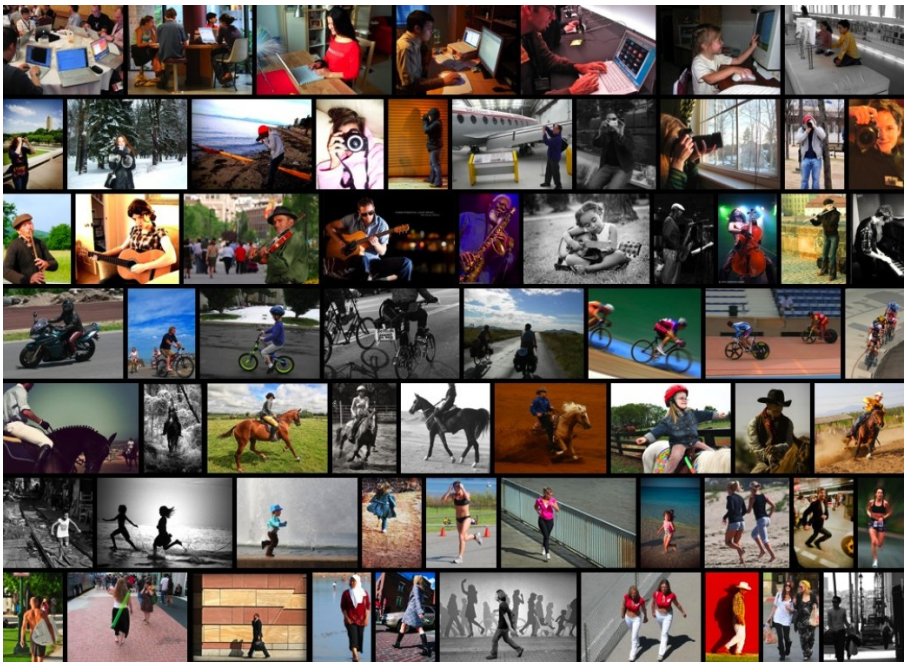
Human Action Recognition using Bag of Features



Jesper Birksø Bækdahl

Master's Thesis

May 2012

Supervisor: Thomas Moeslund

Vision, Graphics and Interactive Systems

Aalborg University

AALBORG UNIVERSITET

Elektroniske Systemer
Institut 8
Fredrik Bajers Vej 7
9220 Aalborg Ø
http://es.aau.dk/

Title:

Human Action Recognition using Bag of Features

Project period:
9 - 10. semester,
August 2011 - May 2012

Project group:
11gr924

Group members:
Jesper Birksø Bækdahl

Supervisor:
Thomas Moeslund

Pages: 51

Finished: May 31st 2012

Abstract:

This thesis explores the popular bag-of-features framework for human action recognition. Different feature detectors and descriptors are described and compared.
The combination of the Harris3D detector and the HOG/HOF descriptor is chosen for use in the experimental setup. Unsupervised and supervised classification methods are compared to show the difference in performance. The supervised algorithm used is a support vector machine, and the unsupervised algorithms are k-means clustering, and affinity propagation, where the latter has not been used before for action recognition. The algorithms are tested on two different datasets. The simple KTH dataset with 6 classes, and the more complicated UCF50 with 50 classes.
The SVM classification obtains good results on both KTH and UCF50. Of the two unsupervised methods, affinity propagation obtains the best performance. SVM outperforms both of the unsupervised algorithms. The strategy used for building the vocabulary, which is a central part of the bag-of-features framework is tested. The results show, that increasing the number of words will slightly increase the classifier performance.

Preface

This report documents the Master's thesis: "Human Action Recognition using Bag of Features" during the 9th and 10th semester of the specialisation: "Vision Graphichs and Interactive Systems".

The software developed in this project has been submitted to the Aalborg University project database[1], and can also be found on the authors website[2].

The front page of shows examples of different actions from a dataset of still images presented in [7].

Acknowledgements

I would like to thank:

Dr. Mubarak Shah and University of Central Florida for hosting me from autumn 2011 to summer 2012. The inspiration and help i received there has been a great support during this project.

