VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HUMAN ACTION RECOGNITION IN VIDEO

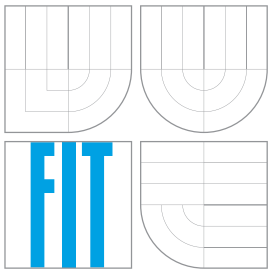
DISERTAČNÍ PRÁCE
PH.D. THESIS

AUTOR PRÁCE
AUTHOR

Ing. IVO ŘEZNÍČEK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁNÍ ČINNOSTÍ ČLOVĚKA VE VIDEU

HUMAN ACTION RECOGNITION IN VIDEO

DISERTAČNÍ PRÁCE
PH.D. THESIS

AUTOR PRÁCE
AUTHOR

Ing. IVO ŘEZNÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2014

Abstrakt

Tato disertační práce se zabývá vylepšením systémů pro rozpoznávání činností člověka. Současný stav vědění v této oblasti jest prezentován. Toto zahrnuje způsoby získávání digitálních obrazů a videí společně se způsoby reprezentace těchto entit za použití počítače. Dále jest prezentováno jak jsou použity extraktory příznakových vektorů a extraktory prostorově-časových příznakových vektorů a způsoby přípravy těchto dat pro další zpracování. Příkladem následného zpracování jsou klasifikační metody. Pro zpracování se obecně obvykle používají části videa s proměnlivou délkou. Hlavní přínos této práce je vyřčená hypotéza o optimální délce analýzy video sekvence, kdy kvalita řešení je porovnatelná s řešením bez restrikce délky videosekvence. Algoritmus pro ověření této hypotézy jest navržen, implementován a otestován. Hypotéza byla experimentálně ověřena za použití tohoto algoritmu. Při hledání optimální délky bylo též dosaženo jistého zlepšení kvality klasifikace. Experimenty, výsledky a budoucí využití této práce jsou taktéž prezentovány.

Klíčová slova

Optimální analyzovaná délka akce, lokální video příznaly, bag-of-words representace videa, visuální slovník, SVM.

Abstract

This thesis focuses on the improvement of human action recognition systems. It reviews the state-of-the-art in the field of action recognition from video. It describes techniques of digital image and video capture, and explains computer representations of image and video. This thesis further describes how local feature vectors and local space-time feature vectors are used, and how captured data is prepared for further analysis, such as classification methods. This is typically done with video segments of arbitrarily varying length. The key contribution of this work explores the hypothesis that the analysis of different types of actions requires different segment lenghts to achieve optimal quality of recognition. An algorithm to find these optimal lengths is proposed, implemented, and tested. Using this algorithm, the hypothesis was experimentally proven. It was also shown that by finding the optimal length, the prediction and classification power of current algorithms is improved upon. Supporting experiments, results, and proposed exploitations of these findings are presented.

Keywords

Optimal analysis length of action, local space-time features, bag-of-words repesentation, visual vocabulary, SVM.

Citace

Ivo Řezníček: Human action recognition in video, disertační práce, Brno, FIT VUT v Brně, 2014

Human action recognition in video

Prohlášení

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně pod vedením pana prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ivo Řezníček
June 23, 2014

Poděkování

Chtěl bych poděkovati všem, kteří se jakkoliv podíleli na vzniku této práce, zejména však prof. Dr. Ing. Pavlovi Zemčíkovi za jeho velkou trpělivost při přípravě finální podoby práce, kterou držíte nyní v rukou. Současně je též nutno mu poděkovati za pomoc při přípravě všech předchozích prací bez nichž by tato práce nemohla vzniknouti.

Contents

Chapter 1

Introduction

Nowadays the world is heading towards to a total monitoring of everything what is going on. That induces a creation of systems for detection, let us say, anomalies that are physically happening around us. One of the needs is, for example, the monitoring of some public areas for violent behavior detection. This can be achieved by a number of humans that are located at monitored places and are able to prevent other beings from performing such activities. This solution is very expensive in terms of a huge number of humans that are performing the monitoring, and all of them are potentially in danger on a place.

Contemporary techniques give an opportunity to adapt above presented procedure and deploy video cameras to the places where monitoring needs to be performed, and the humans are observing the situation remotely. This solution reduces the number of observers on place, and the safety of such humans is higher than before. Such monitoring procedure is often called as surveillance.

As time has gone by the cameras' quality is higher and higher, the output signal can be digitized, and the possibility of processing of the camera signal by computers comes forward. Such processing can be split into two main areas; it is (1) still image processing, detection and localization of humans, faces, animals, or general objects and (2) video processing, such as behavior detection in front of the camera, car accident detection, smoke detection, fight detection, etc. All those detections may be very valuable for the surveillance systems. It is highly needed to remove the human factor from surveillance application and let the computer detect all dangerous or unwanted situations that may happen but that situation did not come up yet.

Nowadays such computer detection techniques are being used as a support of the surveillance system's operator and he/she finally decides whether the detection is valid and further reactions need to be performed.

This work focuses on the human action recognition from digitized video streams but current research in such field generally approaches the problem in a way, that some new algorithm is proposed and afterwards tested on the standardized sets of video sequences and the solution's quality is then measured. All sets of video sequences generally contain totally variable video streams. The main problems can be seen (i) as the variable length of samples, (ii) on the fact that samples contain an amount of other actions and somewhere at the end or inside of the sample the wanted action is performed, (iii) the presented behavior is fully or partially done outside of video frames. Majority of researches use the whole example as one entity and above presented facts are simply not taken into account.

The main question remains; it is whether some, let us say, optimal analysis length of action can be found and consequently whether on-line detection system, which processes

only such restricted length of the video, can be built and whether that system produces comparable results to the currently well known solutions.

In this thesis the action recognition solutions were investigated and the technique which leads to the state-of-the-art performance were replicated. The whole video processing may be understood as a series of transformation which are described in subsequent chapters. In Chapter 2 the image and video representations in a computer are discussed. Chapter 3 presents some possible techniques of a representation of image or video contents in a computer, and Chapter 4 summarizes the pipeline and its components that may be used for action recognition processing and which as well as reaches the state-of-the-art performance. Chapter 5 presents the definition of the *optimal* analysis length and the algorithm for obtaining of that property of human action recognition processing. The conclusion of the work is presented in Chapter 6.

Chapter 2

Image representation

Human being is able to perceive the outer environment/world as a three dimensional vast space with a special equipment which is called the eyes. In the following sections, the image function is mathematically defined including the codomain of the image function with possibilities of transformation of the codomain values for further processing. In the last section the mathematical background of the domain digitizing and of the codomain of analog image function is presented.

2.1 Definition of the image function

The outer world can be mathematically described as a three dimensional orthogonal space [78, 18] with three axis x, y, z as shown in Figure 2.1. When the observation of outer environment is wanted, some specialized sensor, which is able to retrieve similar information about the neighborhood as the human eye does, needs to be used. The origin of the orthogonal space (it is the point [0, 0, 0] in Figure 2.1) needs to be specified in one point of the outer world, and since this has been done, the wanted location of the sensor can be parametrized as a position within our model as $[x_l, y_l, z_l]$ as shown in Figure 2.1, where $x_l \in (-\infty, +\infty)$, $y_l \in (-\infty, +\infty)$ and $z_l \in (-\infty, +\infty)$. This point describes precisely the location of the sensor but it is not enough to specify exactly the point of view where the sensor is capturing some information from outer world.

The basic plane in used orthogonal space needs to be defined. Let B$[x_2, y_2, z_2]$ be the points in orthogonal space where $x_1 \in (-\infty, +\infty)$, $y_1 \in (-\infty, +\infty)$ and $z_1 = 0$. All these points defined above determine the x-y plane. The situation is analogous in other two possible basic planes: x-z and y-z planes. It should be noted that huge number of other planes, which are not orthogonal to axes, may be constructed and will not be required to be defined for rest of this chapter.

The sensor orientation schema is shown in Figure 2.2. It is defined by two angles α and β, where $\alpha \in (0, 2\pi)$ and $\beta \in (0, 2\pi)$. β is the angle between the optical axis of the camera and the x-y plane. When $\beta = 0$, all potential points of the optical axis lie within that x-y plane. When $\beta = \frac{\pi}{2}$, optical axis of the camera is orthogonal to x-y plane and the optical axis of the camera which comes out in the front of the camera is in the same direction as increasing of values on the z axis. More simply, in this case the camera is capturing what is happening above the x-y plane as shown in Figure 2.2. Analogously to that, α is the angle between the optical axis of the sensor and the x-z plane.

Formally the sensor position is defined as:

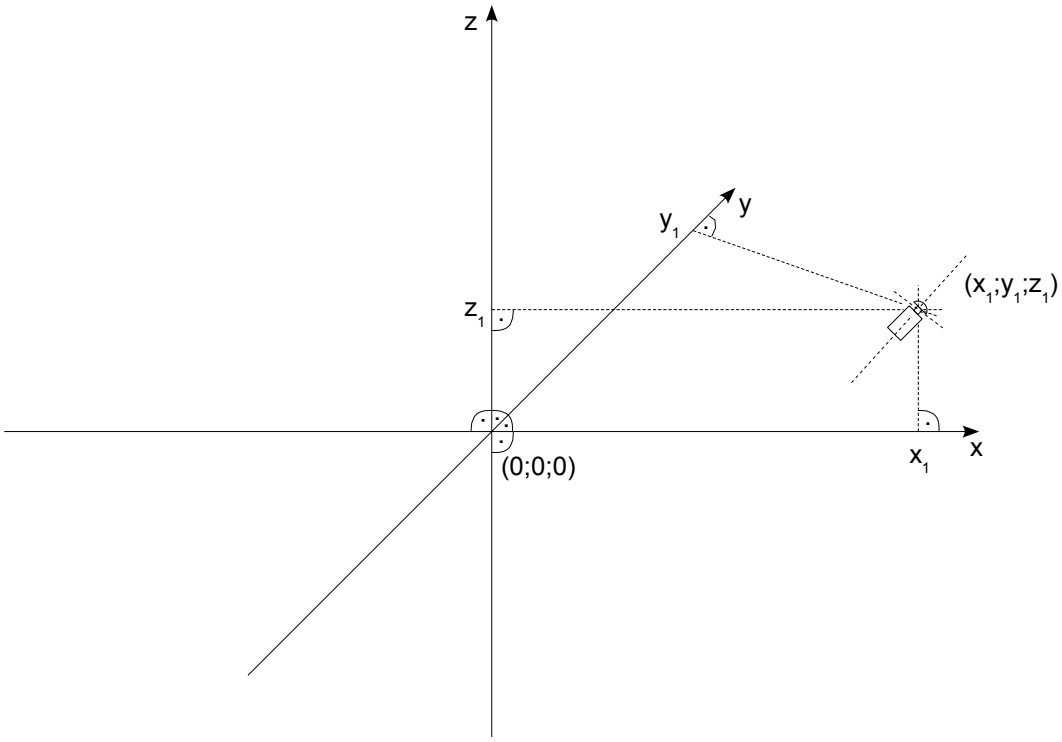


Figure 2.1: Model of orthogonal three dimensional space, where a possible location of a camera is shown.

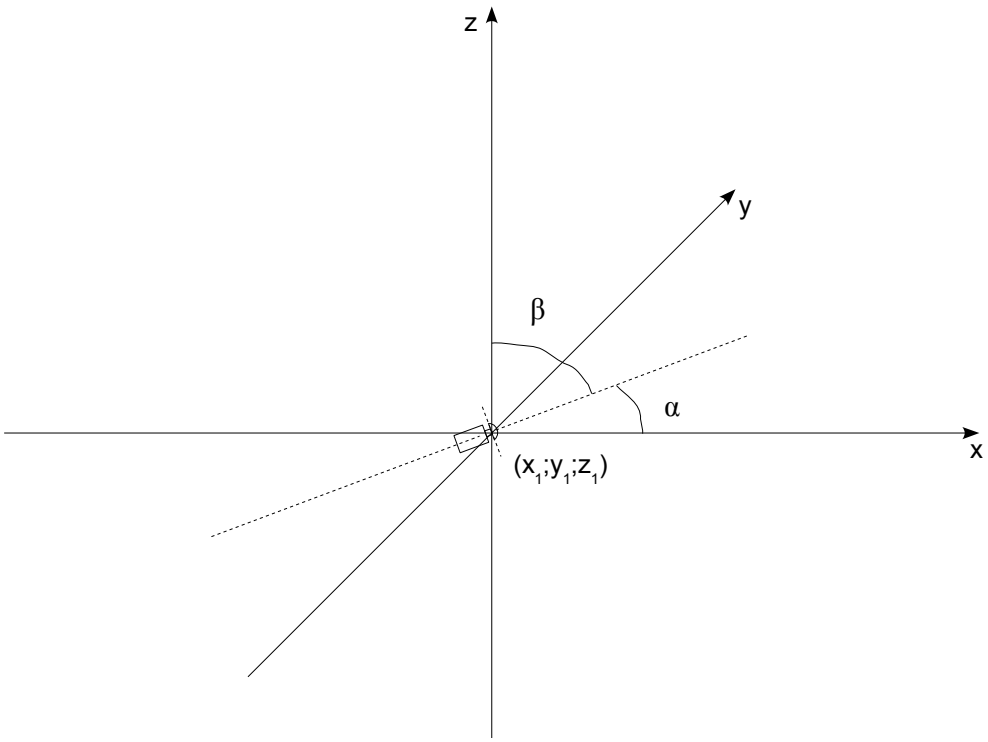


Figure 2.2: Definition of orientation of a sensor at some location according to the x-y and y-z planes.

$$L(x_l, y_l, z_l, \alpha, \beta, \phi) \tag{2.1}$$

where x_l, y_l and z_l is the position of camera and α, β specify the sensor orientation and ϕ represents the sensor rotation around optical axis. Domain of that function is defined:

$$x_l \in (-\infty, +\infty), y_l \in (-\infty, +\infty), z_l \in (-\infty, +\infty),$$
$$\alpha \in \langle 0, 2\pi \rangle, \beta \in \langle 0, 2\pi \rangle, \phi \in \langle 0, 2\pi \rangle$$

Image obtained at this position is simply defined as:

$$O''(X, Y, L, \tau) \tag{2.2}$$

where L describes the sensor position as defined above and

$$X \in (-\infty, +\infty), Y \in (-\infty, +\infty) \tag{2.3}$$

and $\tau \in \langle 0, T)$ in case of video streams and $\tau = 0$ in case that still image is obtained. T represents the maximum length of a recorded video.

More precise definition is:

$$O'(X, Y, x_l, y_l, z_l, \alpha, \beta, \phi, \tau) \tag{2.4}$$

where all known variables have the same meaning as defined above. The variables X and Y specify the area from outer world which is captured by sensor. Sensor is positioned according to the variables $x_l, y_l, z_l, \alpha, \beta, \phi$ as defined above. Theoretically the captured area is infinite, practically a sensor which captures infinite area does not exist. The situation is schematically shown in Figure 2.3. The obtained rectangular area has two main characteristics, it is height and width which can be acquired, and these are dependent on the viewing angle γ and δ. γ angle controls a possible span in vertical direction (Figure 2.3a) and γ angle controls the possible span in horizontal direction (Figure 2.3b). In fact, these two angles define elliptical area which is cropped into an output rectangular area. A content of this output rectangular area is constrained by variables x_c and y_c.

Thus, the area obtained using a sensor is

$$O'(x_c, y_c, x_l, y_l, z_l, \alpha, \beta, \phi, \tau) \tag{2.5}$$

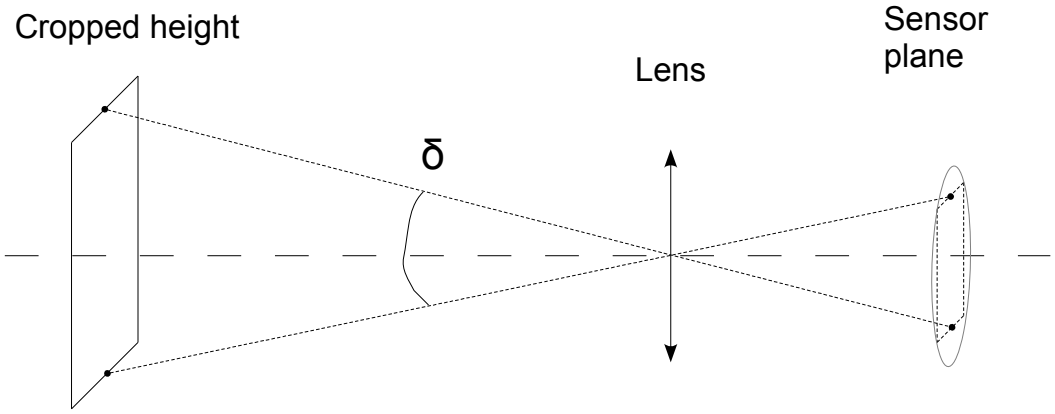
where known variables have the same meaning as defined above and

$$x_c \in \langle -w/2, +w/2 \rangle,$$
$$y_c \in \langle -h/2, +h/2 \rangle$$

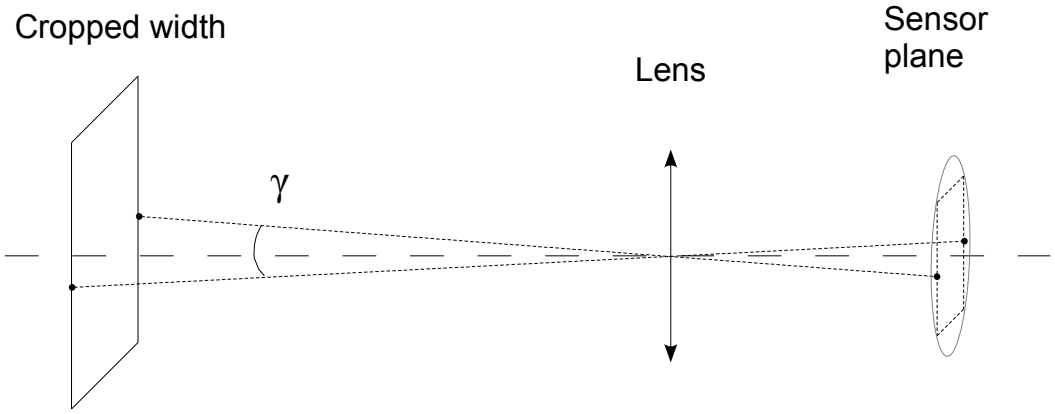
w is with of an image and h is height of an image and formally $w \in (0, +\infty)$ and $h \in (0, +\infty)$.

The width and height of the image is now defined little bit unusual way. It is because of fact that the start point of the captured area lies on the optical axis and thus in the center of the captured area. For the better understanding an displacement of image position [0,0] to top left corner is commonly performed [18].

$$O(x, y, x_l, y_l, z_l, \alpha, \beta, \phi, \tau) = O'(x + w/2, y + w/2, x_l, y_l, z_l, \alpha, \beta, \phi, \tau) \tag{2.6}$$



(a)



(b)

Figure 2.3: The characteristics of each image sensor, the viewing angles δ and γ crop the possible acquired area of the sensor

where $x \in \langle 0, w \rangle$, $y \in \langle 0, h \rangle$, $w \in (0, +\infty)$ and $h \in (0, +\infty)$; w is width of an image and h is height of an image.

The huge subset of videos or images, which are obtained, do not have an exact localization information [84, 44, 50]. This kind of information can be obtained, for example, in controlled environment, where sensors are accurately positioned and the "local" zero position of the environment can be easily selected. In real situation, on the planet earth, the accurate positions of sensors are nearly impossible to be obtained; the global zero position of the environment is not defined and in case that the camera is held by human, the movement of it cannot be obtained and also small movement variances cannot be measured. For this work, the exact position of the sensors is not used, thus can be omitted.

The mapping can be defined as follows

$$f = \{g \mid g : (x, y, x_l, y_l, z_l, \alpha, \beta, \phi, \tau) \rightarrow (x, y, \tau)\} \tag{2.7}$$

2.2 Codomain of the image function

The image function $f(x, y, \tau)$, as described in Equation (2.7), produces color values

$$f(x, y, \tau) \in \mathbb{C} \tag{2.8}$$

where $x \in \langle 0, w \rangle$, $y \in \langle 0, h \rangle$, $w \in (0, +\infty)$ and $h \in (0, +\infty)$ and $\tau = 0$ in case of images and $\tau = \langle 0, T)$ in case of video streams; $\mathbb{C}$ contains all colors, formally

$$\mathbb{C} = \{c \mid c : \langle \lambda_1, \lambda_2 \rangle \rightarrow \mathbb{R}_{0+}\} \tag{2.9}$$

where λ_1 represents lowest wave length [18, 71] boundary of visible light and λ_2 is the highest wave length boundary of visible light, the approximate values are $\lambda_1 = 370nm$ and $\lambda_2 = 720nm$ and $\mathbb{R}_{0+} \in \langle 0, +\infty)$.

The representation using wave length of visible light is problematic. Generally, colors are in computer represented using a color model [18]. One of the widely commonly used one is the RGB model [18]. This model consists of three basic color channels, it is R (red channel), G (green channel) and B (blue channel). By mixing of these three components the similar perception may be observed. The RGB model is formally defined, as follows

$$\begin{aligned} f_{RGB}(x, y, \tau) = (&\int_{\lambda=\lambda_1}^{\lambda_2} \alpha_R(\lambda) f(x, y, \tau) d\lambda, \\ &\int_{\lambda=\lambda_1}^{\lambda_2} \alpha_G(\lambda) f(x, y, \tau) d\lambda, \\ &\int_{\lambda=\lambda_1}^{\lambda_2} \alpha_B(\lambda) f(x, y, \tau) d\lambda) \end{aligned} \tag{2.10}$$

where $\alpha_R(\lambda)$ represents the function which extracts only spectral values which are related to red color, the similar meaning is defined for functions $\alpha_G(\lambda)$ and $\alpha_B(\lambda)$, respectively. Image of this function is defined

$$f_{RGB}(x, y, \tau) \in \mathbb{R}^3_{0+} \tag{2.11}$$

where $\mathbb{R}^3_{0+} = (x_1, x_2, x_3)$ and $x_1, x_2, x_3 \in \langle 0, +\infty)$.

Multiple color representation models exists, basic overview can be found in [18]. In this work another "color" model is needed to be defined because of it is widely used in computer

vision. It is the gray-scale model [18]. In this model only the illumination amplitude of the sensor is considered. This can be obtained by using specialized sensors which cells are sensitive only to illumination but nowadays this signal is more frequently obtained from RGB signal in the following way

$$\begin{aligned} f_{GRAY}(x,y,\tau) = \{&0.299E_R(f_{RGB}(x,y,\tau)) \\ &+0.587E_G(f_{RGB}(x,y,\tau)) \\ &+0.114E_B(f_{RGB}(x,y,\tau))\} \end{aligned} \tag{2.12}$$

where E_R is a function which performs extraction only of the red component of an output of the f_{RGB} function; the E_G and E_B functions extract the other components, respectively. Image of this function is defined

$$f_{GRAY}(x,y,\tau) \in \mathbb{R}_{0+} \tag{2.13}$$

The functions f_{RGB} and f_{GRAY} are fully defined in its analog form.

2.3 Digital form of the image function

For digital image processing f_{RGB} or f_{GRAY} function needs to be digitized [1, 68]. It is done by using sampling of the domain values of the f_{RGB} or f_{GRAY} function. Formally, this is modelled by using multiplication of the input image function with the Dirac impulse [33]. Dirac impulse has the following properties:

(a) its width is sufficiently approaching the zero,

(b) its height is sufficiently approaching the $+\infty$ and

(c) area under the curve is equal to 1.

When obtaining one sample of the function f_{GRAY} the multiplication of f_{GRAY} function with Dirac impulse returns, formally , the average value of the function on an interval which length is the same as the width of the dirac impulse in case of one dimensional signal. The average value of a small area in case of two dimensional image and theoretically the average value of a small cube in case of three dimensional signal.

The real sensors' cells capture the values from wider intervals in case of one dimensional signal and from larger areas in case of two dimensional signal than Dirac impulse does.

