  $$f_j = \phi(S) \quad \text{or} \quad f_j = \phi(I_t) \tag{22}$$

  where $\phi()$ is the response to one of the wavelets similar to those in Figure 2.

- Third type of filter measures magnitude of motion in one of the motion images S:

$$f_k = r_k(S) \tag{23}$$

The AdaBoost training of simple classifiers into the cascade is similar to the static variation of Viola-Jones algorithm. In each round the learning algorithm chooses from a heterogeneous set of filters. The output of the AdaBoost learning algorithm is a classifier that consists of a linear combination of the selected features. Each classifier in the cascade is trained to achieve very high detection rates, and modest false positive rates. Simpler detectors (with a small number of features) are placed earlier in the cascade, while complex detectors (with a large number of features are placed later in the cascade).

3.3 Descriptor-based detection

Detectors based on descriptor features are trying to find distinctive points within the image (key-points, interest points). From each key-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.

3.3.1 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 [13] 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:

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma), \tag{24}$$

where

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \tag{25}$$

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:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{26}$$

$$\theta(x, y) = \text{atan2}((L(x+1, y) - L(x-1, y), (L(x, y+1) - L(x, y-1)) \quad (27)$$

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.

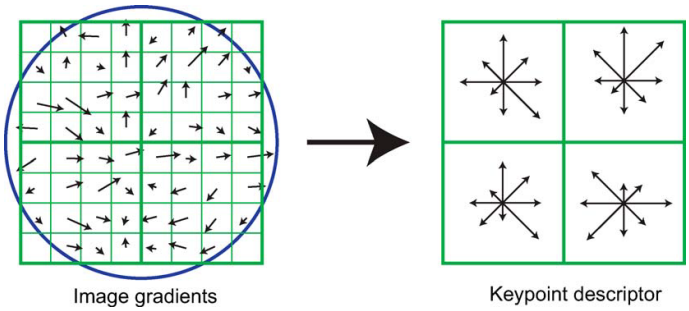

Figure 4: A key-point descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the key-point location (left). These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation 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. [13]

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 [14].

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 [18]. Another improvement has been reported in [10], where the dimensionality reduction of the resulting descriptor has been applied.

3.3.2 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 5.

Figure 5: 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:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{28}$$

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 6.

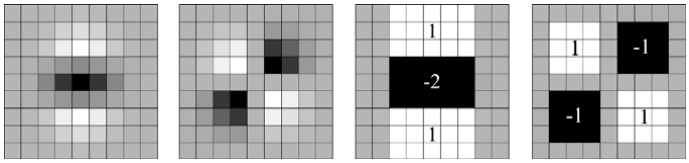

Figure 6: Left to right: The (discretised and cropped) Gaussian second order partial derivative in y (L_{yy}) and xy-direction (L_{xy}), respectively; The approximation for the second order Gaussian partial derivative in y (D_{yy}) and xy-direction (D_{xy}). The grey regions are equal to zero [1].

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 7 for example of Haar wavelet filters and Figure 8 for the example of calculation of the dominant Haar wavelet response.

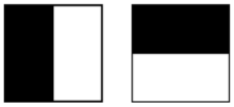

Figure 7: Haar wavelet filters that compute the responses in x (left) and y direction (right). The dark parts have weight -1 and the light parts +1 [1].

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].

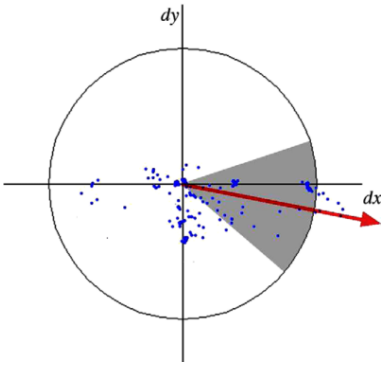

Figure 8: Orientation assignment: a sliding orientation window detects the dominant orientation of the Gaussian weighted Haar wavelet responses at every sample point within a circular neighbourhood around the interest point [1].

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 9.