My supervisor Thomas Moeslund for making it possible for me to go to UCF, and for supporting me both professionally and personally during my stay.

Bhaskar Chakraborty and Michael Holte for supplying me with action recognition code and advice.

[1] http://projekter.aau.dk/projekter/
[2] http://kom.aau.dk/~baekdahl/masters/

Contents

Chapter 1

Introduction

This thesis addresses the task of human action recognition. Action recognition can be described as recognizing what is going on in a video using a computer algorithm. Describing the actions in a video have a large number of applications, and as the amount of video data grows, and our technology becomes more capable, the demand for action recognition grows. Historically, action recognition has mainly been a human task, but automating the process, or parts of it, could provide a number of advantages.

(a) Facial expression classification. [36]

(b) Soccer juggling. UCF50 dataset. [25]

(c) Detection of unusual crowd activity. [26]

Figure 1.1: Examples of different applications and abstractions of human action recognition.

The number of surveillance cameras gathering footage of areas with human activity is increasing each day. The cameras are used by e.g. law enforcement to monitor cities for criminal activity. The large number of cameras in a big city requires many people to monitor the cameras, with an increasing chance of missing important events in the videos. A computer vision based solution would make it possible to monitor most, if not all, video sources at the same time to look for events that requires further human interaction. This would make it possible for a small team of people to monitor a whole city. By configuring the system to look for certain events, it can alert the user if one of these events occur so he can take further action. Figure 1.1c shows an example of action recognition research in the area of surveillance. The proposed application is to detect abnormal crowd activity such as a panic, where people flee from a scene as seen in figure 1.1c.

It is easier than ever to capture video with e.g. a smartphone and upload it to the Internet. The popular video sharing site Youtube.com receives one hour of new video uploaded every second [39]. The person uploading the video has the option to describe the content of the video with a description and tags, which can later be used to locate the video in a search. It is then possible to locate a video that contains a person playing soccer, as in figure 1.1b, if this has been input

Figure 1.2: Children playing a game controlled via motion capture by Microsoft Kinect [21].

in the title and/or description of the video. But what if a user is interested in searching for all goals or free kicks in a soccer match between two teams? It is not common for a description of a video to include this many details because the user who uploaded the video has to input it manually. This kind of detailed search could be possible if the video had been annotated by an action recognition system that recognizes such events. It could even be possible to locate the exact time instant where a goal happens. Using action recognition for video categorization and annotation will enable us to make better use of the vast amounts of video data in ways that are not possible today.

The progress in video capture and processing hardware has also spawned products like the Microsoft Kinect, that acts as a motion capture controller for a gaming console. The Kinect combines action, gesture and voice recognition for a total interactive experience, as seen in figure 1.2, without the use of an old-fashioned joystick. The system is robust enough to work in a normal living room with varying lighting and background clutter. The idea of gesture control is extending to devices such as televisions, where we soon will be able to change channels or lower the volume with the use of gestures.

Action recognition remains a challenging problem, and so far only the tip of the iceberg has been revealed. Most of the existing solutions only work for simple or synthetic scenes which have been created for the purpose of testing action recognition algorithms. The best solutions still require large amounts of supervised training data, which is expensive to obtain. This leads us to the following initiating problem:

- **What is the state of the art in human action recognition, and what is the best way to take advantage of the large number of unlabelled videos?**

Chapter 2

Analysis

This section describes the different methods and algorithms used for Human Action Recognition. Different datasets are presented to show that action recognition has evolved from classifying simple synthetic datasets to new challenging realistic datasets. The main frameworks and models used to represent an action recognition problem are presented in section 2.3 along with the bag-of-features framework which is the one used in this thesis. Finally a problem statement is presented to give a concise description of the contribution and scope of this thesis.

2.1 Human Action Recognition

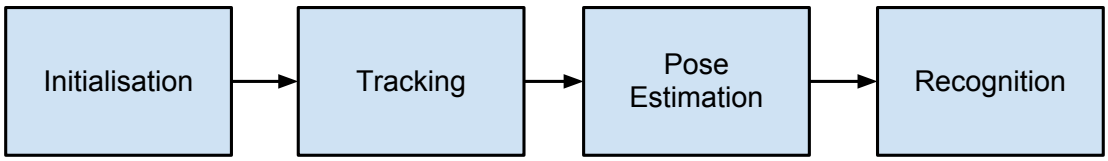


Figure 2.1: Taxonomy for describing the parts of a Human Motion Capture systems, introduced by Moeslund and Granum [22].