In case of two dimensional signal the spatial space is uniformly sampled by a grid. Grid's sampling frequency must be conformed to the Nyquist-Shannon-Kotelnikov theorem [1]:

$$f_s \geq 2f \tag{2.14}$$

where f_s is the sampling frequency and f is the maximum frequency which may be found in the input signal. If Equation (2.14) is not met, the output signal will be aliased and not reconstructible back to the analog form. Special optic devices, such as lens or special filters etc., are used in the front of the image sensors to guarantee this requirement.

The position, at which the input signal is sampled, is called pixel. The signal can be then described as follows

$$f_{RGB}(x,y,\tau) \in \mathbb{R}^3_{0+} \tag{2.15}$$

or

$$f_{GRAY}(x, y, \tau) \in \mathbb{R}_{0+} \tag{2.16}$$

where $x \in \langle 0, w \rangle$, $y \in \langle 0, h \rangle$, $w, h \in \mathbb{N}$, w is with of an image in pixels and h is height of an image in pixels. The image of the functions presented above still contains analog values which needed to be converted to digital form too. This is achieved by quantization process [68]. First of all, the minimum and maximum analog value of the input signal needs to be established and according to this interval the certain number of levels is uniformly emplaced to the interval of analog values. When converting all samples of an analog signal to digital values the closest level's value is used as an output. The number of levels is often chosen as the power of 2. Generally, the converter is called n-bit analog-to-digital converter (ADC) and the interval of output values is defined as $\langle 0, 2^n - 1 \rangle$. In case that the number of levels is $256 = 2^8$ the converter is called as 8-bit ADC and the interval of values is $\langle 0, 255 \rangle$.

The resulting signal can be described as follows

$$f_{RGB}(x, y, \tau) \in (\langle 0, 2^n - 1 \rangle, \langle 0, 2^n - 1 \rangle, \langle 0, 2^n - 1 \rangle) \tag{2.17}$$

or

$$f_{GRAY}(x, y, \tau) \in \langle 0, 2^n - 1 \rangle \tag{2.18}$$

where $x \in \langle 0, w \rangle$, $y \in \langle 0, h \rangle$, $w, h \in \mathbb{N}$, n represents the digital converter characteristics as described above and the assumption, that every channel of function f_{RGB} is converted by n-bit ADC, is done.

Physically, the image sensors are constructed as a grid of cells. Every cell contains three small illumination sensitive areas equipped with filters for red, green and blue channel of the RGB color model. Therefore, each sensor captures directly that part of light spectrum which is needed, as defined in RGB color model. The obtained analog values from each cell are digitized usually using 8-bit ADC and the sensor generally outputs directly digitized form of captured scene. In this case the analog signal path is reduced to avoid noise injection or crosstalk. Unfortunately, the noise is accumulated to the output signal directly in the cell of the sensor, because of a purity level of the material which was used for creating of the sensor. But there is a high probability that the area of neighborhooding pixels will be disturbed by a noise in a similar manner.

Chapter 3

Content representation

When digital image or sequence of digital images is acquired and stored in compressed or uncompressed form, subsequent algorithmic processing is possible using computer. The main goal of computer vision is to understand to its content preferably in a similar way as the human being does.

In the area of computer vision, the content representation can be modeled using for example local low-level features which are extracted from images or videos. It is generally done by extrema searching and each extrema location is described by using neighborhood area of the extrema. The searched space is called spatial domain when the image is searched and is two dimensional. In case of videos the domain is extended with time domain, we call it spatio-temporal domain which is three-dimensional.

The neighborhood area, as defined above, from which some description will be computed, is predefined and it is a small matrix of pixels in case of images and small cube of pixels in case of videos. The output of the description process is generally called feature vector, with one important characteristic, each feature vector obtained using one type of feature extractor with the same settings outputs a number of feature vectors with the same dimensionality.

Formally, the feature vectors are obtained from discretized image function $f_{RGB}(x, y, \tau)$ as defined in section 2.3; the feature extraction M is defined:

$$M(f_{RGB}) = \{(x, y, \tau, D) \mid \textbf{keypoint}(f_{RGB}, x, y, \tau) = 1 \ \wedge \ D = \textbf{fextract}(f_{RGB}, x, y, \tau)\} \tag{3.1}$$

where $\textbf{keypoint}(f_{RGB}, x, y, \tau)$ is the function which determines whether at a given position in the given image function a key point, which should be described, exists. Codomain of the function is defined: $\textbf{keypoint}(f_{RGB}, x, y, \tau) \in \{0, 1\}$. The $\textbf{fextract}(f_{RGB}, x, y, \tau)$ function performs extraction of the feature vector which is stored as a set D at a given position in the given image function. The other variables are defined:

$$x \in \langle 0, \text{width} \rangle$$

$$y \in \langle 0, \text{height} \rangle$$

$$\tau \in \langle 0, \text{T} \rangle$$

and in the case of images $\tau = 0$. Generally the notation represents a set of quadruples of variable length. Each quadruple contains the position information (x, y, τ) and the feature vector as a vector of numbers D. The cardinality of the set represents the number of local features extracted and it can be noted as $|M(f_{RGB})|$.

Generally, the number of output feature vectors is different for each processed image or video. This characteristic brings the need for additional processing, where a number of local feature vectors are processed and converted to one fixed-sized representation, where one video or subvideo or image or part of an image is represented using one "higher-level" feature vector.

In Section 3.1 and in Section 3.2, a basic overview of well known image features will be presented as well as the space-time feature extraction techniques whose are frequently used today. Section 3.3 presents the dense sampling technique based on a fact that under certain conditions searching of keypoints can be skipped. The Section 3.4 presents basic principles for transforming of a set of local low-level features to one fixed-sized representation.

It should be noted that the rest of this chapter refers to many sources and generally it can be said that every source uses slightly different mathematical notation. The notation is adopted without changes to avoid inaccuracies of presented facts.

3.1 Image feature extractors

In this Section, generally known image feature extraction algorithms will be presented, such as SIFT [55, 54], SURF [6] and as well as the latest algorithms, such as FREAK [60].

SURF

Herbert Bay et al. presented a novel scale- and rotation-invariant detector and descriptor in [6]. The detection process is based on Hessian matrix and the description process is based on the Haar wavelet filters.

The detector is based on Hessian matrix because of its good performance in accuracy. Blob-like structures are detected at locations where the determinant is maximum and the determinant of Hessian is also used for the scale selection.

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

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{3.2}$$

where $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in point $\mathbf{x}$, and similarly for $L_{xy}(\mathbf{x}, \sigma)$ and $L_{xy}(\mathbf{x}, \sigma)$.

Gaussians are optimal for scale-space analysis but in practise they have to be discretised and cropped. This leads to a loss in repeatability under certain image rotations. This is a weakness for Hessian-based detectors in general. The computation of hessian matrix is approximated using box filters. These approximate second order Gaussian derivatives can be evaluated at a very low computational cost using integral images and, in advance, the calculation time therefore is independent of the filter size.

The 9×9 box filters in figure 3.1 are approximations of a Gaussian with $\sigma = 1.2$ and represent the lowest scale (i.e. highest spatial resolution). The approximations are denoted as D_{xx}, D_{yy} and D_{xy}. The weights applied to the rectangular regions are kept simple for computational efficiency, but the further balancing of the relative weights in the expression for the Hessian's determinant with $\frac{|L_{xy}(1.2)|_F |D_{xx}(9)|_F}{|L_{xx}(1.2)|_F |D_{xy}(9)|_F} \simeq 0.9$ is needed. $|x|_F$ is the Frobenius norm, this yields

$$det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{3.3}$$

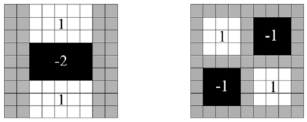


Figure 3.1: Left to right: the approximation of the second order Gaussian partial derivative in y- (D_{yy}) and xy-direction (D_{xy}). The grey regions are equal to zero. The picture is taken from [6].

Furthermore, the filter responses are normalised with respect to their size. This guarantees a constant Frobenius norm for any filter size.

Scale-space representation is created using up-scalling of the filter size. The output of the 9×9 filter (as shown in Figure 3.1) is considered as the initial scale layer which is referred as $s = 1.2$ (it corresponds to Gaussian derivatives with $\sigma = 1.2$). The following layers are obtained by filtering the image with gradually bigger masks. That leads to the filter sizes 9×9, 15×15, 21×21, 27×27, etc. At larger scales, the step between consecutive filter sizes should also scale accordingly. Hence, for each new octave, the filter size increase is doubled (going from 6 to 12 to 24). Simultaneously, the sampling intervals for the extraction of the interest points can be doubled as well.

As the ratios of the filter layout remain constant after scaling, the approximated Gaussian derivatives scale accordingly. Furthermore, as the Frobenius norm remains constant for the filters, they are already scale normalised.

In order to localise interest points in the image and over scales a non-maximum suppression in a $3 \times 3 \times 3$ neighbourhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space with the method proposed by Brown et al. [13]. Scale space interpolation is especially important according to proposed scale-space representation, because of the difference in scale between the first layers of every octave is relatively large.

The SURF descriptor describes the distribution of intensity content within the interest point neighborhood. The Haar wavelet responses in x and y direction within a circular neighbourhood of radius 6s around the interest points, with s the scale at which the keypoint was detected. The sampling step is scale dependent and chosen to be s. The size of the wavelets are also scale dependent and are set to 4s. Again integral images can be used for fast filtering.

Once the wavelet responses are calculated and weighted with a Gaussian ($\sigma = 2.5s$) centered at the interest point, the responses are represented as vectors in a space with the horizontal response strength along the abscissa and the vertical response strength along the ordinate. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window covering an angle of $\frac{\pi}{3}$. The horizontal and vertical responses within the window are summed. The two summed responses then yield a new vector. The longest such vector defines the orientation of the interest point. The size of the sliding window is a parameter, which must be chosen carefully.

When the orientation of the key point is established, the extraction of feature descriptor can be performed. The rectangular area centered around the interest point and oriented along the interest point orientation is used for that purpose. The size of window is set to 20s. The region is split up regularly into smaller 4×4 square sub regions. This step preserves important spatial information.

For each sub-region the Haar wavelet responses at 5×5 regularly spaced sample points

are computed. Haar wavelet response in horizontal direction is denoted as d_x and response in vertical direction is denoted as d_y. It should be noted that the Haar wavelets are calculated in the unrotated image and the responses are then interpolated according to key point orientation. The responses d_x, d_y are first weighted with a Gaussian ($\sigma = 3.3s$) centered at the interest point. Then, the responses d_x and d_y are summed up over each sub-region and form a first set of entries in the feature vector. In order to improve the output description the sum of the absolute values of the Haar wavelet responses $|d_x|$ and $|d_y|$ is extracted. Hence, each sub-region has a four-dimensional descriptor vector $\mathbf{v}$ for its underlying intensity structure $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Concatenating this for all 4×4 sub-regions, this results in a descriptor vector of length 64. The wavelet responses are invariant to a bias in illumination. Invariance to contrast is achieved by turning the descriptor into a unit vector.

SIFT

David G. Lowe presented innovative approach for local features extraction in [55, 54] which is called SIFT keypoint detector. The approach is based on detection of keypoints in an image and its description using feature vectors which are invariant to image size change, translation, rotation and partially invariant to illumination change, affine transformations and 3D projection. Features are localized in spatial domain as well as in frequency domain, a probability of wrong detection ,while the area is partially occluded by another object or some noise is present in image, is reduced. From one input image, usually hundreds of feature vectors are extracted using an effective algorithm. Features are very distinguishing, even when the comparison against huge database of feature vectors is performed, the probability of correct comparison is very high. Computational demandingness is reduced by using cascade filtering where time demanding operations are performed only on preselected areas whose passed a selection test.

The procedure of feature extraction may be summarizes as follows:

1. Scale-space pyramid creation
2. extrema detection accros the whole scale-space
3. removing of unstable extrema (e.g. extrema on the edges, etc.)
4. description of neighborhood of remaining extrema

The whole scale-space is searched for positions, whose are invariant with respect to image translation, scalling and rotation, and are minimally affected by noise and small distortions. It has been shown in [51] that under some general assumptions on scale invariance, the Gaussian kernel and its derivatives are the only possible smoothing kernels for scale space analysis.

Rotation invariance is achieved by selection of the positions of a minima and a maxima of function difference of Gaussian [18], this function is applied in the whole scale space. This can be computed in a efficient way by creation of image pyramid and key points can be localized across all scales with a high variance, this procedure provides stable image characterizing.

2D Gauss function is separable, its convolution with input image can be effectively computed by using two 1D Gauss function in horizontal and vertical directions:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2} \tag{3.4}$$

For keypoints localisation all smoothing operations are done with $\sigma = \sqrt{2}$, it can be approximated with sufficient accuracy using a 1D kernel with 7 points. By doing this image A is obtained, afterwards it is smoothed again by using the same arguments, image B is obtained which is effectively smoothed with parameters: $\sigma = 2$. The difference of Gaussians function is now computed by subtracting image B from image A. For creation of subsequent pyramid level already smoothed B image is resized using bilinear interpolation with a pixel spacing of 1.5 in each direction. Pixel spacing 1.5 means that new pixel will be computed as a combination of 4 adjacent pixels. According to that procedure all pyramid levels are computed.

Maxima (or minima) of this scale space function are determined by comparing each pixel in the pyramid to its neighborhood. First of all, a pixel is compared to its eight neighbors at the same level of the pyramid. If the tested pixel is a maxima (or minima) at this pyramid level, then the closest pixel location is calculated at the subsequent lowest level of the pyramid (1.5 resampling level is taken into account). If the pixel remains higher (or lower) than this closest pixel and its 8 pixel neighborhood, then the test is repeated for the level above.

Each key point location, which were detected, is described. The smoothed image A at each level of the pyramid is processed to extract image gradients (M_{ij}) and orientations (R_{ij}), computation is done using pixel differences:

$$M_{ij} = \sqrt{(A_{i,j} - A_{i+1,j})^2 + (A_{i,j} - A_{i,j+1})^2} \tag{3.5}$$

$$R_{ij} = atan2(A_{i,j} - A_{i+1,j}, A_{i,j} - A_{i,j+1}) \tag{3.6}$$

Difference of two pixels is a very simple task to compute and the result is precise enough thanks to higher level of image smoothness. Robustness to illumination change is improved because of thresholding of gradient value to a value which equals to 10% of maximum possible gradient value. This reduces the effect of a change in illumination direction for a surface with 3D relief, as an illumination change may result in large changes to gradient magnitude but is likely to have less influence on gradient orientation.

A canonical orientation is assigned to each key location so that the image descriptors are invariant to rotation. The orientation is determined by the peak in a histogram of local image gradients. The orientation histogram is created using a Gaussian-weighted window with σ of three times that of the current smoothing scale. These weights are multiplied by the thresholded gradient values and accumulated in the histogram at locations corresponding to the orientation, R_{ij}. 360 degree range of rotations is covered by the histogram with 36 bins. The histogram is smoothed prior to peak selection.

Each key position is now identified by a stable location, scale and orientation; the local image region, in a manner invariant to these transformations, can be described. In addition, it is desirable to make this representation robust against small shifts in local geometry, such as arise from affine or 3D projection. This robustness can be obtained by representing the local image region with multiple images representing each of a number of orientations (reffered to as orientation planes). Each orientation plane contains only the gradients corresponding to that orientation, with linear interpolation used for intermediate orientations. Each orientation plane is blurred and resampled to allow for larger shifts in positions of the gradients.

For each keypoint, the pixel sampling, from the pyramid level at which the key point was detected, is used. All pixels which fall in a circle of 8 pixels radius around the key location are inserted into the orientation plane. The orientation is measured relative to

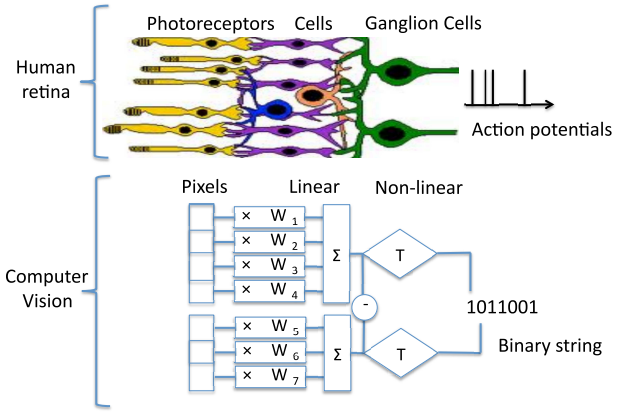


Figure 3.2: Human retina model, the biological pathways leading to action potentials is emulated by simple binary tests over pixel regions. The picture is taken from [60].

that of the key by subtracting the key's orientation. Eight orientation planes are used , each is sampled over 4×4 grid of locations, with a sample spacing 4 times that of the pixel spacing used for gradient detection. The blurring is done by allocation the gradient of each pixel among its 8 closest neighbors in the sample grid, using linear interpolation in orientation and the two spatial dimensions.

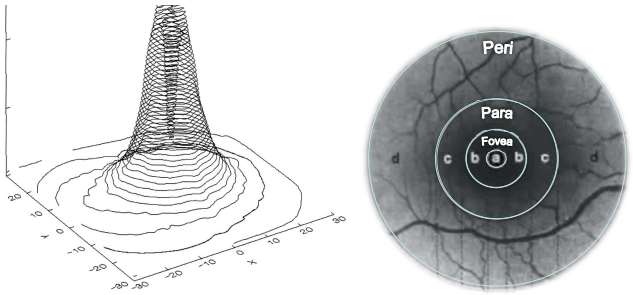
The same process is repeated for a second level of the pyramid one octave higher. This time 2×2 sample region is used (instead of 4×4). Approximately the same region will be examined at both scales, so that any nearby occlusions will not affect one scale more than the other. Therefore, the total number of samples in the SIFT vector , from both scales, is $8 \times 4 \times 4 + 8 \times 2 \times 2$ of 160 elements, giving enough measurements for high specificity.

FREAK

Alahi et al. in [60] presented the fast binary descriptor which is constructed with respect to the known information about human's retina, it is called the FREAK (fast retina keypoint).

It is believed that the human retina extracts details from images using Difference of Gaussians [25] of various sizes and encodes such differences with action potentials (see Figure 3.1). The topology of the retina is very important. First, several photoreceptors influence the ganglion cell. The region where light influences the response of a ganglion cell is the receptive field. Its size and dendritic field increases with radial distance from the foveola (Figure 3.3b). The spatial distribution of ganglion cells reduces exponentially with the distance to the foveal. They are splitted into these areas: foveal, fovea, parafoveal and perifoveal. Each area holds an interesting role in the process of detecting and recognizing objects. The higher resolution is captured in the fovea whereas a low resolution objects are captured in the perifoveal. The decreasing of resolution can be interpreted as a body resource optimization. The analogy is presented in the Figure 3.1.

The sampling grid, inspired by the retina, is used. That grid is circular with the difference of having higher density of points near the center. The density of points drops exponentially (see Figure 3.3a). Each sample point needs to be smoothed to be less sensitive to noise. According to the retina model, the different kernels size for every sample points



(a) Visualization of the density of ganglion cells over the retina.

(b) Retina areas

Figure 3.3: The distribution of ganglion cells over the retina is splitted into four areas: a. the foveola, b. fovea, c. parafoveal, d. perifoveal. The picture is taken from [60].

is used. The size is exponentially changed and the receptive fields are overlapped. The topology of receptive fields are illustrated in Figure 3.4. Each circle represents the standard deviation of the Gaussian kernels [52] applied to the corresponding sampling points.

The changing of the size of the Gaussian kernels [52] with respect to the log-polar retinal pattern and the overlapping of the receptive fields lead to better performance. Another redundancy which brings more discriminative power is added. Let us consider three receptive fields A,B and C with intensities I_A, I_B and I_C, where $I_A > I_B, I_B > I_C$ and $I_A > I_C$. In case that the fields do not have an overlap, the last test $I_A > I_C$ do not add any additional discriminant information. However, in case that the fields overlap, partially new information can be encoded. Generally, adding such redundancy allows to use less receptive fields which is known strategy employed in compressed sensing.

The binary descriptor F is constructed by thresholding the difference between pairs of receptive fields with their corresponding Gaussian kernel. F is a binary string formed by a sequence of one-bit Difference of Gaussians:

$$F = \sum_{0 \leq a < N} 2^a T(P_a) \tag{3.7}$$

where P_a is a pair of receptive fields, N is the desired size of the descriptor, and

$$T(P_a) = \begin{cases} 1 & \text{if } I(P_a^{r_1} - P_a^{r_2}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

with $I(P_a^{r_1})$ is the smoothed intensity of the first receptive field of the pair P_a.

With relatively small number of receptive fields, thousands of pairs are possible; it leads to a large descriptor which may not be useful to efficiently describe the image. Hence, the best pairs have to be learnt from training data according to this procedure:

1. A matrix D of nearly fifty thousand of extracted keypoints is created. Each row corresponds to a keypoint represented with its large descriptor made of all possible pairs in the retina sampling parent illustrated in Figure 3.4. 43 receptive fields, leading to approximately one thousand pairs, is used.

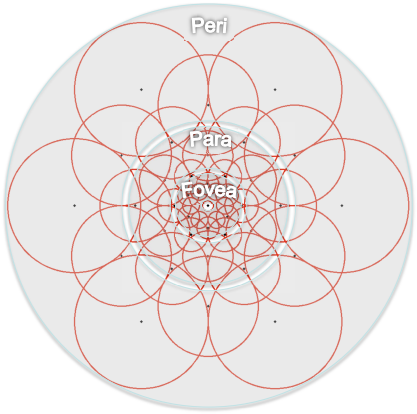


Figure 3.4: Illustration of the FREAK sampling pattern. Each circle represents the receptive field where the image is smoothed with its corresponding Gaussian kernel. The picture is taken from [60].

2. The mean of each column is computed. In order to obtain a discriminant feature, high variance is desired. A mean of 0.5 leads to the highest variance of a binary distribution.
3. The columns are ordered with respect to the highest variance.
4. The best column is kept (mean of 0.5) and remaining columns which have low correlation with the other selected columns are iteratively added

A coarse-to-fine ordering of the output pairs is automatically preferred. The selected output pairs are clustered into four groups (128 pairs per group). It was observed that the first 512 pairs are the most relevant and adding more pairs is not increasing the performance. A symmetric scheme is captured due to the orientation of the pattern along the global gradient. The first cluster involves mainly peripheral receptive fields whereas the last ones implicates highly centered fields.

According to the retina model, the descriptor matching can be cascaded. Firstly, the first 16 bytes of the FREAK descriptor, which represent coarse information, is matched. In case the distance is smaller than threshold the comparison continues. Generally, the comparison of first 16 bytes of descriptor discards 90% of the candidates.

The keypoint rotation is estimated by summing the local gradients over selected short distance pairings. The long pairs are used to compute the global orientation but only from selected pairs with symmetric receptive fields with respect to the center.

3.2 Video feature extractors

In this Section, generally known space-time feature description algorithms will be presented, such as STIP [47, 48], HesSSTIP [88], Cuboids [20], Dense Trajectories [82] and ω-flow [38].