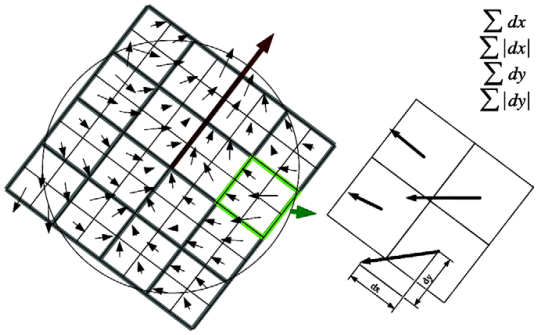

Figure 9: An oriented quadratic grid with 4×4 square sub-regions is laid over the interest point (left). For each square, the wavelet responses are computed from 5×5 samples (for illustrative purposes, only 2×2 sub-divisions are shown here). For each field, the sums dx, $|d_x|$; d_y, and $|d_y|$ are collected, computed relatively to the orientation of the grid (right) [1].

3.3.3 ORB

Rublee [17] proposed a very fast binary descriptor based on BRIEF [4], called ORB. For key-point detection, the FAST corner detector is used [16]. The BRIEF 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 $\mathbf{x}$:

$$\tau(\mathbf{p}, \mathbf{x}, \mathbf{y}) = \begin{cases} 1 & : \quad \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & : \quad \mathbf{p}(\mathbf{x}) \geq \mathbf{p}(\mathbf{y}) \end{cases} \tag{29}$$

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:

$$f_n(\mathbf{p}) = \sum_{i=1}^{n} 2^{i-1} \tau(\mathbf{p}, \mathbf{x}_i, \mathbf{y}_i) \tag{30}$$

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

$$\mathbf{S} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & x_n \end{pmatrix} \tag{31}$$

The rotation θ and corresponding rotation matrix $\mathbf{R}_\theta$ of the corner detected by FAST is measured using the intensity centroid [15]. The steered version of matrix $\mathbf{S}$ is:

$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S} \tag{32}$$

and the steered BRIEF operator g_n becomes:

$$g_n(\mathbf{p}, \theta) = f_n(\mathbf{p}) | (\mathbf{x}_i, \mathbf{y}_i) \in \mathbf{S}_\theta \tag{33}$$

3.3.4 Histogram of Oriented Gradients

Histogram of oriented gradients (HOG) are feature descriptors reminiscent to edge orientation histograms, SIFT descriptors and shape contexts, but they are computed on a dense grid of uniformly spaced cells and they use overlapping local contrast normalizations for improved performance [6]. Object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The implementation of these descriptors can be achieved by dividing the image into small connected regions, called cells, and for each cell compiling a histogram of gradient directions or edge orientations for the pixels within the cell.

The HOG computation can be divided to the following steps:

1. **Color and gamma normalization**: In fact, this step can be omitted in HOG descriptor computation, as the ensuing descriptor normalization essentially achieves the same result.

2. **Gradient computation**: the computation of the gradient values is achieved by applying the 1-D centered, point discrete derivative mask in one or both of the horizontal and vertical directions. In other words, the color or intensity data of the image is convolved with the following filter kernels:

$$[1, 0, -1] \text{ and } [1, 0, -1]^T \tag{34}$$

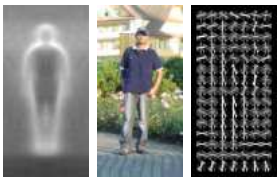

Figure 10: HOG computation. From left to right: Mean image of computed gradients from the training set; input probe image; calculated HOG descriptor. [6]

3. **Histogram binning**: The next step involves creating the cell histograms. Each pixel within the cell casts a vote for the resulting orientation histogram. The histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is ,,unsigned" or ,,signed". Unsigned gradients used in conjunction with 9 histogram channels performed best in their human detection experiments [6].

4. **Descriptor calculation**: In order to account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially connected blocks. The HOG descriptor is then the vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor.