The area of human action recognition is closely related to other lines of research within a common field known as Human Motion Capture. There are different ways of structuring the different disciplines of human motion capture, and defining where one ends and another begins. One notable way of structuring the problem, is the taxonomy introduced by Moeslund and Granum [22]. The same taxonomy is later reused by Moeslund et al. [23] and has also been adopted by many other authors. Figure 2.1 shows the structure, which consists of: initialisation, tracking, pose estimation and recognition. First the system is initialised. Next, the subject is then tracked to obtain the position in the scene, which implies a figure-ground segmentation to separate the subject from the background. Pose estimation might be part of the output or be used as input to the recognition process. Recognition classifies the actions performed by the subject. A system does not need to include all four processes, and some processes might be implicitly performed by one of the subsequent processes. Example of this can be found in action recognition systems described in the following sections, where actions are classified in a clip, but the position and pose of the actor is unknown.

2.2 Datasets

In this section four action recognition datasets are presented: Weizmann, KTH, Hollywood and UCF50. Weizmann and KTH are both scripted datasets, where the actors have been told to perform a specific action in a controlled setting with only minor changes in viewpoint, background and illumination. This makes them easy to classify, and every state of the art algorithm is able to get over 90% average precision, which makes these datasets less interesting to use as a benchmark. Developing an algorithm that solves these datasets does not necessarily mean that the algorithm will be good at solving the more advanced datasets because the challenges there are different.

Hollywood and UCF50 have been sampled from existing video, that was not created for the purpose of action recognition. This provides a more realistic environment but also much more challenging. The huge differences in the videos mean that an algorithm have very few conditions to rely on. It has to be adaptable to different viewpoints, cluttered backgrounds, multiple actors, occlusions, etc. These conditions makes it much harder to obtain good results, especially for algorithms that depend on e. g. being able to get a good segmentation.

2.2.1 Weizmann

The Weizmann dataset, figure 2.2 [1], contains 10 different actions: walk, run, jump, gallop sideways, bend, one-hand wave, two-hands wave, jump in place, jumping jack and skip. Each action is performed by 10 different actors resulting in a total of 100 clips. The viewpoint and background are static and equal for all clips.

Figure 2.2: Example frames from Weizmann dataset.

2.2.2 KTH

KTH, figure 2.3 [32], contains 6 different actions: walking, jogging, running, boxing, hand waving and hand clapping. The actions are performed by 25 different actors in four different scenarios: outdoors, outdoors with zooming, outdoors with different clothing and indoors. Compared to Weizmann, KTH has considerable amounts of intraclass differences. There are differences in duration and somewhat in viewpoint.

2.2.3 Hollywood

The original Hollywood dataset, figure 2.4 [16], contains 8 different actions. It has later been extended with 4 additional actions [20]. The actions are: answer phone, get out of car, handshake, hug, kiss, sit down, sit up, stand up, drive car, eat, fight, run. There are approximately 150 samples per class. All actions are taken from 69 different Hollywood movies, resulting in huge variety between viewpoint, background and action performance.

Figure 2.3: Example frames from KTH dataset.

Figure 2.4: Example frames from Hollywood dataset.

2.2.4 UCF50

UCF50 [25] is an extension of the UCF Youtube dataset. The dataset has 50 different actions having at least 100 clips for each action for a total of 6681 clips. The actions are all user-submitted videos from Youtube.com, and by that nature very different from each other. Many of the videos include large amounts of camera motion, illumination changes and cluttered backgrounds.