STIP

The keypoint detection of this extractor is based on the idea of the Harris interest points detector [30, 73]; i.e. to detect locations in a spatial image f^{sp} where the image values have

significant variations in both directions. For a given scale of observation σ_l^2, such interest points can be found from a windowed second moment matrix integrated at scale $\sigma_i^2 = s\sigma_l^2$

$$\mu^{sp} = g^{sp}(\cdot; \sigma_i^2) * \begin{pmatrix} (L_x^{sp})^2 & L_x^{sp} L_y^{sp} \\ L_x^{sp} L_y^{sp} & (L_y^{sp})^2 \end{pmatrix} \tag{3.9}$$

where L_x^{sp} and L_y^{sp} are Gaussian derivatives defined as

$$\begin{aligned} L_x^{sp}(\cdot; \sigma_l^2) &= \partial_x(g_{sp}(\cdot; \sigma_l^2) * f^{sp}) \\ L_y^{sp}(\cdot; \sigma_l^2) &= \partial_y(g_{sp}(\cdot; \sigma_l^2) * f^{sp}) \end{aligned} \tag{3.10}$$

and where g^{sp} is the spatial Gaussian kernel

$$g^{sp}(x, y; \sigma^2) = \frac{1}{2\pi\sigma^2} exp\left(\frac{-(x^2 + y^2)}{2\sigma^2}\right) \tag{3.11}$$

As the eigenvalues λ_1, λ_2, $(\lambda_1 \leq \lambda_2)$ of μ_{sp} represent characteristics variations of f^{sp} in both image directions, two significant values o λ_1, λ_2 indicate the presence of an interest point. To detect such points, the positive maxima of the corner function should be detected

$$H^{sp} = det(\mu^{sp}) - k\ trace^2(\mu^{sp}) = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{3.12}$$

The idea as presented above for spatial domain can be extended into the spatio-temporal domain by requiring the image values in space-time to have large variations in both the temporal and the spatial dimensions. Such points will be spatial interest points with a distinct location in time corresponding to the moments with non constant motion of the image in a local spatio-temporal neighborhood [46].

Spatio-temporal image sequence is modelled using a function $f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$ and its linear scale-space representation $L : \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R}_+^2 \mapsto \mathbb{R}$ is constructed by convolution of f with an anisotropic Gaussian kernel with distinct spatial variance σ_l^2 and temporal variance τ_l^2

$$L(\cdot; \sigma_l^2, \tau_l^2) = g(\cdot; \sigma_l^2, \tau_l^2) * f(\cdot) \tag{3.13}$$

where the spatio-temporal separable Gaussian kernel is defined as

$$g(x, y, t; \sigma_l^2, \tau_l^2) = \frac{exp\left(-\frac{(x^2+y^2)}{2\sigma_l^2} - \frac{t^2}{2\tau_l^2}\right)}{\sqrt{(2\pi)^3\sigma_l^4\tau_l^2}} \tag{3.14}$$

Similar to the spatial domain, the extended spatio-temporal second-moment matrix is considered. It is a 3-by-3 matrix composed of first order spatial and temporal derivatives averaged with a Gaussian weighting function$g(\cdot; \sigma_i^2, \tau_i^2)$

$$\mu = g(\cdot; \sigma_i^2, \tau_i^2) * \begin{pmatrix} (L_x)^2 & L_x L_y & L_x L_t \\ L_x L_y & (L_y)^2 & L_y L_t \\ L_x L_t & L_y L_t & (L_t)^2 \end{pmatrix} \tag{3.15}$$

where the integration scales are $\sigma_i^2 = s\sigma_l^2$ and $\tau_i^2 = s\tau_l^2$, while the first order derivatives are defined as $L_\xi = (\cdot; \sigma_l^2, \tau_l^2) = \partial_\xi(g * f)$ this definition of second-moment matrix μ has been previously used in the context of optic flow computation.

To detect interest points,the regions in f having significant eigenvalues λ_1, λ_2, λ_3 of μ are searched. For such purposes the extension of the Harris corner function (3.12) defined

for the spatial domain can be extended into the spatio-temporal domain by combining the determinant and the trace of μ in the following way

$$H = det(\mu) - k\ trace^3(\mu) = \lambda_1\lambda_2\lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3 \tag{3.16}$$

To show that the positive local maxima of H correspond to points with high values of $\lambda_1, \lambda_2, \lambda_3 (\lambda_1 \leq \lambda_2 \leq \lambda_3)$, the ratios $\alpha = \lambda_2/\lambda_1$ and $\beta = \lambda_3/\lambda_1$ can be defined, and thus, H can be rewritten in the following way

$$H = \lambda_1^3(\alpha\beta - k(1 + \alpha + \beta)^3) \tag{3.17}$$

Then, from the requirement $H \geq 0$ $k \leq \alpha\beta/(1 + \alpha + \beta)^3$ is obtained and it follows that as k increases towards its maximal value $k = 1/27$, both ratios α and β tend to one. For sufficiently large values of k, positive local maxima of H correspond to points with high variation of the image gray-values in both the spatial and temporal dimensions. Thus, spatio-temporal interest points of f can be found by detecting local positive spatio-temporal maxima in H.

To characterize motion and appearance of local keypoint locations, the histogram descriptors of space-time volumes in the neighborhood of detected points are computed. The size of each volume $(\Delta_x, \Delta_y, \Delta_t)$ is related to the detection scales by $\Delta_x, \Delta_y = 2k\sigma, \Delta_t = 2k\tau$. Each volume is subdivided into a $n_x \times n_y \times n_t$ grid of cells; for each cell, 4-bin histogram of gradient orientation [48] (HOG) and 5-bin histogram of optical flow [48] (HOF) are computed. Normalized histograms are concatenated into HOG and HOF descriptors and are similar in spirit to the well known SIFT descriptor. Authors recommend to use the following paremeters for histograms creation: $k = 9, n_x, x_y = 3, n_t = 2$. The both output histograms can be concatenated, in this case the output descriptor is called HOG/HOF.

HesSTIP

Bay et al. proposed in [6] the use of the Hessian matrix for statio-temporal feature detection:

$$H(\cdot, \sigma^2, \tau^2) = \begin{bmatrix} L_{xx} & L_{xy} & L_{xt} \\ L_{yx} & L_{yy} & L_{yt} \\ L_{tx} & L_{ty} & L_{tt} \end{bmatrix} \tag{3.18}$$

The strength of each interest point at a certain scale is then computed by $S = |det(H)|$. This can be considered as a spatio-temporal extension of the saliency measure for blob detection as proposed in [7]. However, a positive value of S does not guarantee all eigenvalues of $H(\cdot, \sigma^2, \tau^2)$ having the same sign.

Using the Hessian matrix, scale selection can be realized in various ways. The γ-normalization and Simultaneous Localization and Scale Selection will be presented.

Using γ-normalization , the saliency measure is altered to ensure that the correct scales σ_0 and τ_0 are found on a perfect Gaussian blob $g(x, y, t; \sigma_0^2, \tau_0^2)$. At the center of this blob, the determinant is determined by the first term $L_{xx}L_{yy}L_{tt}$ as all other terms vanish. The γ-normalized determinant at the center can be written as

$$L_{xx}^{\gamma norm} L_{yy}^{\gamma norm} L_{tt}^{\gamma norm} = \sigma^{2p}\tau^{2q} L_{xx}L_{yy}L_{tt} \tag{3.19}$$

To obtain the extrema, $det(H)^{\gamma norm}$ should be differentiated with respect to the spatial and temporal scale parameters σ^2 and τ^2, and set these derivatives equal to zero. All terms

but the first vanish at the center of Gaussian blob. From this analysis, it follows that the local extrema $\tilde{\sigma}$ and $\tilde{\tau}$ coincide with the correct scales σ_0 and τ_0 if $p = 5/2$ and $q = 5/4$ is set.

It should be noted, that these values are related to the values a, b, c and d for the normalization of the Laplacian, namely $p = 2a + c$ and $q = 2b + d$. This reflects the fact that the determinant at the center of the Gaussian blob reduces to the product of two spatial second-order derivatives and one temporal second-order derivative. A γ-normalized operator is obtained with $\gamma \neq 1$. This induces, however, that the measure used for scale selection is not truly scale invariant, and it cannot be used to find local maxima over scales.

In contrast with the normalized Laplacian, scale invariance and good scale selection can be achieved simultaneously with the scale-normalized determinant of the Hessian. Using $p = 2$ and $q = 1$ (implies $\gamma = 1$) in equation (3.19), the following relationship between the local extrema $(\tilde{\sigma}, \tilde{\gamma})$ and the correct scales (σ_0, τ_0) for a Gaussian blob is obtained:

$$\tilde{\sigma}^2 = \frac{2}{3}\sigma_0^2 \quad \tilde{\tau}^2 = \frac{2}{-}3\tau_0^2 \tag{3.20}$$

In general, it can be shown that, in D dimensions, the determinant of the scale normalized Hessian, with $\gamma = 1$, reaches an extremum at the center of a Gaussian blob $g(x'\sigma_0)$ with $\sigma_0 = [\sigma_{0,1}, \ldots, \sigma_{0,D}$, for al scales

$$\tilde{\sigma} = \sqrt{\frac{2}{D}\sigma_0} \tag{3.21}$$

Even through the fact that the detected scale $\tilde{\sigma}, \tilde{\tau}$ do not coincide with the correct scales σ_0, τ_0, they are related by a fixed scale factor. This fact makes it trivial to obtain the latter.

Since now a single, scale-invariant measure, that can be used both for the localisation as well as for the selection of the temporal and spatial scale, is known; a non-iterative method can be used for that purpose. To this purpose, the local extrema over the 5D space is selected, the space can be denoted by (x, y, t, σ, τ). The scale of each interest point found must be multiplied with the factor $\sqrt{3/2}$ to obtain the real scales of the underlying signal. This brings a clear speed advantage over the iterative procedure of [45], avoids problems with convergence and allows for the extraction of any number of features simply by changing the threshold of the saliency measure.

Description of neighborhood of the detected interest points an extended version of the SURF descriptor [6] is implemented. Around each interest point with spatial scale σ and temporal scale τ, a rectangular volume is defined. The volume has dimensions: $s\sigma \times s\sigma \times s\tau$ with s a user defined magnification factor (typically 3). The volume is subsequently divided into $M \times M \times N$ bins, where M is the number of bins in the spatial domain and N is the number of bins in temporal domain, respectively. The bins are filled by a weighted sum of uniformly sampled responses of the 3 axis-aligned Haar-wavelets d_x, d_y, d_t. For each bin, the vector $v = (\sum d_x, \sum d_y, \sum d_t)$ is stored.

In case that the invariance to (spatial) rotation is required, the dominant orientation is computed (as proposed in [6]) except that, for the spatio-temporal case all Haar-wavelets used in this step are stretched out over the full length of the temporal scale of the interest point.

Cuboids

Dollar et al. [20] proposed a variation of space-time interest points detector and descriptor to deal with a special problem domains, such as, rodent behaviour recognition or facial

expressions, etc. The detector is designed to err on the side of detecting too many features rather than too few, nothing than object recognition schemes based on spatial interest points deal well with irrelevant and possibly misleading features generated by scene clutter and imperfect detectors. The resulting representation is still orders of magnitude sparser than a direct pixel representation. The response function is calculated by application of separable linear filters. A stationary camera or a process that can account for camera motion is assumed. The response function has the form:

$$R = (I * g * h_{ev})^2 + (I * g * h_{od})^2 \tag{3.22}$$

where $g(x, y; \sigma)$ is the 2D Gaussian smoothing kernel, applied only along the spatial dimensions and h_{ev} and h_{od} are a quadrature pair [28] of 1D Gabor filters which are applied temporally.The even part of the filter is defined as

$$h_{ev}(t; \tau, \omega) = -cos(2\pi t\omega)e^{-t^2/\tau^2} \tag{3.23}$$

and the odd part of the filter is defined as

$$h_{od}(t; \tau, \omega) = -sin(2\pi t\omega)e^{-t^2/\tau^2} \tag{3.24}$$

The term $\omega = 4/\tau$ is generally used, it is efficiently giving the response function R two parameters σ and τ, corresponding roughly to the spatial and temporal scale of the detector.

The detector is set up to fire whenever variations in local image intensities contain periodic frequency components. Generally, no reason to believe, that only periodic movements are interesting, exists. Periodic motions, such as a bird flapping its wings, will indeed evoke the strongest responses, however, the detector responds strongly to a range of other motions, including at spatio-temporal corners positions. Generally, any region with spatially distinguishing characteristics undergoing a complex motion can induce a strong response. Areas undergoing pure translational motion will in general not induce a response as a moving, smoothed edge will cause only a gradual change in intensity at a given spatial location. Areas without spatially distinguishing features cannot induce a response.

At each interest points (local maxima of the response function defined above) a cuboid is extracted. The cuboid contains the spatio-temporally windowed pixel values. The size of the cuboids is set to contain most of the volume of data that contributed to the response function at that interest point.

The authors of the detector later decided to use some kind of descriptor to be able to use Euclidean distance to simply compare two output descriptions. It includes, (1) normalized pixel values, (2) the brigthness gradient and (3) windowed optical flow. The brightness gradient is calculated at each spatio-temporal location (x, y, t), giving rise to three channels (G_x, G_x, G_t) each the same size as the cuboid. To extract motion information Lucas-Kanade optical flow [56] between each pair of consecutive frames is calculated. This induces the creation of two channels (V_x, V_y). Each channel is the same size as the cuboid minus one frame.

Dense Trajectories

Wang et al. [82] presented alternate approach for space-time features extraction called Dense Trajectories. These are extracted for multiple spatial scales (Figure 3.5). Feature points are sampled on a grid spaced by W pixels and tracked in each scale separately. Experimentally, it was observed that sampling step size o $W = 5$ is dense enough to give

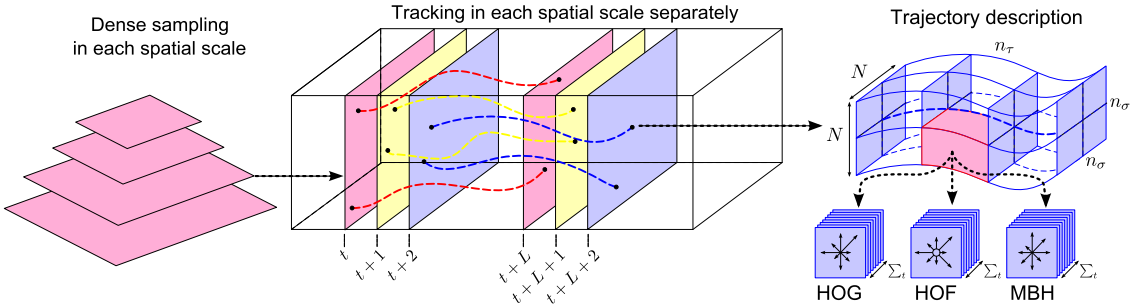


Figure 3.5: Dense trajectory illustration. Left: Feature points are scaled densely for multiple spatial scales. Middle: Tracking process is performed in the corresponding spatial scale over L frames. Right: Descriptors are based on the trajectory shape represented by relative point coordinates as well as appearance and motion information over a local neighborhood of $N \times N$ pixels along the trajectory. In order to capture the structure information, the trajectory neighborhood is divided into a spatio-temporal grid of size $n_\sigma \times n_\sigma \times n_\tau$. The picture is taken from [82].

good results. 8 spatial scales spaced by a factor $1/\sqrt{2}$ were used. Each point $P_t = (x_t, y_t)$ at frame t is tracked to the next frame $t+1$ by median filtering in a dense optical flow field $\omega = (u_t, v_t)$

$$P_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * \sigma)|_{(\overline{x}_t, \overline{y}_t)} \tag{3.25}$$

where M is the median filtering kernel, and $(\overline{x}_t, \overline{y}_t)$ is the rounded position of (x_t, y_t). This is more robust than bilinear interpolation used in [77], especially for points near motion boundaries. Once the dense optical flow field is computed, points can be tracked very densely without additional cost. Points of subsequent frames are concatenated to form a trajectory: $(P_t, P_{t+1}, P_{t+2}, \dots)$. To extract dense optical flow the algorithm, presented by Farneback [24], was used.

A common problem in tracking is drifting. Trajectories tend to drift from their initial location during tracking. To avoid this situation the trajectory length shall be limited to L frames. As soon as a trajectory exceeds length L, it is discarded from the tracking subsystem (Figure 3.5 middle). To assure a dense coverage of the video, the presence of a track on the dense grid is verified in every frame. If no tracked point is found in a $W \times W$ neighborhood, this feature point is sampled and added to the tracking process. The length of the trajectory was chosen experimentally to $L = 15$ fames.

In homogeneous image areas without any structures, it is impossible to track points. In such cases the same criterion as in [72] is used. When a feature point is sampled the smaller eigenvalue of its autocorrelation matrix is checked. In case that it is below a threshold, this point will not be included in the tracking process. Since for action recognition dynamic information is mainly interested, static trajectories are prunned in a pre-processing stage. Trajectories with sudden large displacements, most likely to be erroneous, are also removed.

The shape of a trajectory encodes local motion patterns. Given a trajectory of length L, the shape is described by a sequence $S = (\Delta P_t, \dots, \Delta P_{t+L-1})$ of displacement vectors $\Delta P_t = (P_{t+1} - P_t) = (x_{t+1} - x_t, y_{t+1} - y - t)$. The resulting vector is normalized by the

sum of magnitudes of the displacement vectors:

$$S' = \frac{(\Delta P_t, \ldots, \Delta P_{t+L-1})}{\sum_{j=t}^{t+L-1} ||\Delta P_j||} \tag{3.26}$$

This vector is referred as a trajectory vector descriptor. These trajectories were also evaluated when computed at multiple temporal scales, in order to recognise actions with different speeds. However, this did not practically improve the results. Therefore, length of each trajectory were fixed to length L.

A description of keypoint's 3D neighborhood have become a popular way of video representation [20, 41, 48, 88]. To leverage the motion information in case of dense trajectories, descriptors are computed within a space-time volume around the trajectory (illustration can be seen in Figure 3.5 middle). The size of the volume is $N \times N$ pixels and L frames. To embed structure information in the representation, the volume is subdivided into a spatio-temporal grid $n_\sigma \times n_\sigma \times n_\tau$. The default parameters, which were experimentally obtained, are $N = 32, n_\sigma = 2, n_\tau = 3$.

The descriptors, which were proposed to be obtained along the trajectory, include: HOG [48], HOF [48] and MBH [17]. HOG (histogram of oriented gradients) focuses on static appearance information; HOF (histogram of optical flow) captures the local motion information. For both HOG and HOF. orientations are quantized into 8 bins using full orientations, with an additional zero bin for HOF (i.e., in total 9 bins). Both descriptors are normalized with their L_2 norm.

Optical flow computes the absolute motion, which inevitably includes camera motion [35]. Dalal et al. [17] proposed the MBH (motion boundary histogram) descriptor for human detection, where derivatives are computed separately for the horizontal and vertical components of the optical flow. This descriptor encodes the relative motion between pixels.

The MBH descriptor separates the optical flow field $I_\omega = (I_x, I_y)$ into its x and y component. Spatial derivatives are computed for each of them and orientation information is quantized into histograms, similarly to the HOG descriptor. 8-bin histogram for each component is obtained and each component is normalized separately with the L_2 norm. Since MBH represents the gradient of the optical flow, constant motion information is supressed and only information about changes in the flow field (i.e. motion boundaries) is kept. Compared to video stabilization [35] and motion compensation [79], this is a simple way to eliminate noise due to background motion. This type of descriptor yields excellent results when combined with dense trajectories descriptor. For both HOF and MBH descriptors the dense optical flow that is already computed to extract dense trajectories is re-used, because of efficiency reasons.

ω-flow and ω-descriptors

The approach was proposed by Mihir Jain et al. in [38]. The approach is basically based on a extraction of trajectories from video stream where a separation of a dominant motion and a residual motion at stage of trajectories obtaining and as well as at stage of trajectory description is performed.

The separation as described above, in most cases, will account to distinguishing the impact of camera movement and independent actions. It should be noted that 3D camera motion is not being recovered: The 2D parametric motion model describes the global motion between two frames.

The 2D affine motion model is considered. Simplest motion models such as the 4-parameter model formed by the combination of 2D translation, 2D rotation and scaling, or more complex ones such as the 8-parameter quadratic model (equivalent to a homography), could be used as well. The affine model is a good trade-off between accuracy and and efficiency which is of primary importance when processing a huge video database. It does have limitations, it implies a single plane assumption for the static background. However, this is not that penalizing (especially for outdoor scenes) if differences in depth remain moderated with respect to distance to the camera. The affine flow vector at point $p = (x, y)$ and at time t, is defined as

$$w_{aff}(p_t) = \begin{bmatrix} c_1(t) \\ c_2(t) \end{bmatrix} + \begin{bmatrix} a_1(t) & a_2(t) \\ a_3(t) & a_4(t) \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} \tag{3.27}$$

Horizonal and vertical components of $w_{aff}(p_t)$ are defined

$$u_{aff}(p_t) = c_1(t) + a_1(t)x_t + a_2(t)y_t$$

$$v_{aff}(p_t) = c_2(t) + a_3(t)x_t + a_4(t)y_t$$

The optical flow vector at point p at time t is denoted as

$$w(p_t) = (u(p_t), v(p_t))$$

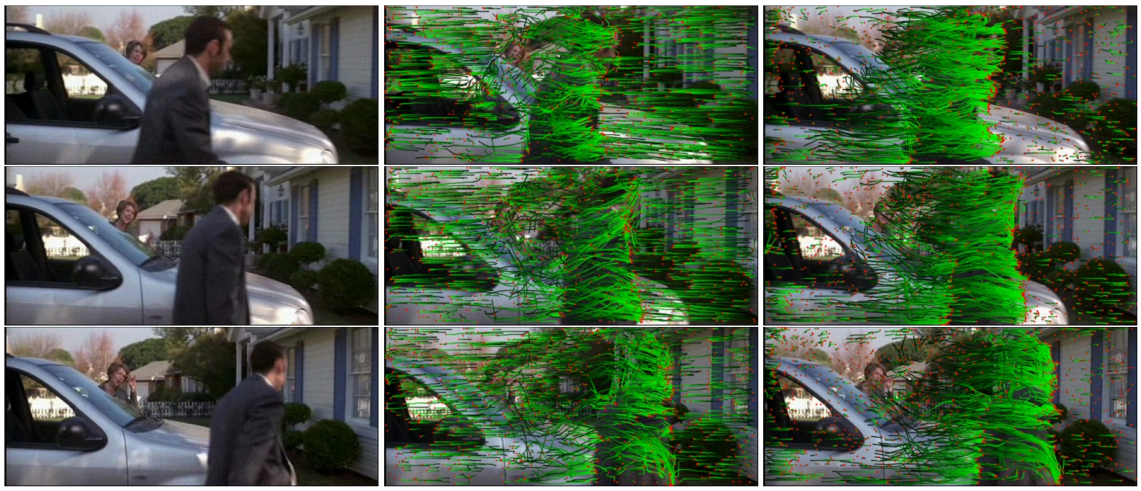
The flow vector $\omega(p_t)$ obtained by removing the affine flow vector from the optical flow vector is defined as

$$\omega(p_t) = w(p_t) - w_{aff}(p_t) \tag{3.28}$$

The dominant motion (which is estimated as $w_{aff}(p_t)$) is usually due to the camera motion. In this case, Equation (3.28) amounts to canceling (or compensating) the camera motion. It should be noted that this is not always truth. For example in case of close-up on a moving actors, the dominant motion will be the affine estimation of the apparent actor motion. The interpretation of the motion compensation output will not be that straightforward in this case, however the resulting ω-field will still exhibit different patterns for the foreground action part and the background part. The compensated flow (as described above) will be referred as ω-flow.

Figure 3.6 illustrates the example videos (Subfigure 3.6a), where a man is moving away from a car and subsequently camera is following walking to the right, thus inducing a global motion to the left in the video. Subfigure 3.6b shows trajectories obtained from optical flow where both camera and scene motion can be observed, this brings noise to the characterization of the current action. In contrast, the ω-trajectories (Subfigure 3.6c) are more active on the actor moving on the foreground, while those localized in the background are now parallel to the time axis enhancing static part of the scene. The ω-trajectories are therefore more relevant for action recognition, since they are more regularly and more exclusively following the actor's motion.

Authors proposed to extract HOG, HOF, Trajectory and MBH descriptors along the compensated trajectories in a similar manner as in [82], these are now called as ω-HOG [38], ω-HOF [38], ω-Trajdesc [38] and ω-MBH [38]. Apart from these a brand new ω-CDS [38] were introduced. Firstly, a number of notes about the compensated descriptors (ω-HOG, ω-HOF, etc.) will be presented, and finally the DCS and ω-DCS descriptors will be introduced.