5. **Histogram normalization**: Let v be the non-normalized vector containing all histograms in a given block, $\|v\|$ is the Euclidean norm and e be some small constant. The L2-norm is then:

$$f = \frac{v}{\sqrt{\|v\|^2 + e^2}} \tag{35}$$

6. **SVM classification** The final step is to feed the descriptors into some recognition system based on supervised learning. Support Vector Machine (SVM) classifier is used for this purpose. Once trained on images containing some particular object, the SVM classifier can make decisions regarding the presence of an object, such as a human being, in additional test images.

4 Experiments

The experiments were conducted on the dataset consisting of 26 individual video files. First ten videos were captured at Faculty of Information Technology, Brno University of Technology[1]. Individual recordings contains persons walking in the

[1] http://www.fit.vutbr.cz/.en

hallways, in front of the building and coming out from the building. Some of them are equipped with gun replicas (knifes, handguns and rifles).

Another 16 videos were captured at University of Defence, Brno in cooperation with Department of Weapons and Ammunition at University of Defence[2]. Several scenarios were used when the videos were captured. In each shot, some subjects are equipped with weapons. Civilians are also present beside soldiers. The following scenarios were captured:

- Individuals walk in a corridor. There is approximately 2 meters gap between them.
- Subjects walk in a corridor in groups consisting of 2 to 4 persons.
- Subjects walk the stairs.
- Subjects walk in front of the building as individuals and in small groups.
- Subjects walk in front of the building towards the camera.

4.1 Motion Detection

The simple motion detection employing the background subtraction technique (see section 2.1) was evaluated first. The result of motion detection is in Figure 11. The main problem of the simple background extraction approach is the requirement of establishing proper decision threshold. In the worst case, the threshold should be manually set for each video.

Figure 11: Masked areas of video frame where the motion using background subtraction technique was detected

The disadvantage of the user-defined threshold may be solved with the involvement of the edge detector algorithm. Our experiments shown that the best results were achieved with the Sobel operator [7]. The process of background-subtraction-based motion detection using Sobel oparator is in Figure 12.

[2]http://www.unob.cz/en/fmt/structure/k201/Pages/default.aspx

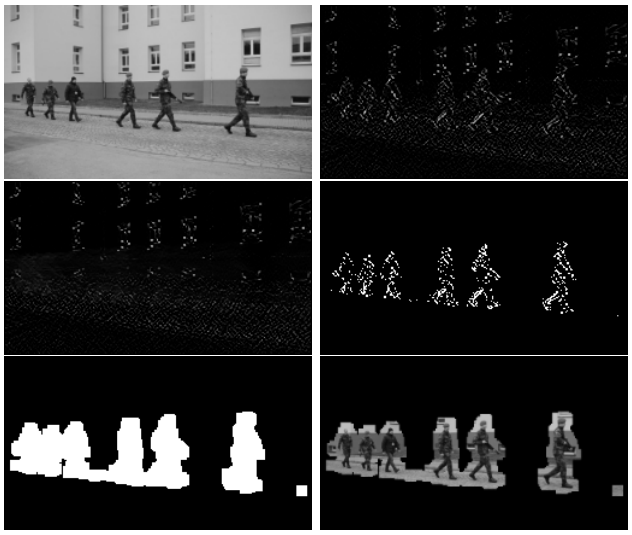

Figure 12: Motion detection using background subtraction applied on individual frames with Sobel Operator. From left to right: input frame I_t, applied Sobel operator, mean of frames $I_{t-1}, \ldots, I_{t-n}$, difference between I_t and the mean, morphological closure applied on the difference image, masked regions containing movement.

It has emerged that the motion detection based on optical flow estimation brings only poor results. However, the optical flow is a good choice for tracking points of interests in the video. See Figure 13.

4.2 Gun Detection and Tracking

Our next experiment involves utilisation of several aforementioned algorithms. We are using HOG-based pedestrian detection (see Section 3.3.4) that marks regions containing persons - either solders ans civilians.

Template matching (Section 3.1) within the regions selected by the HOG detector is used for weapon detection and localisation. Due to the very big intra-class variability of various guns, template matcher is searching for specific hand gesture when the subject is holding a gun – see Figure 14.