The actions in UCF50 are: baseball pitch, basketball shooting, bench press, biking, biking, billiards shot,breaststroke, clean and jerk, diving, drumming, fencing, golf swing, playing guitar, high jump, horse race, horse riding, hula hoop, javelin throw, juggling balls, jump rope, jumping jack, kayaking, Lunges, military parade, mixing batter, nun chucks, playing piano, pizza tossing, pole vault, pommel horse, pull ups, punch, push ups, rock climbing indoor, rope climbing, rowing, salsa spins, skate boarding, skiing, skijet, soccer juggling, swing, playing tabla, taichi, tennis swing, trampoline jumping, playing violin, volleyball spiking, walking with a dog, and yo yo.

2.3 Action Recognition Methods

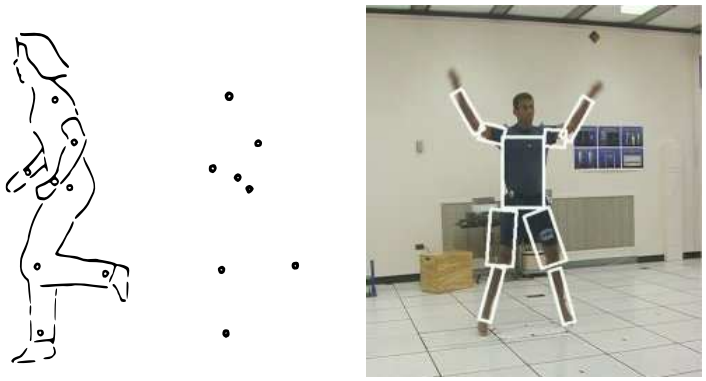
This section provides an overview of the different approaches to action recognition, taken from various surveys [22, 23, 27, 37, 41].

2.3.1 Body models

One of the most intuitive approaches to action recognition is body models. To recognize an action performed by a human, the different body parts are detected according to a body model. The model constraints how one body part is positioned compared to the others, e. g. a foot can not

Figure 2.5: Example frames from UCF50 dataset.

be connected to an arm, and the angle between the upper arm and forearm can only be within the range that is physically possible with the human join configuration. The recognition works by comparing the movement performed by the body model extracted from a video with existing ground truth body models. The ground truth body models can be generated synthetically or taken from motion capture of a human performing the action in a controlled setting, where markers or other remedies are used to obtain a ground truth body model. There are two common approaches in the use of body models: recognition by reconstruction and direct recognition.



(a) Moving Light Display(MLD). [12]

(b) Body model based on rectangular patches. [28]

Figure 2.6: Examples of body models.

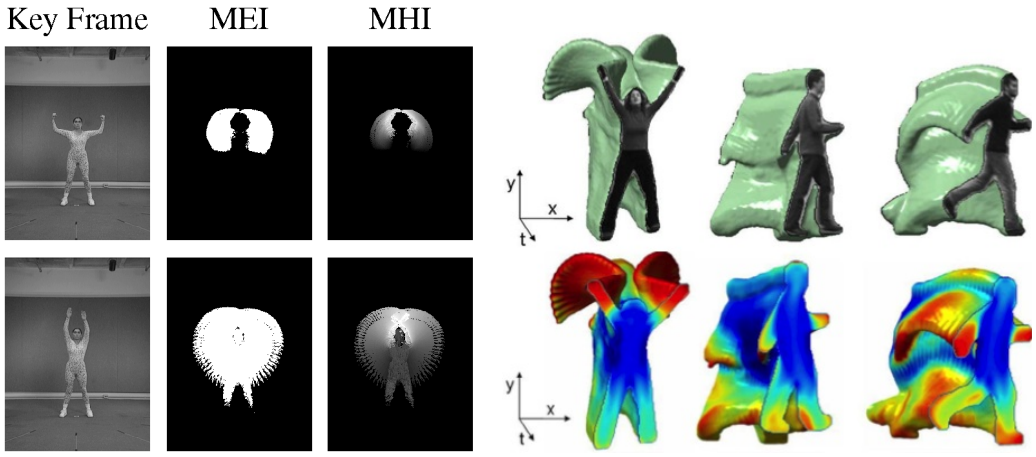
Recognition by reconstruction is a 3D approach, where the motion of the body is captured and a 3D model of the body is estimated. This is a challenging approach because of the large number of degrees-of-freedoms of the human body. Marr and Nishihara [19] proposed a body model consisting of cylindrical primitives . Using such a model is called a top down approach. This refers to the fact that the 3D body models are found by matching 2D projections sampled in the search space of joint configurations. The opposite method is called bottom-up and works by tracking body parts in 2D and subsequently lifting the 2D model into 3D. An example of the

bottom-up approach is seen in figure 2.6b, where the body parts are tracked using rectangular patches and lifted into 3D using an orthographic camera model.