(a) consecutive frames (b) trajectories from Optical flow (c) ω-trajectories

Figure 3.6: Example of trajectories which are obtained from optical flows (middle column) and from compensated optical flows (right column). The green tail is the trajectory over 15 frames with red dot indicating the current frame. The picture is taken from [38].

HOG captures more context with the modified trajectories. More precisely, the original HOG descriptor is computed from a 2D+t sub volume aligned with the corresponding trajectory and hence represents the appearance along the trajectory shape. When using ω-flow, the video sequence is not aligned. As a result, the ω-HOG descriptor is no more computed around the very same tracked physical point in the space-time volume but around points lying in the patch of the initial feature point, whose size depends on the affine flow magnitude. ω-HOG can be viewed as a "patch-based" computation capturing more information about the appearance of the background or of the moving foreground.

When computing the HOF descriptor the ω-flow impacts on both trajectory computation and the descriptor computation itself. Therefore, HOF can be computed along ω-trajectory and along those trajectories extracted from flow and can encode both kinds of flows (ω-flow or flow). Theoretically, all possible combinations can be used and, in addition to that, versions tracked along one trajectory type can be combined. The ω-HOF descriptor is constructed along a trajectory obtained using ω-flow and it consists of two combined parts. Computation of first part is based on ω-flow and computation of second frame is based on standard optical flow. It has been shown [38] that this configuration improves results' accuracy.

Since MBH is computed from gradient of flow and cancels the constant motion, there is practically no benefit in using ω-flow to compute the MBH descriptors. However, by tracking ω-flow and computing of the MBH descriptors using a standard flow, the performance may be slightly improved [38]. The configuration where the trajectory is computed using the ω-flow and the descriptor is also computed using the ω-flow is referred as ω-MBH descriptor.

Mihir Jain et al. [38] proposed the new Divergence-Curl-Shear descriptor (DCS), which encodes scalar first-order motion features, namely the motion divergence [23], curl [23] and shear [23]. It captures physical properties of the flow pattern that are not involved in the best existing descriptors for action recognition, except in the work of [3] which exploits divergence and vorticity among a set of eleven kinematic features computed from optical

flow.

By kinematic features, local first-order differential scalar quantities, computed on the flow field, are meant. The divergence, the curl (or vorticity) and the hyperbolic terms are considered. They inform on the physical pattern of the flow so that they convey useful information on actions in videos. They can be computed from the first order derivatives of the flow at every point p at every frame t as

$$\begin{cases} \mathrm{div}(p_t) & = \frac{\partial u(p_t)}{\partial x} + \frac{\partial v(p_t)}{\partial y} \\ \mathrm{curl}(p_t) & = \frac{-\partial u(p_t)}{\partial y} + \frac{\partial v(p_t)}{\partial x} \\ \mathrm{hyp}_1(p_t) & = \frac{\partial u(p_t)}{\partial x} - \frac{\partial v(p_t)}{\partial y} \\ \mathrm{hyp}_2(p_t) & = \frac{\partial u(p_t)}{\partial y} + \frac{\partial v(p_t)}{\partial x} \end{cases} \tag{3.29}$$

The divergence is related to axial motion, expansion and scaling effects, the curl to rotation in the image plane. The hyperbolic terms express the shear of the visual flow corresponding to more complex configuration. The shear quantity is only taken into account

$$\mathrm{shear}(p_t) = \sqrt{\mathrm{hyp}_1^2(p_t) + \mathrm{hyp}_2^2(p_t)} \tag{3.30}$$

For computing of the DCS descriptor, the spatial derivatives are computed for the horizontal and vertical components of the flow field. These values are used to compute the divergence, curl and shear scalar values as shown in Equation (3.29) and in Equation (3.30). All possible pairs of kinematic features are considered, namely (divergence, curl), (divergence, shear) and (curl, shear). At each pixel the orientation and magnitude of the 2D vector corresponding to each of these three pairs are computed. The orientation is quantized into histograms and the magnitude is used for weighting, in a similar way as in SIFT [55]. The motivation for this type of encoding is that the joint distribution of kinematic features conveys more information than exploiting them independently.

8-bin histogram for each of the three feature pairs or components of DCS is obtained. The range of possible angles is 2π for the (div, curl) pair and π for the other pairs, because the shear is always positive. The descriptor is computed for a space-time volume aligned with a trajectory as usually.

In order to capture spatio-temporal structure of kinematic features, the volume (32×32 pixels and L = 15 frames) is subdivided into a spatio-temporal grid of size $n_x \times n_y \times n_t$, with $n_x = n_y = 2$ and $n_t = 3$. For each pair of kinematic features, each cell in the grid is represented by a histogram. The resulting local descriptors have a dimensionality equal to $288 = n_x \times n_y \times n_t \times 8 \times 3$.

Similarly as in the case of the ω-MBH descriptor, the ω-DCS descriptor is computed along the ω-flow trajectory using ω-flow values as an input for divergence, curl and shear scalar values computation.

3.3 Dense sampling

In some scenarios the searching for keypoints may be omitted. Especially, in the case when the searching for key points is very time consuming operation and thus, for example, cannot be performed. The searching can be replaced by dense sampling technique. It consists in creating a grid of keypoints across the width and height of the space domain in case of images. For videos, the sampling is also performed at certain uniformly defined frames of the video, thus temporal domain is also sampled in this manner.

The main parameters of dense sampling are the grid specifications. In case of images the grid is defined by the number of skipped pixels in both horizontal and vertical axes of the spatial domain; generally, the number of skipped frames is the same for both axes. In the case of videos the another parameter is added it is the number of skipped frames in temporal domain. The processing can also be applied to each layer of the image pyramid [15](as similarly used in many feature extractors). In this case the additional parameters are the scalling factor between the two layers of the image pyramid and the number of layers of the pyramid. The dense points extraction is performed on each layer of the pyramid and description of the keypoint is performed on the "image" of the corresponding layer.

It should be noted that the number of outputting feature vectors is constant for the same-sized images and video sequences (i.e. the same resolution of video stream and the same number of frames) and for the same settings of the dense sampling. The number of outputting feature vectors is generally higher when compared with key point searching, and therefore, the description process needs to be performed on the higher number of keypoints. From a different point of view this fact increases the extraction time.

3.4 Fixed-sized representation

The number of feature vectors which depict an image or a video sequence is generally dependent on a content of an image or a video sequence and on the type of a local low-level feature extractor. A number of techniques for extraction were presented in sections 3.1 and 3.2; they output a set of feature vectors whose are formally described in Equation 3.1. For better readability the definition is stated again:

$$M(f_{RGB}) = \{(x, y, \tau, D) \mid \mathbf{keypoint}(f_{RGB}, x, y, \tau) = 1 \ \wedge \ D = \mathbf{fextract}(f_{RGB}, x, y, \tau)\} \tag{3.31}$$

The set of local feature vectors can be converted to one representation per image or video. The FE function, which extracts only the corresponding feature vector from the structure which produces the extraction function, it is defines as:

$$FE(\,(x, y, \tau, D)\,) = D \tag{3.32}$$

The dimensionality of the extracted feature vectors is defined as $|FE(M)|$. The model for transformation is defined as

$$C = (c_1, c_2, \ldots, c_n) \tag{3.33}$$

where n represents the number of vectors in the model and dimensionality of each model vector is the same as the dimensionality of the extracted feature vectors, thus $|c_1| = |c_2| = \cdots = |c_n| = |FE(M)|$. For each extracted local feature vector the **assign**(D,C) function is evaluated, The codomain of the function is defined:

$$\mathbf{assign}(D, C) \in (v_1, v_2, \ldots, v_n), \text{ where } \ v_1, v_2, \ldots, v_n \in \langle 0, 1\rangle \tag{3.34}$$

and D represents the investigated local feature vector, C is the model and the number of outputting items n of the function is equal to the number of items of the transformation model.

Therefore, the fixed-sized representation for image function f_{RGB} and feature extractor M and model C can be obtained:

$$BOW(f_{RGB}, M, C) = \sum_{i \in FE(M(f_{RGB}))} \mathbf{assign}(i, C) \tag{3.35}$$

This representation is called Bag-Of-Words [74, 82, 38, 48] and is suitable for image classification and action recognition [82, 38, 48].

Chapter 4

Feature processing

The feature representation of an image or a video sequence needs to be processed and decided whether the wanted situation or object or action is happening or is detected in. This can be achieved for example by using a processing pipeline which is presented in Section 4.1. The processing contains two main processing blocks, it is the conversion to fixed-sized description and the classifier unit which are described in Sections 4.2 and 4.3.

Especially, in the field of action recognition, and when the local space-time features are used, the extended version of such processing is needed to be used to provide the state-of-the-art classification performance [82, 48, 62]. The processing is shown and described in Section 4.4.

4.1 Basic pipeline schema

Data transformation diagram is shown in Figure 4.1. The input entity (an image or a video) is (from left to right according to Figure 4.1) processed and its feature representation is obtained in the *Feature extract* box [82, 88]. This box outputs a various number of feature vectors, those are used in the *Vocabulary search* box. This block produces a set of nearest clusters related to particular input feature vector. From these sets a bag-of-words representation is constructed in the *Hist* box. The Bag-of-words [74] representation is one fixed-sized feature representation of input entity (image or video). This representation is used as an input to the *classifier box* which decides the type (class) of the input entity.

The above mentioned procedure requires a visual vocabulary creation, only then the whole processing is possible. Visual vocabulary is created (as shown in the upper part of Figure 4.1) from entities (images or videos) of the training set of a dataset. Those are transformed to feature vectors using the same *feature extract* box with the same settings as the one which is described above. The output feature vectors define the feature space which needs to be quantized and modelled using a clustering algorithm. The *Clustering* box constructs a visual vocabulary (or by other words, the quantized representation of its input feature space) which is latter used for bag-of-words representation creation.

I would like to present a theoretical usage of the described pipeline; please, consider a situation where a set of real-world entities exists and the need of distinguishing among them exists. The initial conditions are

(i) An algorithm M which generates a feature representation of the real-world entity exists and is applicable to each type of entity.

(ii) A number of wanted entities is collected and a number of unwanted (other) entities (counter examples) is collected as well. All these entities are collected in a dataset D and are split into two main parts: the training one (D_{TR}) and the testing one (D_{TE}). Formally, the dataset is defined as

$$D = \{D_{TR}, L_{TR}, D_{TE}, L_{TE}\} \tag{4.1}$$

where L_{TR} and L_{TE} are the sets of labels for both parts D_{TR} and D_{TE} and the number of items in label set and corresponding input images set is equal; this condition is compulsory for all (in this case both) doublets in the dataset.

$$\begin{aligned} D_{TR} &= \{f_{RGB_1}, \ldots, f_{RGB_r}\} \\ D_{TE} &= \{f_{RGB_1}, \ldots, f_{RGB_s}\} \end{aligned} \tag{4.2}$$

represents all the image functions in the dataset and

$$\begin{aligned} L_{TR} &= \{l_1, \ldots, l_r\} \\ L_{TE} &= \{l_1, \ldots, l_s\} \end{aligned} \tag{4.3}$$

are the corresponding labels for each sample in the dataset; $l_1, \ldots, l_r$ and $l_1, \ldots, l_s \in \{+1, -1\}$.

Using the presented pipeline (from Figure 4.1), the recognition system can be built according to the following procedure:

(1) The training entities are used for the visual vocabulary creation (as depicted in the upper part of Figure 4.1) using feature extraction algorithm M and the clustering algorithm.

$$\begin{gathered} M(D_{TR}) \longrightarrow C : C = \textbf{clustering}(\{M(f_{RGB_1}), \ldots, M(f_{RGB_r})\}, P_c); \\ C = \{c_1, \ldots, c_n\}; \\ m_k \in M(f_{RGB_i}), |FE(m_k)| = |c_j| \\ \text{for all } m_k \in M(f_{RGB_i}) \text{ and for all } i \in \{1, \ldots, r\} \text{ and } j \in \{1, \ldots, n\} \end{gathered} \tag{4.4}$$

where $c_1, \ldots, c_n$ represent vectors of the resulting vocabulary, $FE(m_k)$ extracts only the descriptor vector as suggested in Equation (3.32), P_c represents a set of parameters of the clustering algorithm and everywhere $|x|$ represents the dimensionality of the vector x. The **clustering** procedure usually internally removes the all unnecessary information about the location of the input feature vectors.

(2) All entities in the training part of the dataset are transformed to its bag-of-words representation as suggested in the pipeline

$$\begin{gathered} \{M(D_{TR}), C\} \longrightarrow B_{TR} : \; B_{TR} = \{B_1, \ldots, B_r\}; \; |C| = |B_i| \text{ for each } i \in \{1, \ldots, r\}; \\ B_i = \textbf{bow}(M(f_{RGB_i}), C, P_b) \text{ for each } i \in \{1, \ldots, r\} \end{gathered} \tag{4.5}$$

and are subsequently used for training of the classifier, the training of the classifier outputs the model Z.

$$\{B_{TR}, L_{TR}\} \longrightarrow Z : Z = (z_1, \ldots, z_a) | z_a \in \langle 0, 255 \rangle \wedge z_a \in \mathbb{N}; \\ Z = \textbf{train}(\{B_1, \ldots, B_r\}, \{L_1, \ldots, L_r\}, P_t) \tag{4.6}$$

The terms P_b and P_t are the parameters' sets of the related functions, Z simply represents a variable-sized vector of bytes in a computer.

(3) All entities in the testing part of the dataset are transformed to its bag-of-words representation using presented pipeline

$$\{M(D_{TE}), C\} \longrightarrow B_{TE} : \; B_{TE} = \{B_1, \ldots, B_r\}; \; |C| = |B_i| \text{ for each } i \in \{1, \ldots, r\}; \\ B_i = \textbf{bow}(M(f_{RGB_i}), C, P_b) \text{ for each } i \in \{1, \ldots, r\} \tag{4.7}$$

and latter are used for testing of the classifier, the testing procedure can be considered as a transformation from bag-of-words feature vector to classifier's response (generally a real number).

$$\{B_{TE}, Z\} \longrightarrow R : R = \{R_1, \ldots, R_s\} | R_i \in \mathbb{R} \text{ for each } i \in \{1, \ldots, s\}; \\ R = \textbf{test}(\{B_1, \ldots, B_s\}, Z, P_t) \tag{4.8}$$

The terms P_b and P_t are the parameters' sets of the related functions.

(4) All testing responses are used for measuring the classifier's quality.

$$R \longrightarrow Q : \; R = \{R_1, \ldots, R_s\}; \; R_i \in \mathbb{R} \text{ for each } i \in \{1, \ldots, s\}; \; Q \in \langle 0, 1 \rangle \wedge Q \in \mathbb{R} \tag{4.9}$$

Algorithm 1: Basic pipeline algorithm

The processing pipeline has two main parts. The first one covers the conversion of the output of the feature extract box and produces the bag-of-words representation, it is mainly the *Clustering* box and the *Hist* box. The methods which may be used for these purposes are presented in Section 4.2. The second part is the whole *Classifier* box; the possible methods are presented in Section 4.3.

4.2 Fixed-sized descriptor creation

Fixed-sized descriptor is created by using two independent processes. First one is a searching of a vocabulary and second one is to use the nearest codewords to produce a histogram of occurencies which serves as an output representation. In this section multiple methods for creation of such vocabulary are listed and widely used one is presented. The bag-of-words unit will be presented more in detail.

The visual vocabulary can be created by using clustering algorithm, the various methods such as the subsequent ones exist

- Hierarchical clustering [29],
- k-means [29, 89],

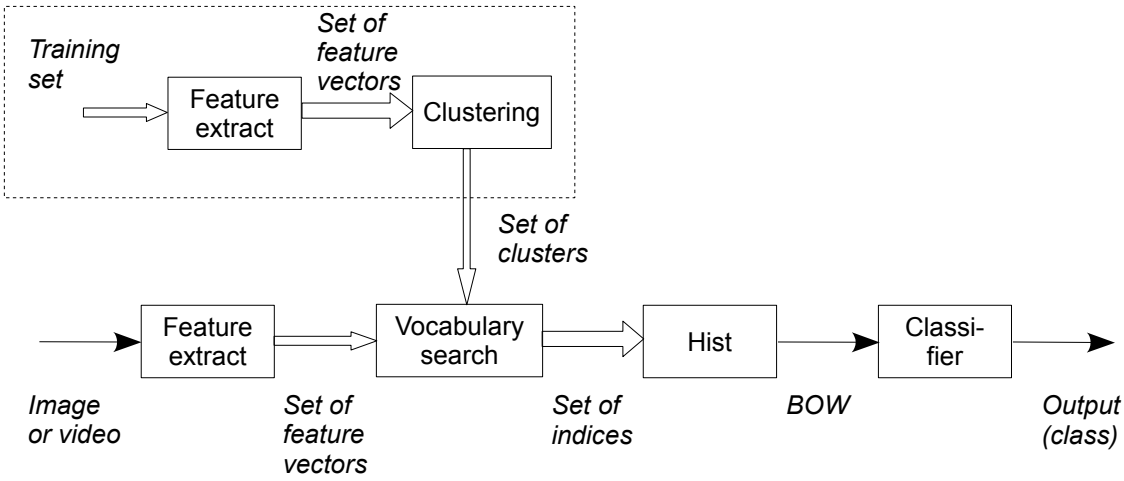


Figure 4.1: Processing pipeline scheme

- Voronoi tesselation [19],
- etc.

The widely used method in such a feature space quantization is the k-means method, which is more in detail described in subsequent subsection. The last subsection presents the principle of the bag-of-words method.

k-means

Let $z_1, z_2, \ldots$ be a random sequence of points (vectors) in E_n, each point being selected independently of the preceding ones using a fixed probability measure p. Thus $P[z_a \in A] = p(A)$ and $P[z_{n+1} \in A | z_1, z_2, \ldots, z_n] = p(A), n = 1, 2, \ldots$, for A any measurable set in E_n. Relative to given k-tuple $x = (x_1, x_2, \ldots, x_k), x_i \in E_n, i = 1, 2, \ldots, k$, a minimum distance partition $S(x) = \{S_1(x), S_2(x), \ldots, S_k(x)\}$ of E_n is defined by

$$\begin{aligned} &S_1(x) = T_1(x), S_2(x) = T_2(x)S_1^{'}(x), \ldots, \\ &S_k(x) = T_k(x)S_1^{'}(x)S_2^{'}(x)S_{k-1}^{'}(x) \end{aligned} \tag{4.10}$$

where

$$T_i(x) = \{\xi : \xi \in E_n, |\xi - x_i| \leq |\xi, x_j|, j = 1, 2, \ldots, k\}. \tag{4.11}$$

The set $S_i(x)$ contains the points in E_n nearest to x_i, with tied points being assigned arbitrarily to the set of lower index. It should be noted that with this convention concerning tied points, if $x_i = x_j$ and $i < j$ then $S_j(x) = \varnothing$. Sample k-means $x^n = (x_1^n, x_2^n, \ldots, x_k^n), x_i^n \in E_n, i = 1, \ldots, k$, with associated integer weights $(w_1^n, w_2^n, \ldots, w_k^n)$, are now defined as follows: $x_i^1 = z_i, w_i^1 = 1, i = 1, 2, \ldots, k$, and for $n = 1, 2, \ldots,$ if$z_{k+n} \in S_i^n, x_i^{n+1} = (x_i^n w_i^n + z_{n+k})/(w_i^n + 1), w_i^{n+1} = w_i^n + 1$ and $x_j^{n+1} = x_j^n, w_j^{n+1} = w_j^n$ for $i \neq j$, where $S^n = S_1^n, S_2^n, \ldots, S_k^n$ is the minimum distance partition relative to x^n.

Stated informally, the k-means procedure consists of simply starting with k groups each of which consists of a single random point, and thereafter adding each new point to the group whose mean the new point is nearest. After a point is added to a group, the mean of that group is adjusted in order to take account of the new point. Thus at each stage the k-means are, in fact, the means of the groups they represent (hence the term k-means).

In studying the asymptotic behaviour of the k-means, the convenient assumptions were made,

(i) p is absolutely continuous with respect to Lebesgue measure [76] on E_n, and

(ii) $p(R) = 1$ for a closed and bounded convex set $R \subset R$.

For a given k-tuple $x = (x_1, x_2, \ldots, x_k)$ — such an entity being referred to hereafter as a k-point — let

$$\begin{aligned} W(x) &= \sum_{i=1}^{k} \int_{Si} |z - x_i|^2 \, dp(z), \\ V(x) &= \sum_{i=1}^{k} \int_{Si} |z - u_i(x)|^2 \, dp(z), \end{aligned} \tag{4.12}$$

where $S = \{S_1, S_2, \ldots, S_k\}$ is the minimum distance partition relative to x,
and $u_i(x) = \int_{S_i} z dp(z)/p(S_i)$ or $u_i(x) = x_i$, according to whether $p(S_i) > 0$ or $p(S_i) = 0$. If $x_i = u_i(x), i = 1, 2, \ldots, k$ we say the k-point is unbiased

The principal result is as follows.

(i) The sequence of random variables $W(x^1), W(x^2), \ldots$ converges and $W_\infty = \lim_{n\to\infty} W(x^n)$ is equal to $V(x)$ for some x in the class of k-points $x = (x_1, x_2, \ldots, x_k)$ which are unbiased, and have the property that $x_i \neq x_j$ if $i \neq j$.

(ii) Let $u_i^n = u_i(x^n)$ and $p_i^n = p(S_i(x^n))$; then

$$\sum_{n=1}^{m} \left(\sum_{n=1}^{k} p_i^n \, |x_i^n - u_i^n| \right) \Big/ m \to 0 \text{ as } m \to \infty. \tag{4.13}$$

The aforementioned Theorems can be proved, more information can be found, for example, in [37, 57, 70].

In a number of cases covered by (i) in the preceding list, all the unbiased k-points have the same value of W. In this situation, (i) implies $\sum_{i=1}^{k} p_i^n \, |x_i^n - u_i^n|$ converges to zero. An example is provided by the uniform distribution over a disk i E_2. If $k = 2$, the unbiased k-point (x_1, x_2) with $x_1 \neq x_2$ consist of the family of points x_1 and x_2 opposite one another on a diameter, and at a certain fixed distance from the center of the disk. (There is one unbiased k-point with $x_1 = x_2$, both x_1 and x_2 being at the center of the disk in this case.) The k-means thus converge to some such relative position, but (i) does not quite permit us to eliminate the interesting possibility that the two means oscillate slowly but indefinitely around the center.

(i) provides the convergence of $\sum_{i=1}^{k} p_i^n \, |x_i^n - u_i^n|$ to zero in a slightly broader class of situations. This is where the unbiased k-point $x = (x_1, x_2, \ldots, x_n)$ with $xi \neq x_j$ for $i \neq j$, are all stable in the sense that for each such x, $W(y) \geq W(x)$ (and hence $V(y) \geq V(x)$) for all y in a neighborhood of x. In this case, each such x falls in one of finitely many equivalence classes such that W is constant on each class. This is illustrated by the above example, where there is only a single equivalence class. If each of the equivalence classes contains only a single point, (i) implies convergence of x^n to one of those points.

K-means methos has two main parameters, the number of output vectors (clusters) and the number of iterations performed by the algorithm, these two parameters need to be carefully set while used in some experiments.

Bag Of Words model

The very basic version of the bag-of-words function (as formally suggested in Equation 4.7) is to remove every information about the location of the keypoint and simply obtain the list of words in the vocabulary which are the closest ones to some of input feature vectors. This list is then converted to output binary vector, where bins with the indexes of the words in the list are set to 1. The transformation can be formally described as follows

$$\begin{gathered} \{M(f_{RGB}), C\} \longrightarrow H : \; H = (h_1, \ldots, h_n); \; C = \{c_a, \ldots, c_n\}; \\ (\text{for each } m \in M(f_{RGB}) : \; i = \textbf{closest}(FE(m), C) \wedge \text{if } (h_i == 0) \Rightarrow h_i = 1) \end{gathered} \tag{4.14}$$

where function **closest** returns index of the closest word in the vocabulary.