We are using normalized correlation coefficient metric (Equation 11) for measuring the distance between template and candidate image patch. In order to reduce false alarm rate, the required correlation coefficient threshold has been set to 0.9. However, this yields to lower detection rates – the gun-holding gesture is not detected in all frames. Therefore, when the gun is not detected at point where it was in last frame, optical flow (Section 2.2) is used for estimation of the gun movement.

Some results from the gun detection and tracking expriments are in Figures 15, 16, and 17. Subjects detected with pedestrian HOG detector are marked

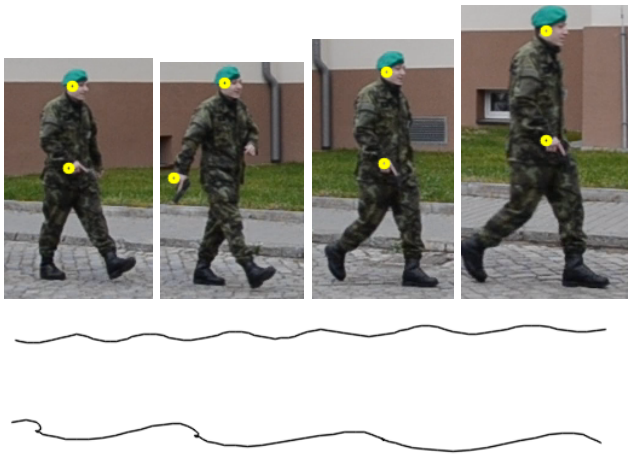

Figure 13: Tracking of selected points of interest and their trajectory.

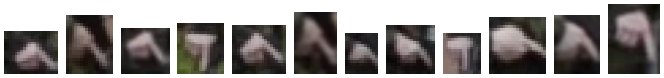

Figure 14: Templates used for detection of gun-holding gesture.

with green rectangle. The red circle denotes a point where the template matcher located hand-holding gesture. The small yellow circle shows the result of tracking a point where the gun was previously detected.

5 Conclusion

This report briefly introduced a key concepts of motion and object detection. We have evaluated some of described algorithms on our testing dataset. The motion detection based on background subtraction as well as the enhancements utilizing Sobel operator or mixture of Gaussians were tested.

The pedestrian detection based on the histogram of oriented gradients and subsequent SVM classifier were used in conjunction with template matcher and Lukas-Kanade optical flow estimator for gun detection and tracking.

Probably the most weak part of the gun detection and tracking pipeline is the template matcher. Although our experiments show that the correlation coefficient (Equation 11) outperforms other similarity metrics (see Equations 6, 7, 8, 9, and 10) used in template matching, the false alarms as well as detection misses are still present.

In further works, detection relying on descriptors (see Section 3.3) will be evaluated.

Figure 15: Although the subject on the right side of the image was not detected, previously detected gun is correctly tracked.

Figure 16: The gun of the soldier on the left hand side of the image was previously detected. The civilian has no gun.

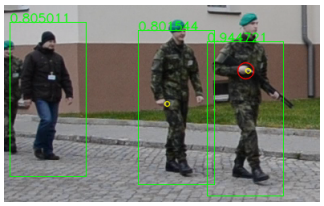

Figure 17: False detection of hand-holding gesture in one of previous frames for soldier in the center of the image.

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] Jean-yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker. Technical report, Intel Corporation, 2001.

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

[5] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[6] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[7] K. Engel. Sobel operator. In *Real-time volume graphics*, pages 112–114. 2006.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] M. Muja and D. G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

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

[16] 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.

[17] 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.

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

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

[20] Paul Viola, M. Jones, and D. Snow. Detecting Pedestrians Using Patterns of Motion and Appearance. Technical report, 2003.

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

[22] 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.

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

[24] Zoran Zivkovic. Improved Adaptive Gaussian Mixture Model for Background Subtraction. *Proceedings of the 17th International Conference on Pattern Recognition*, 2:4, 2004.