Direct recognition works on a 2D model without lifting this model into 3D. Many different 2D models have been investigated. Some models consist of anatomical landmarks like joint positions. This was pioneered by Johansson [12], who showed that humans can recognize actions merely from the motion of a few moving light displays(MLD) attached to the human body as seen in figure 2.6a. Other models use stick figures which can be detected from a space-time volume.

2.3.2 Holistic approaches

The holistic approach, also referred to as global model or image model, encodes the whole subject as one entity. This often involves locating a ROI via segmentation and/or tracking. Common representations are silhouettes, edges or optical flow. This representation can be much simpler and faster to compute than body model, but still be as discriminative in problems consisting of a large number of classes.



(a) MEI and MHI for 3 different ballet actions. [2]

(b) Space-time shapes. [1]

Figure 2.7: Examples of holistic methods.

Silhouettes have proved to be a useful global representation of a subject. The silhouette can be found by background subtraction, and can be encoded in a contour or by the area inside. Silhouettes are insensitive to color, texture and contrast changes, but have problems with self-occlusion and depend on robust segmentation. An early examples of the use of silhouettes, extracts silhouettes from each frame of the video which results in a motion energy image(MEI). A motion histogram image(MHI) is also computed which is function of the history at that pixel over time as seen in figure 2.7a. Silhouettes can also be combined over time to form space-time shapes as shown in figure 2.7b.

Other holistic approaches are based on flow and gradients. This works by extracting optical flow or gradients within the ROI or in the entire image. A grid is often used to divide the extraction area to partly overcome partial occlusions and changes in viewpoint. This grid can be defined in different ways. Danafar and Gheissari [6] divides the ROI into horizontal slices corresponding approximately to head, body and leg, and use optical flow extracted in these areas. Histograms of oriented gradients(HOG) have also been used to represent the ROI by a number of histograms that consists of gradient directions.

2.3.3 Local features

Local features describes the action in a video as a set of local descriptors instead of one global descriptor as in the global approach. These local descriptors are combined in one representation of the video, which can be done in different ways to either ignore, or maintain the spatial and/or temporal relationship between the features. The features can either be extracted densely or at spatio-temporal interest points(STIPs), which are points that contain information important to determining the action in the video.

STIP detectors usually works by evaluating the response of one or more filters applied to the video. Laptev [14] extended the Harris corner detector to 3D and detects STIPs by finding local maxima of the extended corner function. Dollár et al. [8] uses a Gabor filter in the temporal dimension, and overcomes the problem of too few interest points that predecessors had. Rapantzikos et al. [29] apply discrete wavelet transform in spatial and temporal direction of the video, and uses a filtered version to detect salient regions. Willems et al. [38] identifies saliency as the determinant of a 3D Hessian matrix. Chakraborty et al. [3] filters out points that belong to the background using a surround suppression measure. The final points are selected based on point matching and Gabor filtering in the temporal direction.

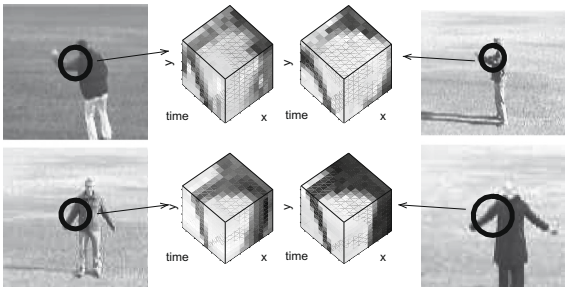


Figure 2.8: Extraction of space–time cuboids at interest points from similar actions performed by different persons [15].