Other option is to count the number of occurencies and fill up the output histogram directly with the number of occurencies for each word in the vocabulary.

$$\begin{gathered}\{M(f_{RGB}), C\} \longrightarrow H : H = (h_1, \ldots, h_n);\, C = \{c_a, \ldots, c_n\};\\ (\text{for each } m \in M(f_{RGB}) : i = \mathbf{closest}(FE(m), C) \wedge h_i = h_i + 1)\end{gathered} \tag{4.15}$$

The above presented versions are called the set-of-words method and bag-of-word method with a hard assignment, respectively. The main disadvantage of the hard assignment version is that only slightly different input local feature vectors may be accumulated into totally different output histogram bins (nearest codewords are different); this may cause total dissimilarity of two similar input vectors.

The above issue is addressed in the soft assignment approach; the soft assignment is performed as follows. A small group of the clusters very close to the vector being processed is retrieved instead of a single cluster. A weight is assigned to each cluster in such group; the weight corresponds to the closeness to the vector being processed. Finally, each clusters' weight is added to relevant bin of the output histogram. Formally, the situation can be described as follows

$$\begin{gathered}\{M(f_{RGB}), C\} \longrightarrow H : H = (h_1, \ldots, h_n);\, C = \{c_1, \ldots, c_n\};\\ (\text{for each } m \in M(f_{RGB}) : W = \mathbf{assign}(FE(m), C, P_a) \wedge H = H \oplus W | W = (w_1, \ldots, w_n) \wedge\\ H \oplus W = (h_1 + w_1, \ldots, h_n + w_n))\end{gathered} \tag{4.16}$$

where **assign** function returns the partial histogram with the same size as the output histogram, where the affected bins have the nonzero values and P_a is the set of parameters for the assign function. Common parameter for assign functions is the number of searched closest vectors, which will be affected by single call of the **assign** function. The **assign** function can be described formally as follows

(i) Firstly, the closest vectors need to be searched

$$\{F, C, P_a\} \longrightarrow Z : Z = \{(z_1, i_1), \ldots, (z_c, i_c)\}; Z = \mathbf{search}(F, C, P_a); \tag{4.17}$$

The **search** function returns the doublets with the distance to the visual word and the index of the word in the vocabulary, c is the number of returned closest vectors.

(ii) Secondly, a weighting function **f** is applied

$$\begin{gathered}Z \longrightarrow w : Z = \{(z_1, i_1), \ldots, (z_c, i_c)\};\, w = \{(w_1, i_1), \ldots, (w_c, i_c)\};\\ \text{for each } (z_j, i_j) \in Z : (w_j, i_j) = (\mathbf{f}, i_j)\,;\end{gathered} \tag{4.18}$$

(iii) Finally, the output partial histogram is constructed

$$\begin{gathered}w \longrightarrow W : \; w = \{(w_1, i_1), \ldots, (w_c, i_c)\};\\ W = (W_1, W_2, \ldots, W_n),\ \text{initially } W_a = 0 \text{ for each } a \in \{1, \ldots n\};\\ \text{and then for each } (w_j, i_j) \in w : W_{(i_j)} = w_j;\end{gathered} \tag{4.19}$$

The output vector W has the same dimensionality n as the visual vocabulary and c bins in the output partial histogram are affected by the content of the input w.

Several weighting strategies can be applied. The basic one

$$\begin{gathered}Z \longrightarrow w : Z = \{(z_1, i_1), \ldots, (z_c, i_c)\};\, w = \{(w_1, i_1), \ldots, (w_c, i_c)\};\\ \text{for each } (z_j, i_j) \in Z : (w_j, i_j) = \left(\left(\frac{\mathbf{min}(Z)}{z_j}\right), i_j\right)\,;\end{gathered} \tag{4.20}$$

is computed as the ratio between the distance of the descriptor point to its c nearest visual words, where the $\mathbf{min}(Z)$ function returns the minimal value of the distance value of all tuples in Z and

z_j is the currently processed distance. This behaviour can be simply described as the closest vector has the weight of 1 and the other ones have the weight lower than 1 depending on the distance.

Jiang et al. [39] use the sorted list of c nearest visual words and design the weighting function based on the rank of the visual word in this list. Transformation shown in Equation (4.18) can be updated as follows

$$\begin{aligned} Z \longrightarrow w : \; & Z = \{(z_1, i_1), \ldots, (z_c, i_c)\}; \; w = \{(w_1, i_1), \ldots, (w_c, i_c)\}; \\ & \text{for each } (z_j, i_j) \in Z : (w_j, i_j) = \left(\left(\tfrac{1}{2^{\mathbf{rank}(Z,j)-1}}\right), i_j\right); \end{aligned} \tag{4.21}$$

where function $\mathbf{rank}(Z, j)$ returns the the rank of the visual word l in the distance item in Z. The list is ordered by the distances.

Philbin et al. [61] use the idea, that is usual in soft-assignment (for example in estimating Gaussian Mixture Models), that the weight assigned to a visual word is an exponential function of the distance to the cluster center thus Equation (4.18) can be updated as follows

$$\begin{aligned} Z \longrightarrow w : \; & Z = \{(z_1, i_1), \ldots, (z_c, i_c)\}; \; w = \{(w_1, i_1), \ldots, (w_c, i_c)\}; \\ & \text{for each } (z_j, i_j) \in Z : (w_j, i_j) = \left(\left(e^{-\frac{z_j^2}{2\sigma^2}}\right), i_j\right); \end{aligned} \tag{4.22}$$

The σ should be chosen so that a substantial weight is only assigned to a small number of visual words. The σ value for SIFT descriptors has been experimentally evaluated and suggested in [61]. It should be noted that the σ value is highly dependent not only on used feature description method but also on data domain.

The above presented approaches do not take into account the positions where the individual feature vectors were obtained. This can be used and the input local feature vectors can be split into groups according to its position in an image. The bag-of-words representation is computed for each group, and individual vectors are concatenated in predefined order. This approach may improve the over-all results of the designed recognition system.

When the spatio-temporal features are used, not only the position in the space domain but also the position in the temporal domain are known. In a similar matter the temporal domain can be splitted accordingly.

4.3 Classification methods

A classifier is a procedure that accepts a set of features and produces a class label for them. There could be two, or many, classes, though it is usual to produce multi-class classifiers out of two-class classifiers. Classifiers are built by taking a set of labeled examples and using them to come up with a rule that assigns a label to any new example. In general, we have a training dataset (x_i ,y_i); each of the feature vectors x_i consists of measurements of the properties of different types of object, and the y_i are labels giving the type of the object that generated the example. Classifiers are a crucial tool in high-level vision, action recognition, etc., because many problems can be abstracted in a form that looks like classification.

Generally, variety of classification algorithms exist. In computer vision or pattern recognition field are commonly used Neural Networks, Support Vector Machines and AdaBoost algorithms these days. The Neural Networks algorithm is on the decline when used purely as the general classifier of feature vectors. The AdaBoost algorithm is widely used for detection and localization of objects in the images. It is based on the composing of a weak classifiers or hypotheses into a boosted (strong) classifier.

Support Vector machines are today the most frequently used algorithms in the processings such as presented in Section 4.1 when fixed-sized representation of input objects needs to be classified. It has been shown that support vector machines reach the state-of-the-art performance in action recognition [82, 38, 83].

Neural networks are today widely used in feature learning procedures, where the feature extractors are created automatically for a given purpose. The algorithm is also called deep learning [8].

Generally can be said, that such feature extractors creating requires huge amount of input images or videos. The action recognition using deep learning is also presented in [4, 49].

Neural Networks

The concept of an artifical neural network that could be useful for pattern recognition started in the 1950s. For example, Bledsoe and Browning [11] developed "n-tuple" type of classifier that involved bit-wise recording and lookup of binary feature data, leading to the "weightless" or "logical" type of ANN.

The simple perceptron is a linear classifier that classifies patterns into two classes. It takes a feature vector $x = (x_1, x_2, \ldots, x_N)$ as its input, and produces a single scalar output $\sum_{i=1}^{N} w_i x_i$, the classification process being completed by applying a threshold function at θ. The mathematics is simplified by writing $-\theta$ as w_0, and taking it to correspond to an input x_0 that is maintained at a constant value of unity. The output of the linear part of the classifier is then written in the form:

$$d = \sum_{i=1}^{N} w_i x_i - \theta = \sum_{i=1}^{N} w_i x_i + w_0 = \sum_{i=0}^{N} w_i x_i \tag{4.23}$$

and the final output of the classifier is given by:

$$y = f(d) = f\left(\sum_{i=0}^{N} w_i x_i\right) \tag{4.24}$$

This type of neuron can be trained using a variety of procedures, such as the *fixed increment rule* [87]. The basic concept of this algorithm was to try to improve the overall error rate by moving the linear discriminant plane a fixed distance toward a position where no misclassification would occur – but only doing this when a classification error had occured:

$$w_i(k+1) = w_i(k) \quad y(k) = \omega(k) \tag{4.25}$$

$$w_i(k+1) = w_i(k) + \eta[\omega(k) - y(k)]x_i(k) \quad y(k) \neq \omega(k) \tag{4.26}$$

In these equations, th parameter k represents the kth iteration of the classifier and $\omega(k)$ is the class of the kth training pattern. It is clearly important to know whether this training scheme is effective in practice. In fact, it is possible to show that if the algorithm is modified so that its main loop is applied sufficiently many times, and if the feature vectors are linearly separable, then the algorithm will converge to a correct error-free solution.

Unfortunately, most sets of feature vectors are not linearly separable. Thus, it is necessary to find an alternative procedure for adjusting the weights. This is achieved by the Widrow-Hoff delta rule [34], which involves making changes in the weights in proportion to the error $\delta = \omega - d$ made by the classifier. It should be noted that the error is calculated before thresholding to determine the actual class, i.e., δ is calculated using d rather than $f(d)$. Thus, we obtain the Widrow-Hoff delta rule in the form:

$$w_i(k+1) = w_i(k) + \eta\delta x_i(k) = w_i(k) + \eta[\omega(k) - y(k)]x_i(k) \tag{4.27}$$

There are two important ways in which the Widrow-Hoff rule differs from the fixed increment rule:

1. An adjustment is made to the weights whether or not the classifier makes an actual classification error.
2. The output function d used for training is different from the function $y = f(d)$ used for testing.

These differences underline the revised aim of being able to cope with nonlinearly separable feature data. However, the fixed increment rule is not designed to cope with nonseparable data and results in instability during training and instability to arrive at an optimal solution. On the other hand, the Widrow-Hoff rule copes satisfactorily with this type of data. An interesting addendum

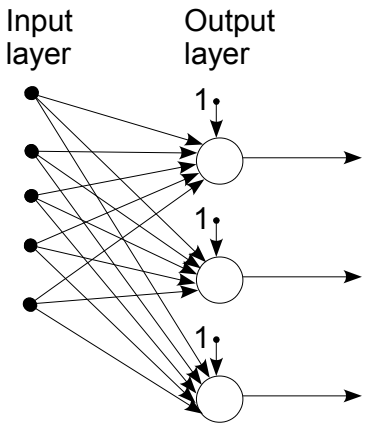


Figure 4.2: Single-layer perceptron. The single-layer perceptron employs a number of simple perceptrons in a single layer. Each output indicates a different class (or region of feature space).

is that although the fixed increment rule apparently reaches an optimal solution, the rule becomes "complacent" once a zero error situation has occured, whereas an ideal classifier would arrive at a solution that minimizes the probability of error. Clearly, the Widrow-Hoff rule goes some way to solving this problem.

It has been considered what can be achieved by a simple perceptron. Clearly, although it is only capable of dichotomizing feature data, a suitably trained array of simple perceptrons – the single layer perceptron of Figure 4.2 – should be able to divide feature space into a large number of subregions bounded (in multidimensional space) by hyperplanes. However, in a multiclass application, this approach would require a very large number of simple perceptrons – up to ${}^cC_2 = \frac{1}{2}c(c-1)$ for a c-class system. Hence, there is a need to generalize the approach by other means. In particular, multilayer perceptron (MLP) networks – which would emulate the neural networks in the brain – seem poised to provide a solution since they should be able to recode the outputs of the first layer of simple perceptrons.

Rosenblatt [65] himself proposed such networks, but was unable to propose general means for training them systematically. In 1969, Minsky and Papert [58] published their famous monograph, and in discussing the MLP raised the specter of "the monster of vacuous generality"; they drew attention to certain problems that apparently would never be solved using MLPs. For example, diameter-limited perceptrons (those that view only small regions of an image within a restricted diameter) would be unable to measure large-scale connectedness within images. These considerations discouraged effort in this area, and for many years attention was diverted to other areas such as expert systems. It was not until 1986 that Rumelhart et al. [67] were successful in proposing a systematic approach to the training of MLPs. Their solution is known as the back-propagation algorithm.

The problem of training an MLP can be simply stated: a general layer of an MLP obtains its feature data from the lower layers and receives its class data from higher layers. Hence, if all the weights in the MLP are potentially changeable, the information reaching a particular layer cannot be relied upon: there is no reason why training a layer in isolation should lead to overall convergence of the MLP toward an ideal classifier (however defined). In addition, it is not evident what the optimal MLP architecture should be. While it might be thought that this is a rather minor difficulty, in fact this is not so: indeed, this is but one example of the so-called "credit assignment problem".

One of the main difficulties in predicting the properties of MLPs and hence of training them reliably is the fact that neuron outputs swing suddenly from one state to another as their inputs change by infinitesimal amounts. Hence, we might consider removing the thresholding functions from the lower layers of MLP networks to make them easier to train. Unfortunately, this would result in these layers acting together as larger linear classifiers, with far less discriminatory power than

initialize weights with small random numbers;
select suitable value of learning rate coefficient η in the range 0-1;
do {
    for all patterns in the training set {
        for all nodes j in the MLP {
            obtain feature vector x and target output value t;
            compute MLP output y;
            if (node is in output layer)
                $\delta_j = y_j(1 - y_j)(t_j - y_j)$
            else
                $\delta_j = y_j(1 - y_j)(\sum_m \delta_m W_{jm})$
            adjust weights i of node j according to $w_{ij} = w_{ij} + \eta \delta_j y_i$
        }
    }
} **until** changes are reduced to some predetermined level;

Figure 4.3: Pseudocode of the Back-Propagation Algorithm

the original classifier (in the limit we would have a single linear classifier with a single thresholded output connection, so the overall MLP would act as a single-layer perceptron).

The key to solving these problems was to modify the perceptrons composing the MLP by giving them a less "hard" activation function than the Heaviside function. As we have seen, a linear activation function would be of little use, but one of "sigmoid" shape, such as the tanh function, is effective, and indeed is almost certainly the most widely used of the available functions. Once these softer activation functions were used, it became possible for each layer of the MLP to "feel" the data more precisely and thus training procedures could be set up on a systematic basis. In particular, the rate of change of the data at each individual neuron could be communicated to other layers which could then be trained appropriately – though only on an incremental basis. We shall not go through the detailed mathematical procedure, or proof of convergence, beyond stating that it is equivalent to energy minimization and gradient descent on a (generalized) energy surface. Instead, we give an outline of the backpropagation algorithm in Figure 4.3. Nevertheless, some notes on the algorithm are in order:

1. The outputs of one node are the inputs of the next, and an arbitrary choice is made to label all variables as output (y) parameters rather than as input (x) variables; all output parameters are in the range 0 to 1.
2. The class parameter ω has been generalized as the target value t of the output variable y.
3. For all except the final outputs, the quantity δ_j has to be calculated using the formula $\delta_j = y_j(1 - y_j)(\sum_m \delta_m w_{jm})$, the summation having to be taken over all the nodes in the layer above node j.
4. The sequence for computing the node weights involves starting with the output nodes and then proceeding downward one layer at a time.
5. If there are no hidden nodes, the formula reverts to the Widrow-Hoff delta rule, except that the input parameters are now labeled y_i, as indicated above.
6. It is important to initialize the weights with random numbers to minimize the chance of the system becoming stuck in some symmetrical state from which it might be difficult to recover.
7. Choice of value for the learning rate coefficient η will be a balance between achieving a high rate of learning and avoidance of overshoot: normally a value of around 0.8 is selected.

When there are many hidden nodes, convergence of the weights can be very slow, and indeed this is one disadvantage of MLP networks. Many attempts have been made to speed convergence, and a method that is almost universally used is to add a "momentum" term to the weight update formula, it being assumed that weights will change in a similar manner during iteration k to the change during iteration $k-1$:

$$w_{ij}(k+1) = w_{ij}(k) + \eta\delta_j y_i + \alpha[w_{ij}(k) - w_{ij}(k-1)] \tag{4.28}$$

where α is the momentum factor. This technique is primarily intended to prevent networks becoming stuck at local minima of the energy surface.

SVM

Support vector machines method is a supervised learning method widely used in computer vision, chemistry, matematics, etc. Generally, it is based on searching of a optimal separating hyperplane between the d-dimensional vectors (input samples) which are labelled into two classes. The training process creates a svm model using labels and feature points, testing procedure "labels" each unknown testing sample by response which indicates the class current sample belongs to.

The following description is structured into multiple sections. Firstly, the optimal separation hyperplane is matematically defined; Secondly, the separation hyperplane is defined in a situation when the input data are not separable. Finally, the support vector machine is constructed and existence of kernel function is provided and the description of widely used kernel functions is shown.

The Optimal hyperplane

Let us determine that two finite subsets of vectors x from the training set

$$(y_1, x_1), \ldots, (y_\ell, x_\ell), \qquad x \in \mathbb{R}^n,\ y \in \{-1, 1\} \tag{4.29}$$

one subset I for which $y = 1$, and the another one II for which $y = -1$ are separable by the hyperplane

$$(x * \phi) = c \tag{4.30}$$

if there exist both a unit vector $\phi(|\phi| = 1)$ and a constant c such that the inequalities

$$\begin{aligned} (x_i * \phi) &> c, \qquad \text{if } x_i \in I \\ (x_j * \phi) &< c, \qquad \text{if } x_j \in II \end{aligned} \tag{4.31}$$

hold true where we denoted by $(a * b)$ the inner product between vectors a and b.

Let us determine for any unit vector ϕ two values

$$\begin{aligned} c_1(\phi) &= \underset{x_i \in I}{max}(x_i * \phi), \\ c_1(\phi) &= \underset{x_j \in I}{min}(x_j * \phi). \end{aligned} \tag{4.32}$$

Consider the unit vector ϕ_0 which maximizes the function

$$\rho(\phi) = \frac{c_1(\phi) - c_2(\phi)}{2}, \qquad |\phi| = 1 \tag{4.33}$$

under the condition that inequalities (4.31) are satisfied. The vector ϕ_0 and the constant

$$c_0 = \frac{c_1(\phi_0) + c_2(\phi_0)}{2} \tag{4.34}$$

determine the hyperplane that separates vectors $x_1, \ldots, x_a$ of the subset I from $x_1, \ldots, x_a$ of the subset II, $(a + b = \ell)$ and has the maximal margin (4.33). We call this hyperplane the *maximal margin hyperplane* or the *optimal hyperplane*.

A theorem with the following definition exists: *The optimal hyperplane is unique.* Proof of the Theorem exceeds the need of this text and the persons concerned can found it for example in [81].

The goal is to have an effective methods for constructing the optimal hyperplane. To do so we consider an equivalent statement of the problem: Find a pair consisting of a vector ψ_0 and a constant (threshold) b_0 such that they satisfy the constraints

$$\begin{array}{l}(x_i * \psi_0) + b_0 \geq 1, \\ (x_j * \psi_0) + b_0 \leq -1,\end{array} \tag{4.35}$$

and the vector ψ_0 has the smallest norm

$$|\psi|^2 = (\psi * \psi). \tag{4.36}$$

A theorem with the following definition exists: *Vector ψ_0 that minimizes (4.36) under constraints (4.35) is related to the vector that forms the optimal hyperplane by the equality*

$$\phi_0 = \frac{\psi_0}{|\psi_0|}. \tag{4.37}$$

The margin ρ_0 between the optimal hyperplane and separated vectors is equal to

$$\rho(\phi_0) = \sup_{\phi_0} \frac{1}{2} \left(\min_{i \in I} (x_i * \phi_0) - \max_{i \in II} (x_i * \phi_0) \right) = \frac{1}{|\psi_0|}. \tag{4.38}$$

Proof of the Theorem above exceeds the need of this text and the persons concerned can found it for example in [81].

Thus the vector ψ_0 with the smallest norm satisfying constraints (4.35) defines the optimal hyperplane. The vector ψ_0 with the smallest norm satisfying constraints (4.35) with $b = 0$ defines the optimal hyperplane passing through the origin.

To simplify preceding notation, rewriting of the constraint (4.35) in the equivalent form is needed

$$y_i((x_i * \psi_0) + b) \geq 1, \quad i = 1, \ldots, \ell. \tag{4.39}$$

Therefore in order to find the optimal hyperplane the following quadratic optimization has to be solved: To minimize the quadratic form (4.36) subject to the linear constraints (4.39).

The problem can be solved in the primal space – the space of parameters ψ and b. However, the deeper results can be obtained by solving this quadratic optimization problem in the *dual space* – the space of Lagrange multipliers. Below this type of solution will be considered.

In order to solve this quadratic optimization problem, the saddle point of the Lagrange function [9] has to be found

$$L(\psi, b, \alpha) = \frac{1}{2}(\psi * \psi) - \sum_{i=1}^{\ell} \alpha_i (y_i[(x_i * \psi) + b] - 1), \tag{4.40}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. To find the saddle point that function has to be minimized over ψ and b and has to be maximized over the nonnegative Lagrange multipliers $\alpha_i \geq 0$.

According to the Fermat theorem [81], the minimum points of this functional have to satisfy the conditions

$$\begin{array}{l}\frac{\partial L(\psi,b,\alpha)}{\partial \psi} = \psi - \sum_{i=1}^{\ell} y_i \alpha_i x_i = 0, \\ \frac{\partial L(\psi,b,\alpha)}{\partial b} = \psi - \sum_{i=1}^{\ell} y_i \alpha_i = 0.\end{array} \tag{4.41}$$

From these conditions it follows that for the vector ψ that defines the optimal hyperplane, the equalities

$$\psi = \sum_{i=1}^{\ell} y_i \alpha_i x_i, \tag{4.42}$$

$$\sum_{i=1}^{\ell} y_i \alpha_i = 0 \tag{4.43}$$

holds true. By substituting (4.42) into (4.40) and taking into account (4.43) can be obtained

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i * x_j). \tag{4.44}$$

It should be noted that the notation has been changed from $L(\psi, b, \alpha)$ to $W(\alpha)$ to reflect the last transformation. Now to construct the optimal hyperplane the coefficients a_i^0 have to be found that maximize the function (4.44) in the nonnegative quadrant

$$\alpha_i \geq 0, \qquad i = 1, \ldots, \ell, \tag{4.45}$$

under the constraint (4.43). Using these coefficients $\alpha_i^0, i = 1, \ldots, \ell$, in Equation (4.42) the solution is obtained

$$\psi_0 = \sum_{i=1}^{\ell} y_i \alpha_i^0 x_i. \tag{4.46}$$

The value of b_0 is chosen to maximize margin (4.33). It should be doted that the optimal solution ψ_0 and b_0 must satisfy the Kuhn-Tucker conditions [43]

$$\alpha_i^0 (y_i ((x_i * \psi_0) + b_0) - 1) = 0 \qquad i = 1, \ldots, \ell \tag{4.47}$$

It can be concluded that from conditions (4.47) nonzero values α_i^0 correspond only to the vectors x_i that satisfy the equality

$$y_i ((x_i * \psi_0) + b_0) = 1. \tag{4.48}$$

Geometrically, these vectors are the closest to the optimal hyperplane and are called *support vectors*. The support vectors play a crucial role in constructing a new type of learning algorithm since the vector ψ_0 that defines the optimal hyperplane is expanded with nonzero weights on support vectors:

$$\psi_0 = \sum_{i=1}^{\ell} y_i \alpha_i^0 x_i. \tag{4.49}$$

Therefore the optimal hyperplane has the form

$$f(x, \alpha_0) = \sum_{i=1}^{\ell} y_i \alpha_i^0 (x_s * x) + b_0, \tag{4.50}$$

where $(x_s * x)$ is the inner product of two vectors.

It should be noted that both the separation hyperplane (4.50) and the objective function of the optimization problem

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i * x_j) \tag{4.51}$$

do not depend explicitly on the dimensionality of the vector x but depend only on the inner product of two vectors. This fact will allow later constructing of separating hyperplanes in high-dimensional spaces even in infinite-dimensional Hilbert spaces [31].

The Optimal hyperplane of the nonseparable case

In this section the concept of the optimal hyperplane for the nonseparable case will be generalized.

Let the set of training set

$$(y_1, x_1), \ldots, (y_\ell, x_\ell), \qquad x \in \mathbb{R}^n, \; y \in \{-1, 1\} \tag{4.52}$$

be such that it cannot be separated without error by a hyperplane. According to the definition of nonseparability (see Equation 4.31), this means that there is no pair ψ, b such that

$$(\psi * \psi) \leq \frac{1}{\rho^2} = A^2 \tag{4.53}$$

and the inequalities

$$y_a((x_i * \phi) + b) \geq 1, \quad i = 1, 2, \ldots, \ell \tag{4.54}$$

hold true. The goal is to construct the hyperplane that makes the smallest number of errors. To get a formal setting of this problem the nonnegative variables are introduced

$$\xi_1, \ldots, \xi_\ell \tag{4.55}$$

In terms of these variables the problem of finding the hyperplane that provides the minimal number of training errors has the following formal expression: Minimize the functional

$$\Phi(\xi) = \sum_{i=1}^{\ell} \theta(\xi_i) \tag{4.56}$$

subject to the constraints

$$y_i((x_i * \psi) + b) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, \ell \quad \xi_i \geq 0 \tag{4.57}$$

and the constraint

$$(\psi * \psi) \leq A^2, \tag{4.58}$$

where $\theta(\xi) = 0$ if $\xi = 0$ and $\theta(\xi) = 1$ if $\xi > 0$. It is known that for the nonseparable case this optimization in NP-complete [27]. Therefore the following approximation to this problem is considered: To minimize the functional

$$\Phi(\xi, b) = \sum_{i=1}^{ell} \xi_i \tag{4.59}$$

under the constraints (4.57) and (4.58). The hyperplane

$$(\psi_0 * x) + b = 0 \tag{4.60}$$

constructed on the basis of the solution of this optimization problem is called the *generalized optimal hyperplane* or, for simplicity, the *optimal hyperplane*.

To solve this optimization problem the saddle point of the Lagrangian have to be found

$$L(\psi, b, \alpha, \beta, \gamma) = \sum_{i=1}^{\ell} \xi_i - \frac{1}{2}\gamma(A^2 - (\psi * \psi)) - \sum_{i=1}^{\ell} \alpha_i(y_i((\psi * x_i) + b) - 1 + \xi_i) - \sum_{i=1}^{\ell} \beta_i \xi_i. \tag{4.61}$$

The minimum with respect to ψ, b, ξ_i and the maximum with respect to nonnegative multipliers $\alpha_i, \beta_i, \gamma$ is searched. The parameters that minimize the Lagrangian must satisfy the conditions

$$\begin{aligned} \tfrac{\partial L(\psi,b,\xi,\alpha,\beta,\gamma)}{\partial \psi} &= \gamma\psi - \textstyle\sum_{i=1}^{\ell} y_i \alpha_i x_i = 0, \\ \tfrac{\partial L(\psi,b,\xi,\alpha,\beta,\gamma)}{\partial b} &= \psi - \textstyle\sum_{i=1}^{\ell} y_i \alpha_i = 0, \\ \tfrac{\partial L(\psi,b,\xi,\alpha,\beta,\gamma)}{\partial \xi_i} &= 1 - \alpha_i - \beta_i = 0. \end{aligned} \tag{4.62}$$

From these conditions can be derived

$$\psi = \frac{1}{\gamma} \sum_{i=1}^{\ell} \alpha_i y_i x_i, \tag{4.63}$$

$$\begin{aligned} &\textstyle\sum_{i=1}^{\ell} \alpha_i y_i = 0, \\ &\alpha_i + \beta_i = 1. \end{aligned} \tag{4.64}$$

By substituting (4.63) into the Lagrangian and taking into account (4.64) the functional can be obtained

$$W(\alpha,\gamma)=\sum_{i=1}^{\ell}\alpha_i-\frac{1}{2\gamma}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j(x_i*x_j)-\frac{\gamma A^2}{2}, \tag{4.65}$$

which must be maximized under the constraints

$$\begin{array}{l}\sum_{i=1}^{\ell}y_i\alpha_i=0,\\ 0\le\alpha_i\le 1,\\ \gamma\ge 0\end{array} \tag{4.66}$$

Functional (4.65) can be maximized under the constraints (4.66) by solving a quadratic optimization problem several times for fixed values of γ and conducting maximization with respect to γ by a line search. Another possibility is to find the parameter γ that maximizes (4.65) and substitute it back into (4.65). Is is easy to check that the maximum of (4.65) is achieved when

$$\gamma=\frac{\sqrt{\sum_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j(x_i*x_j)}}{A}. \tag{4.67}$$

Putting this expression back into (4.65) it is obtained that to find the desired hyperplane the following functional has to be maximized

$$W(\alpha)=\sum_{i=1}^{\ell}\alpha_i-A\sqrt{\sum_{i,j}^{\ell}\alpha_i\alpha_j y_i y_j(x_i*x_j)}. \tag{4.68}$$

subject to constraints

$$\begin{array}{l}\sum_{i=1}^{\ell}y_i\alpha_i=0,\\ 0\le\alpha_i\le 1.\end{array} \tag{4.69}$$

The vector of parameters $\alpha^0=(\alpha_1^0,\ldots,\alpha_\ell^0)$ defines the generalized optimal hyperplane

$$f(x)=\frac{A}{\sqrt{\sum_{i,j=1}^{\ell}\alpha_i^0\alpha_j^0 y_i y_j(x_i*x_j)}}\sum_{i=1}^{\ell}\alpha_i^0 u_i(x*x_i)+b. \tag{4.70}$$

The value of the threshold b is chosen to satisfy the Kuhn-Tucker conditions [43].

$$\alpha_t^0\left(\frac{A}{\sqrt{\sum_{i,j=1}^{\ell}\alpha_i^0\alpha_j^0 y_i y_j(x_i*x_j)}}\sum_{i=1}^{\ell}\alpha_i^0 u_i(x*x_i)+b\right)=0,\qquad t=1,\ldots,\ell \tag{4.71}$$

constructing SVM, kernel functions

The support vector machine is the implementation of the following idea: Mapping of the input vectors x into the high-dimensional feature space Z through some chosen nonlinear mapping. And in this space the optimal separating hyperplane is constructed but two problems arise in: (i) How to find a separating hyperplane that generalize well and (ii) How to treat such high-dimensional spaces computationally.

The generalization ability of the constucted hyperplane is high (even if the feature space has a high dimensionality) in the case when it is expected that is created from a small number of support vectors. It should be noted that for constructing the optimal separating hyperplane in the feature space Z, the feature space in explicit form do not need to be considered. Only inner products between support vectors and the vectors of the feature space have to be calculated (see, for example, (4.51) or (4.65)).

Let us consider a general property of the inner product in a Hilbert space. Suppose the mapping of the vector $x \in \mathbb{R}^n$ into a Hilbert space [31] with coordinates

$$z_1(x), \ldots, z_n(x), \ldots \tag{4.72}$$

According to the Hilbert-Schmidt theory the inner product in a Hilbert space has an equivalent representation

$$(z_1 * z_2) = \sum_{r=1}^{\infty} a_r z_r(x_1) z(x_2) \Longleftrightarrow K(x_1, x_2), \qquad a_r \geq 0, \tag{4.73}$$

where $K(x_1, x_2)$ is a symmetric function satysfying the Mercer conditions [2].

The Mercer's theorem with the following definition exists: *To guarantee that a continuous symmetric function $K(u, v)$ in $L_2(C)$ has an expansion*

$$K(u, v) = \sum_{k=1}^{\infty} a_k z_k(x_1) z_k(x_2) \tag{4.74}$$

with positive coefficients $a_k > 0$ (i.e., $K(u, v)$ describes an inner product in some feature space), it is necessary and sufficient that the condition

$$\int_C \int_C K(u, v) g(u) g(v) du dv \geq 0 \tag{4.75}$$

be valid for all $g \in L_2(C)$ (C being a compact subset of $\mathbb{R}^n$).

The remarkable property of the structure of the inner product in Hilbert space leads to construction of the support vector machine with properties that for any kernel function $K(u, v)$ which is satistying the Mercer's condition a feature space $(z_1(u), \ldots, z_k(u), \ldots$ where the function generates the inner product (4.74) exists. This allows the construction of decision functions that are nonlinear in the input space

$$f(x, \alpha) = \text{sign}\left(\sum_{\text{support vectors}} y_i \alpha_i^0 K(x, x_i) + b \right) \tag{4.76}$$

and are quivalent to linear decision function in the feature space $z_1(x), \ldots, z_k(x), \ldots$

$$f(x, \alpha) = \text{sign}\left(\sum_{\text{support vectors}} y_i \alpha_i^0 K(x, x_i) + b \right) \tag{4.77}$$

($K(x, x_i)$ is the kernel that generates the inner product for this feature space). Therefore to construct function (4.76) the methods for constructing linear hyperplanes presented above can be used where instead of the inner product defined as (x, x_i) the kernel defined as $K(x, x_i)$ will be used. The procedure shall be described as follows:

1. To find the coefficients α_i in the separable case

$$y_i f(x_i, \alpha) = 1 \tag{4.78}$$

it is sufficient to find the maximum of the functional

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{4.79}$$

subject to the constraints

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0, \tag{4.80}$$

$$\alpha_i \geq 0, \qquad i = 1, 2, \ldots, \ell. \tag{4.81}$$

2. To find the optimal soft margin solution for the nonseparable case, it is sufficient to maximize (4.79) under constraints

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0, \tag{4.82}$$

$$0 \le \alpha_i \le C. \tag{4.83}$$

3. Finally, to find the optimal solution for a given margin $\rho = 1/A$

$$f(x, \alpha) = \text{sign}\left(\frac{A}{\sqrt{\sum_{i,j=1}^{\ell} \alpha_i^0 \alpha_j^0 y_i y_j K(x_i, x_j)}} \sum_{i=1}^{\ell} \alpha_i^0 y_i K(x_i, x) + b \right) \tag{4.84}$$

the functional

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i = A \sqrt{\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j)} \tag{4.85}$$

has to be maximized subject to constraints

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0, \tag{4.86}$$

$$0 \le \alpha_i \le 1. \tag{4.87}$$

The kernel functions which are widely used for pattern recognition are listed below. It should be noted that the complete list of kernels can be found, for example, in [36].

1. Polynomial kernel,
2. Radial basis kernel,
3. Two-layer neural network kernel,
4. χ^2 distance kernel.

The polynomial support vector machines are based on a d-degree polynomial decision rules denoted as $K(x, x_i) = [(x * x_i) + 1]^d$. This symmetric function, rewritten in coordinates space, describes the inner product in the feature space that contains all the products $x_{i_1}, \ldots, x_{i_d}$ up to the degree d.

Classical radial basis function (RBF) [14] machines use the following set of decision rules

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i K_\gamma(|x - x_i|) - b \right) \tag{4.88}$$

where $K_\gamma(|x - x_i|)$ depends on the distance $|x - x_i|$ between two vectors. The function $K_\gamma(z)$ is a positive definite monotonic function function for any fixed γ; it tends to zero as $|z|$ goes to infinity. The most popular function of this form is $K_\gamma(|x - x_i|) = e^{-\gamma |x - x_i|^2}$.

Two-layer Neural Support Vector Machine is defined when the kernels $K(x, x_i) = S[(x * x_i)]$, where $S(u)$ is a sigmoid function, are chosen. In contrast to kernels for polynomial machines or for radial basis function machines, the sigmoid kernel

$$S[x * x_i] = \frac{1}{1 + e^{\nu(x * x_i) - c}} \tag{4.89}$$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = -1, +1$
Initialize $D_a(i) = 1/m$.
For $t = 1, \ldots, T$:

1. Train weak learner using distribution D_t.
2. Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error $\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$.
3. Choose $\alpha_t = \frac{1}{2}\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
4. Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{d_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{d_t(i)\ \exp(-\alpha_t y_i h_t(x_i)}{Z_t} \end{aligned} \tag{4.90}$$

where Z_t is a normalization factor (chosen so that d_{t+1} will be a distribution)

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{4.91}$$

Figure 4.4: Schema of the AdaBoost boosting algorithm

satisfies the Mercer condition only for some values of parameters c and ν. For example if $|x| = 1, |x| = 1$ the parameters c and ν of the sigmoid function has to satisfy the unequality $c \leq \nu$.

χ^2 kernel has the following form

$$K(x, y) = \frac{1}{2}\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(x_i + y_i)}$$

and is based on χ^2 distribution [59].

AdaBoost

The AdaBoost algorithm were introduced in [26]. Pseudocode for that algorithm is given in Figure 4.4. The algorithm takes as an input a training set $(x_1, y_1), \ldots, (x_m, y_m)$ where each x_i belongs to some instance space X, and each label y_i is in some label set Y. For simplifying purposes the $Y = -1, +1$ can be assumed. AdaBoost "calls" a given weak learning algorithm repeatedly in a series of rounds $t = 1, \ldots, T$. One of the main ideas of the algorithm is to maintain a distribution on training example i on round t is denoted $D_t(i)$. Initially, all weights are set to the same value, but on each round, the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set.

The weak learner's job is to find a weak hypothesis $h_t : X \to \{-1, +1\}$ appropriate for the distribution D_t. The quality of a weak hypothesis is measured by its error

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i : h_t(x_i) \neq y_i} D_t(i) \tag{4.92}$$

It should be noted that the error is measured with respect to the distribution D_t on which the weak learner was trained. Practically, the weak learner may be an algorithm that can use the weights D_t on the training samples. Alternatively, when this is not possible, a subset of the training examples can be sampled according to D_t, and these resampled examples can be used to train the weak learner.

Once the weak hypothesis h_t has been received, AdaBoost chooses a parameter α_t as in Figure 4.4. Intuitively, α_t measures the importance that is assigned to to h_t. It should be noted that $\alpha_t \geq 0$ id $\epsilon_t \leq 1/2$, and that α_t gets larger as ϵ_t gets smaller.

The distribution D_t is subsequently updated using the rule shown in Figure 4.4. The effect of this rule is to increase the weight of examples misclassified by h_t, and to decrease the weight of correctly classified examples. Thus, the weight tends to concentrate on "hard" examples.

The final hypothesis H is a weighted majority vote of the T weak hypotheses where α_t is the weight assigned to h_t. It has been shown that AdaBoost and its analysis can be extended to handle weak hypotheses whose return real-valued or confidence-rated predictions. That is, for each instance x, the weak hypothesis h_t returns a prediction $h_t(x) \in \mathbb{R}$ whose sign is the predicted label (-1 or +1) and whose magnitude gives a measure of "confidence" in the prediction.

The most basic theoretical property of AdaBoost concerns its ability to reduce the training error. Let us write the error ϵ_t of h_t as $\frac{1}{2} - \gamma_t$. Since a hypothesis that guesses each instance's class at random has an error of 1/2 (on binary problems), γ_t thus measures how much better than random are h_t's predictions. It has been proved [26] that the training error (the fraction of mistakes on the training set) of the final hypothesis H is at most

$$\prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2} \tag{4.93}$$

Thus, if each weak hypothesis is slightly better that random so that $\gamma_t \geq \gamma$ for some $\gamma > 0$, then the training error falls exponentionally.

A similar property is enjoyed by previous boosting algorithms. However, previous algorithms required that such a lower bound γ be known a priori before boosting begins. In practice, knowledge of such a bound is very difficult to obtain. AdaBoost, on the other hand, is adaptive in that is adapts on the error rates of the individual weak hypotheses.

The bound given in Equation (4.93), combined with the bounds on generalization error given below, prove that AdaBoost is indeed a boosting algorithm in the sense that it can efficiently convert a weak learning algorithm into a strong learning algorithm.

It has been shown [26] how to bound the generalization error of the final hypothesis in terms of its training error, the sample size m, the VC-dimension[1] d of the weak hypothesis space and the number of boosting rounds T. Specific technique has been used to show that the generalization error is, with high probability, at most

$$\hat{Pr}[H(x) \neq y] + \hat{O}\left(\sqrt{\frac{Td}{m}}\right) \tag{4.94}$$

where $\hat{Pr}[\cdot]$ denotes empirical probability on the training sample. This bound suggests that boosting will overfit if run for too many rounds, i.e. as T becomes large. In fact, this sometimes does happen. However, it has been observed empirically that boosting often does not overfit, even when run for thousand of rounds. Moreover, it was observed that AdaBoost would sometimes continue to drive down the generalization error long after the training error had reached zero, clearly contradicting the spirit of the bound above.

In response to these empirical findings, it has been presented an alternative analysis [5] in terms of the margins of the training examples. The margin of example (x, y) is defined

$$\frac{y\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t}. \tag{4.95}$$

[1]The VC-dimension is a standard measure of the "complexity" of a space of hypotheses, for more information see [12].

It is a number in $[-1, +1]$ which is positive if and only if H correctly classifies the example. Moreover, the magnitude of the margin can be interpreted as a measure of confidence in the prediction. Larger margins on the training set translate into a superior upper bound on the generalization error. The generalization error is at most

$$\hat{P}r[\text{margin}(x, y) \leq \theta] + \hat{O}\left(\sqrt{\frac{Td}{m}}\right) \tag{4.96}$$

for any $\theta > 0$ with high probability. It should be noted that this bound is entirely independent on T, the number of rounds of boosting. It has been proved that boosting is particularly aggressive at reducing the margin since it concentrates on the examples with the smallest margins.

Finally, It should be noted that a number of AdaBoost algorithm variants exists, for instance Real Adaboost [66], WaldBoost [75] etc., exists.

4.4 Extended pipeline schema

The extended pipeline is shown in Figure 4.5 and basically it is the extension of the procedure which has been described in Section 4.1. The extension consists of an usage of multiple feature extractors [82, 88] as well as visual vocabularies, several classifiers and finally it should lead to the creation of better output classifier. This approach has generally a potential to improve the quality of a video processing solution because every solved problem has its own particularity and this processing can select well performing feature extractors for such purpose.

All those facts cause an usage of more of bag-of-words [74] units. The outputs from bag-of-words units ale called a channels and are "concatenated" into one long multiple channels feature vector. Every element in this feature vector consists of one bag-of-words full representation which is not divided into smaller parts.

From the multiple channels feature vector a number of selections are constructed and from each selection a classifier is created. The term selection is considered as a "reduction" of number of channels, one extreme possibility is when all channels are selected, on the other side is when only one channel is selected. More classifiers are produced in this way and the best one needs to be selected.

Selected best classifier uses only finite set of input channels which were selected in corresponding selection unit. Classifier produced by this pipeline should be used for prediction of unwanted entites, only corresponding feature extractors need to be provided for the computation (this can reduce the computation effort).

The selection of best classifier is demanding problem, it can be solved by using technique called leave-one-out [21] where all the classifiers are trained using the training data, where one sample has been removed and thereafter the test is performed using the removed sample. This procedure is repeated several times and everytime a different sample is removed from the sample set (this can be repeated up to size of the training data minus one). The best classifier can be evaluated by computing multiple statistic metrics. In this work another technique is used. It is the usage of validation dataset which also leads to selection of the best performing classifier.

Theoretical usage of the pipeline can be formally descripted as follows; The initial conditions are

(i) A set of algorithms $M_1, \ldots, M_m$ for feature vector production exist.

(ii) A dataset $D = \{D_{TR}, L_{TR}, D_{VA}, L_{VA}, D_{TE}, L_{TE}\}$ with wanted examples and as well as with counter examples exists. All parts are defined in the same way as in Equations (4.2) and (4.3):

$$\begin{gathered} D_{TR} = \{f_{RGB_1}, \ldots, f_{RGB_r}\},\ D_{TE} = \{f_{RGB_1}, \ldots, f_{RGB_s}\},\ D_{VA} = \{f_{RGB_1}, \ldots, f_{RGB_t}\} \\ L_{TR} = \{l_1, \ldots, l_r\},\ L_{TE} = \{l_1, \ldots, l_s\},\ L_{VA} = \{l_1, \ldots, l_t\} \end{gathered} \tag{4.97}$$

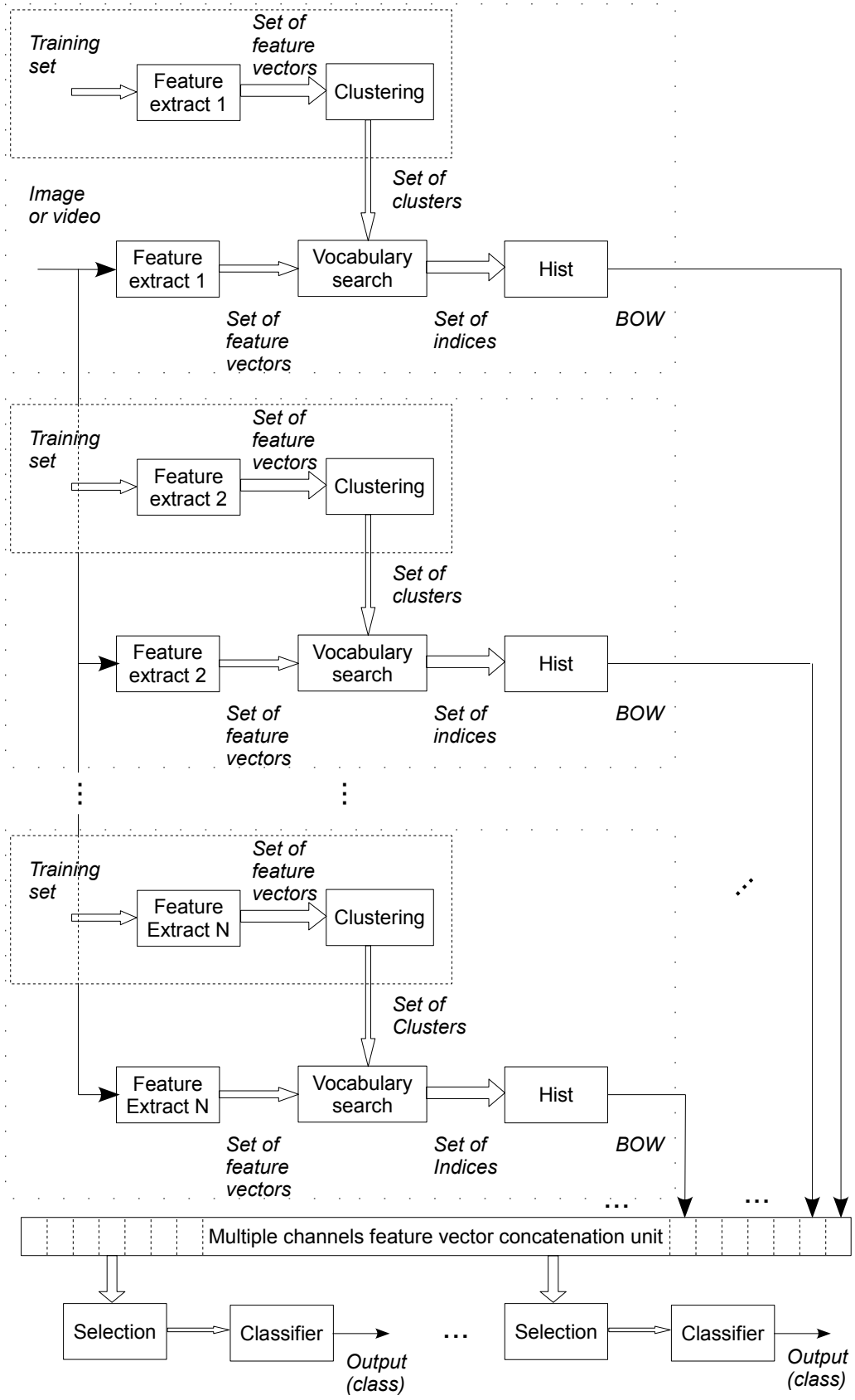


Figure 4.5: Extended pipeline schema

The recognition system can be built according to the following procedure

(1) All training samples are used to create a number of vocabularies for each extraction algorithm.

$$\{M_i(D_{TR})\} \longrightarrow C_{i,c}\,, \quad \text{for each } M_i \in M \text{ and for each } c \in \{1, 2, \dots\} \tag{4.98}$$

the transformation function is defined in Equation (4.4)

(2) The bag-of words representation is created for all extraction algorithms and for corresponding vocabularies.

$$\{M_i(D_{TR}), C_{i,c}\} \longrightarrow B_{TR(i,c)}\,, \quad \text{for each } M_i \in M \text{ and for each } c \in \{1, 2, \dots\} \tag{4.99}$$

the transformation function is defined in Equation (4.5)

(3) All types of bag-of-words representations (as produced in the step above) are concatenated into one long multiple channel feature vector

$$\begin{gathered}\{B_{TR(1,1)}, B_{TR(1,2)}, \dots, B_{TR(1,c)}, B_{TR(2,1)}, \dots\} \longrightarrow K : K = (K_1, K_2, \dots), \\ \text{for each } i \in \{1, \dots |M|\} \text{ and for each } c \in \{1, 2, \dots\}\end{gathered} \tag{4.100}$$

where $K_1 = B_{TR(1,1)}$, $K_2 = B_{TR(1,2)}, \dots$.

(4) Create a number of selections: $S = \{S_1, S_2, \dots, S_s\}$ for each $S_i \in S : S_i \subseteq K \wedge$ for each $i \in \{1, \dots, s\}$ and $j \in \{1, \dots, s\}$ applies $S_i \neq S_j$

(5) Create classifiers based on selections:

$$\begin{gathered}\{S_i, L_{TR}\} \longrightarrow Z_i : \text{ for each } S_i \in S;\ Z_i = (z_{i(1)}, \dots, z_{i(a)}) | z_{i(a)} \in \langle 0, 255\rangle \wedge z_{i(a)} \in \mathbb{N}; \\ Z_i = \mathbf{trainM}(S_i, \{L_1, \dots, L_r\}, P_t)\end{gathered} \tag{4.101}$$

where the **trainM** function performs the training of the classifier and accepts a set of input representations (specified as S_i) and is able to produce a classifier Z_i.

(6) Step (2) is repeated for validation part of the dataset

$$\{M_i(D_{VA}), C_{i,c}\} \longrightarrow B_{VA(i,c)}\,, \quad \text{for each } M_i \in M \text{ and for each } c \in \{1, 2, \dots\} \tag{4.102}$$

(7) The multiple channels feature vector K' of validation dataset is created from validation dataset identically as in step (3) and the selections S' are produced as in step (4).

(8) Perform test on each sample of validation dataset

$$\begin{gathered}\{S'_i, Z_i\} \longrightarrow R_i : \text{ for each doublet } S'_i, Z_i | i \in \{1, \dots, |S'|\}; \\ R_i = \{R_{i(1)}, \dots, R_{i(t)}\} | R_{i(y)} \in \mathbb{R} \text{ for each } y \in \{1, \dots, t\}; \\ R_i = \mathbf{testM}(S'_i, Z_i, P_t)\end{gathered} \tag{4.103}$$

where the **testM** function performs the testing of the classifier Z_i and accepts a set of input representations (specified as S'_i) and is able to produce responses, t is the size of validation dataset.

(9) Evaluate a quality of classifier of each particular combination

$$R_i \longrightarrow Q_i : \text{ for each } i \in \{1, \dots, |S'|\};\ Q_i \in \langle 0, 1\rangle \wedge Q \in \mathbb{R} \text{ for each } i \in \{1, \dots, |S'|\} \tag{4.104}$$

and select the best classifier

$$\{\{Z_1, Z_2, \dots\}, \{Q_1, Q_2, \dots\}\} \longrightarrow Z_b : \text{ where } b = \mathbf{max}(\{Q_1, Q_2, \dots\}) \tag{4.105}$$

function **max** returns the index of the best performing classifier. The best performing selection S_b is also identified.

(10) Step (2) is repeated for testing part of the dataset

$$\{M_i(D_{TE}), C_{i,c}\} \longrightarrow B_{TE(i,c)}\,, \quad \text{for each } M_i \in M \text{ and for each } c \in \{1, 2, \dots\} \tag{4.106}$$

It should be noted that according to selection S_b not necessarily all channels need to be evaluated.

(11) The testing part of the dataset selection S_b'' is created identically as in step (3) and in step (4).

(12) Perform testing of the classifier Z_b using the testing part of the dataset.

$$\begin{gathered}\{S_b'', Z_b\} \longrightarrow R: \; R = \{R_1, \dots, R_s\} | R_y \in \mathbb{R} \text{ for each } y \in \{1, \dots, s\}; \\ R = \mathbf{testM}(S_b'', Z_b, P_t)\end{gathered} \tag{4.107}$$

and evaluate the quality of the solution

$$R \longrightarrow Q: \; R = \{R_1, \dots, R_t\}; \; R_i \in \mathbb{R} \text{ for each } i \in \{1, \dots, t\}; \; Q \in \langle 0, 1\rangle \wedge Q \in \mathbb{R} \tag{4.108}$$

Algorithm 2: Extended pipeline algorithm

Standard classification methods, as described in Section 4.3, can be used as functions **trainM** and **testM** in the algorithm above but the input set of vectors needs to be transformed to one input vector. Simple concatenation of all input vectors can be used for such purpose. More valuable method is to use multi-kernel support vector machines method [16, 90](especially in the field of action recognition as shown in [82, 48, 62]).

The bag-of-words feature vectors are combined using multi-kernel support vector machine with a multichannel gaussian kernel [40, 16]. The kernel shall be defined as:

$$K(A, B) = exp(-\sum_{c \in C} \frac{1}{A_c} D_c(A, B)) \tag{4.109}$$

where A_c is the scaling parameter which is determined as a mean value of mutual distances D_c between all the training samples (Equation 4.112) for the channel c from a set of channels C, $D_c(A, B)$ is the χ^2 distance between two bag-of-words vectors, A and B are the input vectors of the form:

$$A_i = (\underbrace{a_1 \dots a_{n1}}_{channel\ \langle 1, n_1\rangle}, \underbrace{a_{n_1+1} \dots a_{n2}}_{channel\ \langle n_1+1, n_2\rangle}, \dots, \underbrace{a_{n_i-1} \dots a_{n_i}}_{channel\ \langle n_i-1, n_i\rangle}) \tag{4.110}$$

where set of channels C can be defined as:

$$C = \{\langle 1, n_1\rangle, \langle n_1 + 1, n_2\rangle, \dots, \langle n_i - 1, n_i\rangle\} \tag{4.111}$$

The bag-of-words distance $D_c(A, B)$ is defined as:

$$D_c(A, B) = \frac{1}{2} \sum_{n \in c} \frac{(a_n - b_n)^2}{a_n + b_n} \tag{4.112}$$

The best ratio of combined channels $\{c_k, c_l, \dots, c_z\} \in C$ for a given training set is estimated using a coordinate descend method. The set of input channels needs to be specified outside of the training process. Althought this SVM building procedure requires the number of input parameters that affect the classifier accuracy; these parameters are automatically evaluated using the cross-validation approach [32]. The classifier creation process may be seen as a black-box unit which for a given input automatically creates the best performing classifier.

Chapter 5

Optimal analysis length

Many scientific papers concerning human action recognition focus on the improving some part or some parts of the video processing pipeline. Currently, the improvements are mainly in the development of new local space-time features [47, 88, 20, 82] and in combining known space-time features [82, 38]. Other solutions where the processing pipeline is different also exist such as deep-learning techniques [49]. All proposed procedures have something in common; the use of a dataset for evaluating quality.

Datasets usually contain a number of videos with variable length, the proposed actions are usually located anywhere within a particular example. When using processing similar to the one presented in Section 4.1, the bag-of-words unit processes the whole video sequence which is potentially large and will have varying size. Such a procedure is called off-line processing.

Some scenarios require the usage of a processing step, in which the number of frames processed by the bag-of-words unit is fixed to a predefined value. In other words, the bag-of-words representation is extracted from a sub-shot of the video with constant length. This is called on-line processing.

The following sections are organized as follows, in Section 5.1 the main contribution of the thesis is presented. Information about datasets available today and used in current research is given in Section 5.2. The main experiments done in this thesis require a state-of-the-art method, which is presented and evaluated in Section 5.3. The main experiments about on-line processing are presented in Section 5.4. Finally, Section 5.5 presents the summary and potential applications.

5.1 Determining the optimal length of the analyzed video

The contribution of this thesis is in: (i) to state the hypothesis that an *optimal* analysis length exists for on-line action recognition solutions and (ii) the proof of this hypothesis through an algorithmic solution.

The *optimal* length of analyzed video ℓ_o exists for each on-line human action recognition system where the solution has a quality q_o which is comparable to the off-line solution quality q, formally $q_o \geq q - \epsilon$ and this assumption applies for each arbitrarily small ϵ. For appropriate values of ϵ, the *optimal* analysis length of videos of certain actions can be much smaller than the potentially unrestricted length, which is processed by the off-line solution.

Subsequently, according to the hypothesis, for certain actions an analysis action length may exist, such that the on-line solution gives better quality than the off-line solution.

Such an algorithmic proof of the statements has two inputs:

(i) A dataset with start and end positions of each annotated action (DS),

(ii) A state-of-the-art method in which quality is expressed by a function $\mathbf{M}$: DS $\longrightarrow \mathbb{R}$.

Formally, the algorithm can be interpreted as a mapping:

$$(\mathbf{M}, DS, q, \epsilon) \longrightarrow ((\ell_o, q_o), (\ell_b, q_b)) \tag{5.1}$$

L represents all lengths to be analyzed (in number of frames)

$$L = \langle \ell_{min}, \ell_{max} \rangle \cap \mathbb{N} \tag{5.2}$$

Metrics of qualities for each length of sequence are represented by a vector D.

$$D = \{(\ell_i, q_i) \,|\, \ell_i \in L, q_i = \mathbf{M}(\mathbf{CO}(DS, \ell_i))\} \tag{5.3}$$

The *optimal* analysis length of the video is ℓ_o, and the analysis length where the maximum quality is achieved is ℓ_b. Solution quality q_o and q_b are defined as well.

$$(\ell_o, q_o) \in D \,|\forall (\ell_j, q_j) \in D, \ell_j < \ell_o, q_j < q - \epsilon) \tag{5.4}$$

$$(\ell_b, q_b) \in D \,|\forall (\ell_j, q_j) \in D, q_j \leq q_b \tag{5.5}$$

The function $\mathbf{CO}(DS, \ell_i)$ returns a dataset (which is based on DS) where each video sample has constant length ℓ_i.

It should be noted that it is not guaranteed that (l_o, q_o) exist but it does for most cases.

A practical way to obtain (l_o, q_o) and (l_b, q_b) is described in the algorithm below:

(1) Get ℓ_{min} and ℓ_{max}

(2) **for** i = ℓ_{min}:ℓ_{max} {
    D[i]= $\mathbf{M}(\mathbf{CO}(\mathrm{DS}, \mathrm{i}))$
    }

(3) perform:
    o = NULL; **for** i = ℓ_{min}:ℓ_{max} {
    if(D[o] $\geq$ (q - ϵ)) { o = i; **break**; }
    }
    and return ℓ_o = o and q_o = D[o].

(4) perform:
    b = ℓ_{min}; **for** i = ℓ_{min}:ℓ_{max} {
    if(D[i] > D[b]) { b = i; }
    }
    and return ℓ_b = b and q_b = D[b].

Algorithm 3: Verification algorithm pseudocode.

It should be noted that the algorithm is not optimal. The main reason why it has been constructed, is to prove the presented hypothesis, and to obtain the *optimal* analysis length. The creation of a more powerful algorithm is not the focus of this thesis.

The experimental proof of the hypothesis using this algorithm is shown below.

5.2 Action recognition datasets

A dataset needs to be selected before performing some experiments. The list of datasets being currently used has to be found out, explored and finally the suitable one must be selected. In case that no dataset for wanted purpose is already created, the experiments must be based on newly created one.

This section presents available datasets and includes also the selection of the most suitable one which will be used in experiments.

- **KTH** [69]
  This dataset was introduced by Chritian Schulds et al. in 2004. It consists of 6 action classes, namely: *walking, jogging, running, boxing, hand waving* and *hand clapping*. All videos were acquired by a static camera in a controlled environment. The whole dataset consists of 600 videos which are split into three parts: training part (192 videos), validation part (192 videos) and testing part (216 videos). The solution which reached the 100% accuracy has been already provided by research comunity.

  Example screenshots[1] from dataset are shown below

- **Weizmann** [10]
  The dataset was introduced by Moshe Blank et al. in 2005. It consists of 10 action classes, namely: *walk, run, jump, gallop sideways, bend, one hand wave, two hands wave, jump in place, jumping jack* and *skip*. All videos were acquired by a static camera in a homogeneous outdoor environment. The whole dataset consists of 90 videos with small resolution.

  Various examples[2] are shown in the image below; each line consists of 5 frames of the same action with a temporal distance, these actions are gradually shown: *two hands wave, run, walk*.

- **IXMAS** [86, 85]
  The dataset was introduced by H. Kuehne et al. in 2006. It consists of 11 action classes which are performed 3 times by 10 actors. To demonstrate the view invariance, the actors freely change their orientation for each aquisition and no further indications on how to perform the actions beside the labels were given. The videos are captured in a controlled environment, the action classes are namely: *check watch, cross arms, schratch head, sit down, get up, turn around, walk, wave, punch kic* and *pick up*.

[1] Pictures are taken from [69]
[2] Pictures are taken from [10]

- **UCF sports** [64]
  The dataset was introduced by Mikel D. Rodriguez et al. in 2008. It consists of 9 action classes collected from broadcast television channels such as the BBC and ESPN. Actions in this dataset include *diving, golf swinging, kicking, lifting, horseback riding, running, skating, swinging a baseball bat* and *pole vaulting*. The dataset contains over 200 video sequences at a resolution of 720 $\times$ 480. The collection represents a natural pool of actions featured in a wide range of scenes and view points.

- **Youtube** [53]
  The dataset was introduced by Jingen Liu et al. in 2008. It consists of 11 action classes, namely: *basketbal shooting, volleybll spiking, trampoline jumping, soccer juggling, horseback riding, cycling, diving, swinging, golf swinging, tennis swinging* and *walking (with a dog)*. The dataset contains the 1168 video sequnce in total. The videos were not created within a human controlled environment.

- **HMDB** [42]
  The dataset was introduced by H. Kuehne et al. in 2011. It consists of 51 action classes with, at least, 101 examples for each action class. The total number of action clips is 6766 and are extracted from wide range of sources. Each clip was validated by at least two human observers to ensure consistency. Additional meta information allows precise selection of testing data, as well as training and evaluation of recognition systems.

- **Hollywood2** [48]
  The Hollywood2 dataset was introuced in 2009 by Ivan Laptev et al. with 12 action classes and 10 scene classes annotated, which is acquired from 69 Hollywood movies. The dataset[3] is built from movies containing human actions and processed using script documents and subtitle files which are publicly available. The script documents contain the scene captions, dialogs and the scene descriptions; however, they are usually not quite precisely synchronized with the video. The subtitles have video synchronization so they are matched to the movie scripts and this fact can be used for improvements in video clip segmentation. By analyzing the content of movie scripts, the twelve most frequent action classes and their video clip segments are obtained. These segments are split into test and training subsets so that the two subsets do not share segments from the same movies.

  These 12 action classes are included in the dataset: *answering the phone, driving car, eating, fighting, getting out of the car, hand shaking, hugging, kissing, running, sitting down, sitting up and standing up*. The framerate of the videos is 25 fps. Complete set of examples of the frames contained in these video sequences are shown in Appendix A. Subsequently, examples[4] of the classes: *answering the phone, standing up* are shown.

[3]The Hollywood2 dataset can be downloaded from: http://www.di.ens.fr/~laptev/actions/hollywood2/
[4]Pictures are taken from [48]

Two training parts of the dataset exist: the automatic part generated using the above mentioned procedure, and the clean part which is manually corrected using visual information from the video. The test part is manually corrected in the same way as in the clean training part of the dataset. In both cases, the correction is performed in order to eliminate "noise" from the dataset and consequently to create better classifiers. In the work described above, some experiments were performed. The processing chain consisted of feature extraction the SIFT [55] image and STIP [47] space-time extractors, both converted into a bag-of-words representation and then used in multichannel χ^2 Support Vector Machine for classification purposes. The results are measured using a mean average precision metric across all of the classes and presented as the first evaluation performed on this dataset.

The Hollywood2 dataset, described above, was selected for testing of the state-of-the-art method as well as in the optimal analysis length obtaining experiments. The Hollywood2 dataset has been chosen, because of two facts: (i) it contains the real-world actions performed by humans which are generally valuable to be detected and (ii) the dataset is "hard" enough to be "solved", the today's best algorithms reaches approximately the 0.6 of mean average precision [83, 62, 63, 82, 38, 3].

5.3 Off-line action recognition experiments

The main achievement of the experiments in this section is confirmation that the suitable combination of different features for action recognition (as described in Section 4.4) produces the state-of-the-art results. This has been evaluated using one of the most challenging datasets Hollywood2[48] available today. The following twelve action classes were evaluated, namely: *answering the phone, driving car, eating, fighting, getting out of the car, hand shaking, hugging, kissing, running, sitting down, sitting up and standing up.*

In our experiments, the clean part of the training dataset was used for classifiers' training procedure (823 samples). The automatic part of the training dataset was re-anotated and used for validation purposes (810 samples). The original testing dataset (884 samples) was used for measuring the solution using average precision for every class, over-all classes the mean average precision is reported.

The following feature extractors were used in the experiment, in parentheses the associated list of descriptors is stated, every combination extractor vs. descriptor was used as a standalone features set plus all the dense trajectories descriptors were concatenated together used as well:

- Dense Trajectories (Trajectory, HOG, HOF, MBH),
- HesSTIP (ESURF)
- Cuboids (Cuboids)
- STIP (HOGHOF)

Some vocabularies were created using k-means algorithm with 12 iterations; this number presents compromise between the processing duration and the output vocabulary achievement. For creating those vocabularies the cca 2 millions of local low-evel features were used and were extracted from all training videos of the dataset. Vocabulary sizes were set to 1000, 6000 and 8000, all possible combinations, feature extractors vs. vocabularies sizes were used.

The soft-assignment approach, based on [61] as presented earlier in Equation (4.22), was used for bag-of-words representation with the following parameters: $\sigma = 1$, the number of searched closest vectors is 16; these values were evaluated in [22] and are suitable for bag-of-words creation from space-time low-level features.

Bag-of-words representations generated from all the possible combinations feature extractors vs. vocabularies become the input channels to the SVM creation process. SVMs were created as described in chapter 4.4. The dataset used induces the multiclass classification. The one-against-rest approach was used and no relation between classes has been considered.

Table 5.1: Results of average precision of the four best performing experiments on the validation dataset.

Action	1	2	3	4	**BEST**	Selected Classifier
answering the phone	0.379	0.299	0.322	0.423	0.423	4
driving car	0.571	0.62	0.554	0.578	0.62	2
eating	0.327	0.355	0.295	0.37	0.37	4
getting out of the car	0.377	0.237	0.304	0.273	0.377	1
running	0.629	0.683	0.736	0.702	0.736	3
sitting down	0.487	0.559	0.511	0.574	0.574	4
sitting up	0.286	0.204	0.385	0.331	0.385	3
standing up	0.486	0.55	0.394	0.527	0.55	2
fighting	0.625	0.594	0.55	0.561	0.625	1
hand shaking	0.493	0.541	0.439	0.594	0.594	4
hugging	0.355	0.339	0.417	0.369	0.417	3
kissing	0.531	0.630	0.594	0.609	0.630	2
mean average precision	0.462	0.468	0.458	0.478	0.484	

The number of input channels in our experiment is 24 and the total number of possibilities is then:

$$\binom{24}{1} + \binom{24}{2} + \cdots + \binom{24}{24} \simeq 16.7 \,.\, 10^6. \tag{5.6}$$

We have searched about 0,1% of the desired space in a semi-automatic way and the four most interesting results (combinations) for validation part of the dataset are presented in Table 5.1. The average precision is reported for each class and the mean average precision is reported for the whole validation dataset.

Table 5.2 represents the results *for our class-based best input channel combinations (as shown in Table 5.1)* achieved using the test part of the Hollywood2 dataset in column OUR and it is compared to the three other authors' papers [82] and [49] and [80] which represent the today's state-of-the-art for Hollywood2 dataset.

Our combination-based solution outperformed all other state-of-the-art methods in four classes, namely *driving car, running, sitting down, standing up*; in other casses, the solution does not reach the state-of-the-art performance but it is still comparable.

As the performace of classifiers based on the combination of features is known after the validation phase, best solution based on the combination of the features or other approach can be chosen individually for each type of action; therefore, improvement in four out of twelve actions leads into the best known classification mechanism shown in Table 5.2 as well.

5.4 Optimal length experiments

The purpose of the experiments performed in this section is the verification of the hypothesis about the *optimal* analysis length proposed in Section 5.1.

We have used the pipeline presented in Chapter 4.4 and the Hollywood2 [48] dataset presented in Chapter 5.2. This dataset, as mentioned earlier, contains twelve action classes from Hollywood movies, namely: *answering the phone, driving car, eating, fighting, getting out of the car, hand shaking, hugging, kissing, running, sitting down, sitting up and standing up.*

We investigated the recognition algorithm behavior in such a way that pieces of video containing the action were presented to the algorithm at randomly selected positions inside the actions. For

Table 5.2: Results of average precision of the selected classifiers, compared to the state-of-the-art.

Action	**OUR**	Wang [82]	Q. V. Le [49]	Ullah [80]	**BEST KNOWN**
answering the phone	0.259	**0.326**	0.299	0.248	**0.326**
driving car	**0.91**	0.880	0.852	0.881	**0.91**
eating	0.491	**0.652**	0.597	0.614	**0.491**
getting out of the car	0.408	**0.527**	0.454	0.474	**0.527**
running	**0.834**	0.821	0.757	0.743	**0.834**
sitting down	**0.655**	0.625	0.594	0.613	**0.655**
sitting up	0.206	0.200	**0.257**	0.255	**0.257**
standing up	**0.663**	0.652	0.647	0.604	**0.663**
fighting	0.723	**0.814**	0.772	0.765	**0.814**
hand shaking	0.286	0.296	0.203	**0.384**	**0.384**
hugging	0.364	**0.542**	0.382	0.446	**0.542**
kissing	0.601	**0.658**	0.579	0.615	**0.658**
mean average precision	0.533	**0.583**	0.533	0.553	**0.589**

example, the actions were known to start earlier than the beginning of the processed piece of video, and ended only after the end of the presented piece of video. For this purpose, we had to reannotate the Hollywood2 dataset (all three its parts - train, autotrain and test) to obtain precise beginning and ending frames of the actions.

In our experiments, we have been trying to depict a dependency between the length of video shot, being an input to the processing, and the accuracy of the output. We have set the minimum shot length to 5 frames, more precisely the 5 frames from which the space-time point features are extracted. The maximum shot length was set to 100 frames and the frame step was set to 5 frames.

The space-time features extractor process N previous and N consequent frames of the video sequence in order to evaluate the points of interest for a single frame. Therefore, 2*N should be added to every figure concerning the number of frames to get the total number of frames of the video sequence to be processed. In our case, N was equal to 4 so that, for example, the 5 frames processed in Figure 5.1 mean 13 frames of the video.

A classifier has been constructed for every video shot length considered. The training samples were obtained from the training part of the dataset in the following way: the information of the start and stop position in the currently processed sample was used and large number of the randomly selected subshots were obtained. The training dataset has 823 video samples in total and from each sample, we extracted 6 subshots on average.

The actual evaluation of the classifier has been done four times in order to obtain the information about reliability of the solution. Also, the above mentioned publications used the 823 samples for evaluation purposes and we wanted our results to be directly comparable. The results shown in Table 5.3 and Figure 5.1 present the average of the results of the four runs. For this purpose we have randomly determined a position of starting frame of a testing subshot within a testing sample four times. The above approach brings us two benefits - the final solution accuracy can be measured using an average precision metric and the results obtained through the testing can be easily comparable to the published state-of-the-art solutions. The results were compared with the accuracy achieved on the video sequences with completely unrestricted size that are close to the state-of-the-art [62].

The parameters for feature processing and classification purposes were as follows: the tested fea-

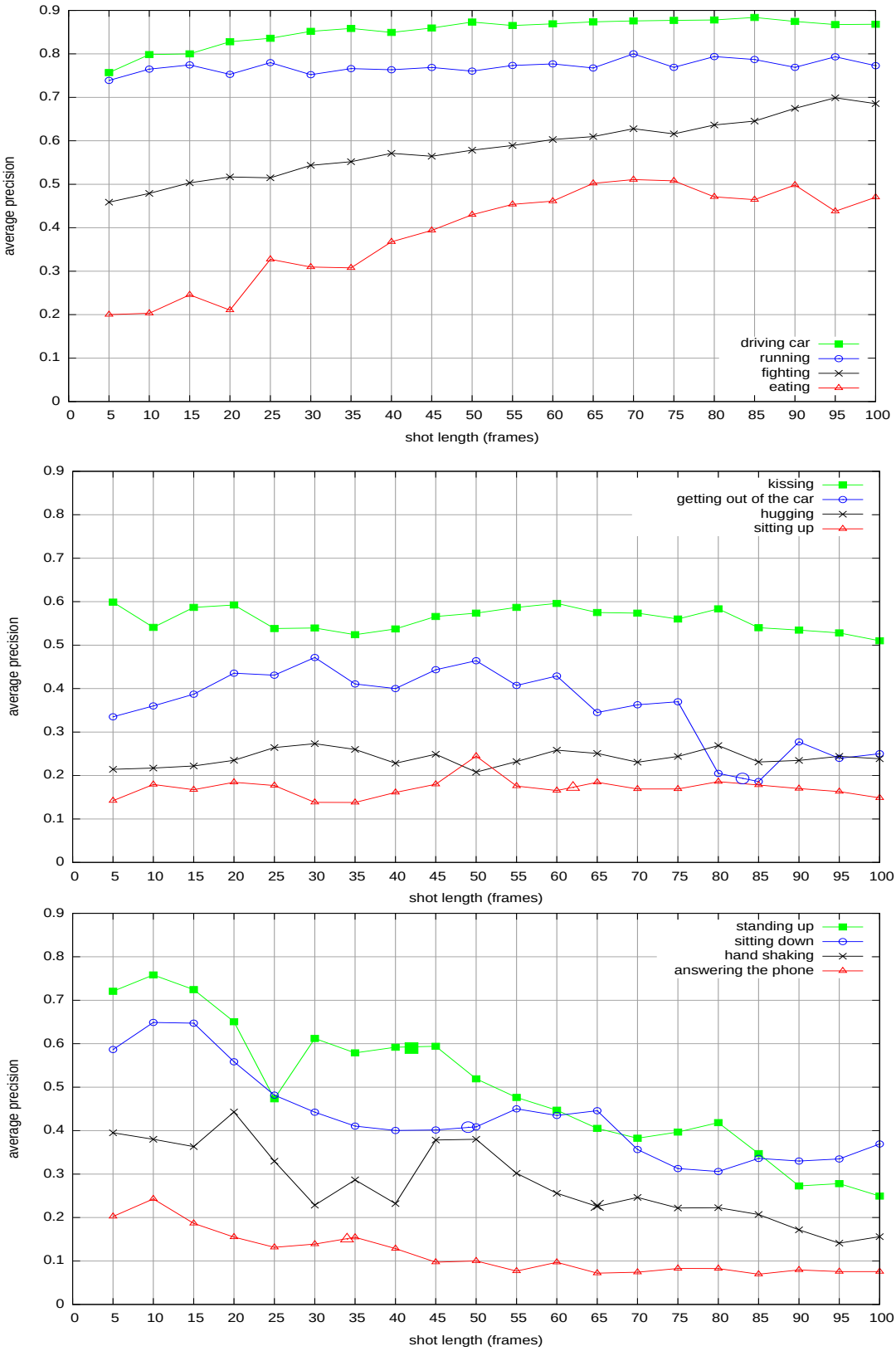


Figure 5.1: Dependency of the average precision on the length of the shot achieved for all classes contained in the Hollywood2 dataset. The dependency is split into 3 separate charts in order to improve readability. It should be noted that the big marks indicating the average action length shown in the charts for actions shorter than 100 frames.

Table 5.3: Results of the experiments. The first four columns show the accuracy (average precision) for the selected video sizes, the consequent column shows the reference accuracy reached for unrestricted video size, and the final column shows the minimum number of frames needed to achieve 90% of the precision achieved using the unrestricted video size. Column description: **A** – Unrestricted video size accuracy, **B** – Number of frames to achieve 90% accuracy

	Video size (frames)				**A**	**B**
Action	5	10	30	90		
driving car	0.757	0.798	0.852	0.874	0.848	10
running	0.739	0.765	0.752	0.769	0.812	10
fighting	0.459	0.479	0.543	0.675	0.718	90
eating	0.2	0.203	0.309	0.498	0.326	25
kissing	0.599	0.541	0.540	0.535	0.597	5
getting out of the car	0.335	0.36	0.471	0.277	0.358	5
hugging	0.214	0.217	0.273	0.235	0.264	25
sitting up	0.142	0.179	0.138	0.17	0.163	10
standing up	*0.721*	*0.758*	*0.612*	*0.273*	*0.598*	*5*
sitting down	*0.587*	*0.649*	*0.442*	*0.33*	*0.654*	*10*
hand shaking	*0.395*	*0.38*	*0.229*	*0.172*	*0.232*	*5*
answering the phone	*0.201*	*0.243*	*0.139*	*0.08*	*0.225*	*10*

ture extractor is the dense trajectories extractor, which produces four types of descriptors, namely: HOG, HOF, DT and MBH. These four feature vectors were used separately. For each descriptor a vocabulary of 4000 words was produced using the k-means method and the bag-of-words representation was produced with the following parameters: $\sigma = 1$, the number of searched closest vectors is 16; these values and codebook size were evaluated in [62] and are suitable for bag-of-words creation from space-time low-level features. In the multi-kernel SVM creation process all four channels (bag-of-words representations of HOG, HOF, DT and MBH descriptors) are combined together, no searching for a better combination is performed.

The above described evaluation procedure was repeated for every class contained in the Hollywood2 dataset. For each class, we are presenting the graph of dependency between the video sample shot length and the system best accuracy, as well as the figure showing the number of frames needed to achieve 90% of state-of-the-art accuracy.

It should be noted that the first group of results (*driving car, running, fighting, eating*) corresponds well to the expectation that the accuracy will be increasing with the length of the shot. The second group (*kissing, getting out of the car, huggung, sitting up*) showed approximatelly constant accuracy depending on the length. This was probably due to the fact that the actions in these shots are recognized based on some short motions inside the actions. The final group (*standing up, sitting down, hand shaking, answering the phone*) showed decreased accuracy depending on the length. The reason is that the actions were too short (length shown using markers in Figure 5.1) and so increasing the length of the shot only "increased noise" and did not bring any additional information. The expectations were also not fulfilled for generally poorly recognised actions.

Based on our experiments, for example, the running activity can be recognized in 10 frames of space-time features with 0.765 accuracy (90% of the state-of-the-art) which corresponds to the 18 frames in total and approximately 0.72s of real-time.

5.5 Summary and proposed future exploitation

The off-line experiments in Section 5.3 show that state-of-the-art results have been achieved. In certain cases, the method even exceeds the state-of-the-art. This method will be referred to as "extended off-line processing". The extended off-line processing has been parametrized and used for verification of the existence of an optimal analysis length of action (as described in Section 5.4). In these experiments, the input size of video samples was restricted as needed by the verification algorithm, and the processing used for obtaining the quality of partial solutions is reffered as "on-line processing". The optimal analysis lengths were obtained for each class in the Hollywood2 dataset. For 4 classes out of 12 in the Hollywood2 dataset, the on-line processing outperformed the state-of-the-art off-line solution.

According to the results achieved, an exploitation is proposed. On-line processing could improve the performance in two real problems:

- action recognition of the live video stream,
- searching of a huge video database.

When a live video stream is processed, an "unlimited" length of the video cannot be processed because the an unlimited computation time would be required, and also the time required to reach a decision would also be "unlimited", which is impossible.

In live video stream processing systems, the decision needs to be made with a reasonable delay. The goal is to reach a delay as close to the theoretical limit of zero, while an unlimited computation time is also an unreachable hard limit. The length of a processed video has to be set rationally and this length directly influences the actual detection delay. If the action length is much longer than the processed sub-shot of the video, the algorithm will be able to detect an action from a few beginning video frames of the action.

In the case of searching through huge databases, the same assertions as the ones in the previous paragraph apply, with one exception. The action must be detected at the analysis position within the sub-shot containing the action. A search through the huge database can be performed at any

possible position; not only at the beginning of the action. This guarantees better quality of search matches. The dependency evaluated in the on-line experiments depicts the relation between the analysis action length and the solution quality which is achievable when sequences with restricted length are used. Nevertheless, the conflict between the detection delay and the accuracy is still here.

The heuristically calculated "optimal analysis action length" is designed to help with the selection of a resonable analysis length because it achieves a good on-line solution quality (for example, 90% of off-line solution) while minimizing the analysis action length.

Chapter 6

Conclusion

The goal of this thesis was to improve human action recognition. The state-of-the-art in human action recognition has been explored, and an off-line recognition system based on multiple types of space-time features was implemented. This off-line system improves upon existing human action recognition solutions in certain situations.

The experiments presented in this thesis showed that the off-line solution was able to outperform the stat-of-the-art off-line methods for 4 of 12 action types from the Hollywood2 dataset; the results were comparable for the other 8 action types.

As the main contribution, the thesis explored a hypothesis that an optimal length of video-segments for recognition of different actions for on-line processing exists. The optimality was defined as a minimal video-segment length which provides close to off-line recognition quality (see Section 5.1). The existence of the optimal analysis length was verified experimentally by a novel algorithm which finds the optimal segment lengths.

The proposed algorithm was used to find the optimal video-segment lengths for on-line recognition for the Hollywood2 dataset with allowable quality drop set to 10%. The performed experiments showed that the optimal lengths exist for all actions in the Hollywood2 dataset and that 11 actions can be detected by using 25 video frames and only one action (fighting) requires 90 video frames. The on-line solution outperformed the off-line recognition as it was actually able to find segment lengths which gave better results than the whole videos from the dataset. The experiments showed that the actions can be divided into three basic groups: (i) the actions for which the recognition quality increases with the segment length, (ii) the actions for which the recognition quality is nearly constant for different segment lengths, (iii) the actions for which the quality decreases with increasing segment length.

The evaluation was performed on a computer cluster due to the large amount of data and high requirements for computational resources. When further using the on-line solution, where optimal analysis length is used, the process is not so computationally demanding as above.

Future usage of results of this work includes the on-line detection in live video streams and content-based search for large video databases. In both cases the smaller detection latency and better accuracy can be achieved. The proposed verification algorithm shall be improved to be able to reach higher efficiency and smaller computational demandingness while finding the optimal analysis length. Overall, the proposed approach could be further adapted to allow automatical conversion of a general off-line recognition system to an on-line system.

Bibliography

[1] Kotelnikov V. A. On the transmission capacity of "ether" and wire in electrocommunications. Moscow, 1933. Svyazzi RKKA.

[2] M. A. Aizerman, E. A. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control,*, number 25, pages 821–837, Berlin, Heidelberg, 1964. Springer Science+Business Media.

[3] Saad Ali and Mubarak Shah. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):288–303, February 2010.

[4] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *Proceedings of the Second International Conference on Human Behavior Unterstanding*, HBU'11, pages 29–39, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] P. L. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, September 2006.

[6] 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.

[7] P. R. Beaudet. Rotationally invariant image operators. In *Proceedings of the 4th International Joint Conference on Pattern Recognition*, pages 579–583, Kyoto, Japan, November 1978.

[8] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

[9] D.P. Bertsekas. *Nonlinear programming*. Athena Scientific optimization and computation series. Athena Scientific, 1999.

[10] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *The Tenth IEEE International Conference on Computer Vision (ICCV'05)*, pages 1395–1402, Beiging, 2005.

[11] W. W. Bledsoe and I. Browning. Pattern recognition and reading by machine. In *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '59 (Eastern), pages 225–232, New York, NY, USA, 1959. ACM.

[12] Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *J. ACM*, 36(4):929–965, October 1989.

[13] Matthew Brown and David Lowe. Invariant features from interest point groups. In *In British Machine Vision Conference*, pages 656–665, Cardiff, UK, 2002. BMVA Press.

[14] Martin D. Buhmann and M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA, 2003.

[15] Peter J. Burt and Edward H. Adelson. Readings in computer vision: Issues, problems, principles, and paradigms. chapter The Laplacian Pyramid As a Compact Image Code, pages 671–679. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[16] O. Chapelle, P. Haffner, and V.N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, Sep 1999.

[17] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part II*, ECCV'06, pages 428–441, Berlin, Heidelberg, 2006. Springer-Verlag.

[18] Jean Ponce David A. Forsyth. *Computer Vision - A modern approach*. Prentice Hall, Upper Saddle River, New Jersey, 2002.

[19] S.L. Devadoss and J. O'Rourke. *Discrete and Computational Geometry*. Princeton University Press, William St 1, Princeton, NJ, 2011.

[20] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proceedings of the 14th International Conference on Computer Communications and Networks*, ICCCN '05, pages 65–72, Washington, DC, USA, 2005. IEEE Computer Society.

[21] A. Elisseeff and M. Pontil. Leave-one-out Error and Stability of Learning Algorithms with Applications. In J. Suykens, , G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Learning Theory and Practice*, NATO ASI Series. IOS Press, Amsterdam; Washington, DC, 2002.

[22] Ivo Řezníček and Pavel Zemčík. On-line human action detection using space-time interest points. In *Zborník príspevkov prezentovaných na konferencii ITAT, september 2011*, pages 39–45. Faculty of Mathematics and Physics, 2011.

[23] Raudies F., Mingolla E., and Neumann H. Active gaze control improves optic flow-based segmentation and steering. *PLoS ONE 7(6): e38446.*, 2012.

[24] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, pages 363–370, Berlin, Heidelberg, 2003. Springer-Verlag.

[25] Greg D. Field, Jeffrey L. Gauthier, Alexander Sher, Martin Greschner, Timothy A. Machado, Lauren H. Jepson, Jonathon Shlens, Deborah E. Gunning, Keith Mathieson, Wladyslaw Dabrowski, Liam Paninski, Alan M. Litke, and E. J. Chichilnisky. Functional connectivity in the retina at the resolution of photoreceptors. *Nature*, 467:673–677 (07 October 2010), 2010.

[26] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[27] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[28] GFosta H. Granlund and Hans Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1995.

[29] J. Han and M. Kamber. *Data Mining, Southeast Asia Edition: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 360 Park Avenue South, New York, NY, 2006.

[30] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[31] D. Hilbert, J.v. Neumann, and L. Nordheim. Über die grundlagen der quantenmechanik. *Mathematische Annalen*, 98(1):1–30, 1928.

[32] Chih-Wei Hsu, Chih chung Chang, and Chih jen Lin. A practical guide to support vector classification, 2010. further information can be found at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.224.4115.

[33] John F. et al. Huges. *Computer Graphics: Principles and Practice (3Rd Ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2014.

[34] S. Hui and S. H. Zak. The widrow-hoff algorithm for mcculloch-pitts type neurons. *Trans. Neur. Netw.*, 5(6):924–929, November 1994.

[35] Nazli Ikizler-Cinbis and Stan Sclaroff. Object, scene and actions: Combining multiple features for human action recognition. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 494–507, Berlin, Heidelberg, 2010. Springer-Verlag.

[36] Ovidiu Ivanciuc. Chapter 6 applications of support vector machines in chemistry. further information can be found at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.104.5640.

[37] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[38] Mihir Jain, Hervé Jégou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *CVPR - International Conference on Computer Vision and Pattern Recognition*, Portland, États-Unis, April 2013.

[39] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, CIVR '07, pages 494–501, New York, NY, USA, 2007. ACM.

[40] Feng Jing, Mingjing Li, Hong-Jiang Zhang, and Bo Zhang. Support vector machines for region-based image retrieval. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 2, pages II–21–4 vol.2, Baltimore, MD, July 2003.

[41] Alexander Klaser, Marcin Marszalek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *In BMVC 08*, Leeds, UK, 2008.

[42] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 2011.

[43] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In Jerzy Neyman, editor, *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkley, CA USA, 1950. University of California Press, Berkeley, CA, USA.

[44] Rakesh Kumar. Aerial video survelliance and exploitation. In Paolo Remagnino, GraemeA. Jones, Nikos Paragios, and CarloS. Regazzoni, editors, *Video-Based Surveillance Systems*, pages 29–38. Springer US, 2002.

[45] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, September 2005.

[46] Ivan Laptev and Tony Lindeberg. Interest point detection and scale selection in space-time. In LewisD. Griffin and Martin Lillholm, editors, *Scale Space Methods in Computer Vision*, volume 2695 of *Lecture Notes in Computer Science*, pages 372–387. Springer Berlin Heidelberg, 2003.

[47] Ivan Laptev and Tony Lindeberg. Space-time interest points. In *In IEEE International Conference on Computer Vision (ICCV)*, pages 432–439, Nice, France, 2003.

[48] Ivan Laptev, Marcin Marszałek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *In Conference on Computer Vision & Pattern Recognition*, Anchorage, Alaska, USA, 2008.

[49] Q.V. Le, W.Y. Zou, S.Y. Yeung, and A.Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3361–3368, 1420 Austin Bluffs Pkwy, Colorado Springs, CO USA, June 2011.

[50] Chung-Ching Lin and M. Wolf. Detecting moving objects using a camera on a moving platform. In *20th International Conference on Pattern Recognition (ICPR)*, pages 460–463, Istanbul, Turkey, Aug 2010.

[51] Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, pages 224–270, 1994.

[52] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, November 1998.

[53] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos "in the wild". In *In IEEE International Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, 2009.

[54] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision - Volume 2*, ICCV '99, Washington, DC, USA, 1999. IEEE Computer Society.

[55] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[56] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[57] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkley, CA USA, 1967.

[58] M. L. Minsky and S. A. Papert. *Perceptrons*. MIT Press, Cambridge, MA USA, 1969.

[59] A.M.F. Mood and F.A. Graybill. *Introduction to the Theory of Statistics*. International Student Edition: McGraw-Hill Series in Probability and Statistics. McGraw-Hill Book Company, Inc., New York, NY USA, 1963.

[60] Raphael Ortiz. Freak: Fast retina keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society.

[61] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008.*, Anchorage, Alaska, USA, June 2008.

[62] Ivo Reznicek and Pavel Zemcik. Action recognition using combined local features. In *Proceedings of the IADIS Computer graphics, Visulisation, Coputer Vision and Image Processing 2013*, pages 111–118, Prague, Czech Republic, 2013. IADIS.

[63] Ivo Reznicek and Pavel Zemcik. Human action recognition for real-time applications. In *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods*, pages 646–653, Angers, France, 2014.

[64] M.D. Rodriguez, J. Ahmed, and M. Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008.*, Anchorage, Alaska, USA, June 2008.

[65] F. Rosenblatt. Neurocomputing: Foundations of research. chapter The Perception: A Probabilistic Model for Information Storage and Organization in the Brain, pages 89–114. MIT Press, Cambridge, MA, USA, 1988.

[66] Chengxiong Ruan, Qiuqi Ruan, and Xiaoli Li. Real adaboost feature selection for face recognition. In *IEEE 10th International Conference on Signal Processing (ICSP), 2010*, pages 1402–1405, Beijing, China, Oct 2010.

[67] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[68] S. Salivahanan and A. Vallavaraj. *Digital Signal Processing*. McGraw-Hill Education (India) Pvt Limited, B-4, Sector 63, Noida, India, 2000.

[69] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36 Vol.3, Cambridge, England, UK, Aug 2004.

[70] Shokri Z. Selim and M. A. Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):81–87, January 1984.

[71] Raymond Serway and John Jewett. *Principles of Physics: A Calculus-Based Text vol. 1.* Cengage Learning, Independence, KY USA, 2006.

[72] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, Ithaca, NY, USA, 1993.

[73] Ivan Sipiran and Benjamin Bustos. A robust 3d interest points detector based on harris operator. In *Proceedings of the 3rd Eurographics Conference on 3D Object Retrieval*, EG 3DOR'10, pages 7–14, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[74] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606, April 2009.

[75] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005.*, volume 2, pages 150–156 vol. 2, San Diego, CA, USA, June 2005.

[76] Robert M. Solovay. A model of set-theory in which every set of reals is Lebesgue measurable. *Annals of Mathematics Second Series*, 92, 1970.

[77] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 438–451, Berlin, Heidelberg, 2010. Springer-Verlag.

[78] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. Society of Industrial and Applied Mathematics, Philadelphia, PA USA, 1997.

[79] H. Uemura and S. Ishikawa K. Mikolajczyk. Feature tracking and motion compensation for action recognition. In *In BMVC*, Leeds, UK, 2008.

[80] Muhammad Muneeb Ullah, Sobhan Naderi Parizi, and Ivan Laptev. Improving bag-of-features action recognition with non-local cues. In *BMVC*, Aberystwyth, Wales, UK, 2010. British Machine Vision Association.

[81] V.N. Vapnik. *Statistical learning theory*. Adaptive and learning systems for signal processing, communications, and control. John Wiley & Sons Inc., New York, NY USA, 1998.

[82] Heng Wang, A. Klaser, C. Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 3169–3176, Washington, DC, USA, 2011. IEEE Computer Society.

[83] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proceedings of the British Machine Vision Conference*, London, England, UK, 2009. BMVA Press.

[84] Yunfei Wang, Zhaoxiang Zhang, and Yunhong Wang. Moving object detection in aerial video. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 446–450, Boca Raton, Florida, USA, Dec 2012.

[85] Daniel Weinland, Edmond Boyer, and Remi Ronfard. Action recognition from arbitrary views using 3d exemplars. In *In IEEE International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil, 2007.

[86] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2), November 2006.

[87] B. Widrow and E. Walach. *Adaptive Inverse Control, Reissue Edition: A Signal Processing Approach*. John Wiley & Sons, Inc., Hoboken, New Jersey USA, 2008.

[88] Geert Willems, Tinne Tuytelaars, and Luc Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 650–663, Berlin, Heidelberg, 2008. Springer-Verlag.

[89] J. Wu. *Advances in K-means Clustering: A Data Mining Thinking*. Springer Theses. Springer-Verlag, Berlin, Heidelberg, 2012.

[90] Jianguo Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. In *Conference on Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06.*, New York, NY, USA, 2006.

Appendix A

Hollywood2 dataset sample images

The dataset contains these twelve action classes: *answering the phone, driving car, eating, fighting, getting out of the car, hand shaking, hugging, kissing, running, sitting down, sitting up and standing up*. All videos are obtained from hollywood movies, from an archive located in: `ftp://ftp.irisa.fr/local/vistas/actions/Hollywood2-actions.tar.gz`. Subsequently, the illustrative examples will be presented herein.

- driving car

- eating

- fighting

- getting out of the car

- hand shaking

- hugging

- kissing

- running

- sitting down

- sitting up

- standing up

- answering the phone