The feature descriptor should capture the important information in the area around a STIP and represent it in a sparse and discriminative way. The size of the volume or patch used to compute the descriptor is usually determined by the detection scale. Examples of volumes extracted at interest points is seen in figure 2.8. Schuldt et al. [32] uses patches based on normalized derivatives with respect to space and time. Similar to the global approach, descriptors can also be grid-based like HOG [5]. Many of these grid-based methods have also been extended to 3D. ESURF features is an extension of the SURF descriptor to 3D introduced in [38]. HOG is extended to HOG3D by Kläser et al. [13]. A 3D version of the popular 2D descriptor SIFT was introduced by Scovanner et al. [33].

The set of feature descriptors in a video needs to be combined to form a representation that makes is possible to compare videos to each other. A popular representation is the bag-of-features representation [32], where a vocabulary is generated by clustering feature descriptors from all possible classes into a number of words. Each video is then represented by a histogram that counts the number of occurrences of each possible word in the video. The histograms can then be used to train a classifier, like e. g. a support vector machine(SVM).

2.3.4 Un- and semisupervised approaches

The amounts of unlabelled data is increasing and there is a high cost of labelling the large amounts of data needed for typical machine learning applications. This makes methods that require little or no supervised data become attractive. Supervised methods have so far received far more attention, but the interests in un- and semisupervised methods is growing.

Niebles et al. [24] proposed unsupervised classification of bag-of-features histogram using latent topic models such as the probabilistic Latent Semantic Analysis(pLSA) model and Latent Dirichlet Allocation (LDA). Savarese et al. [31] replaced bag-of-features with spatio-temporal correllograms to better describe the temporal pattern of the features. Dueck and Frey [9] uses affinity propagation [10] for unsupervised learning of image categories, obtaining better results than using k-means clustering.

One of the most simple and intuitive methods for semisupervised learning is self-training. In self-training a supervised classifier is trained on available the labelled samples. The unlabelled points are then classified using this weakly trained classifier, and the most confident points are used to retrain the classifier together with the labelled points. This is repeated until all points are used. Rosenberg et al. [30] used self-training for object detection. A common way of developing a semisupervised algorithm is to extend an unsupervised algorithm to respect constraints. One notable example is constrained expectation maximization [34].

2.4 Framework

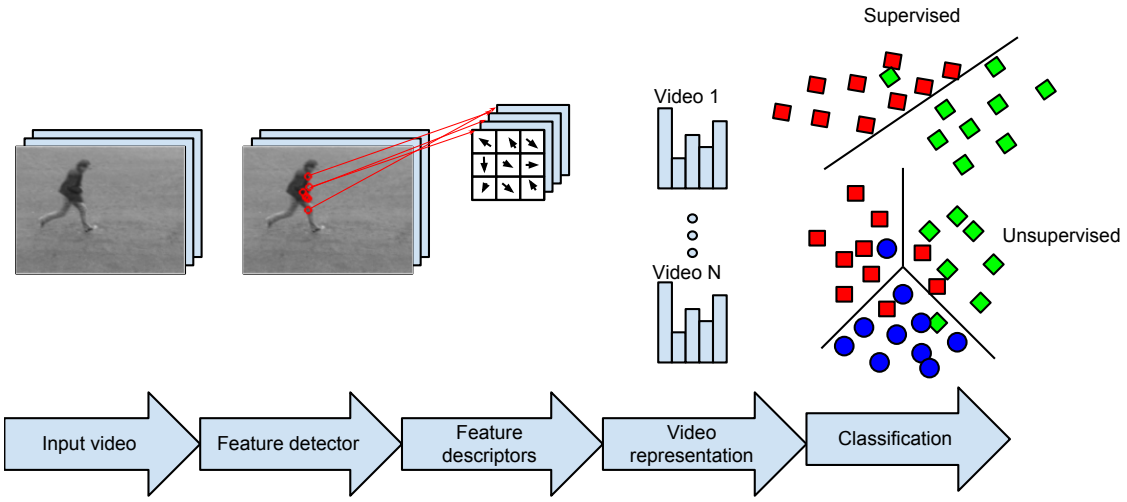


Figure 2.9: Human action recognition framework.

This section describes the action recognition framework that is used in this thesis. The framework focuses on a subset of the methods presented in the former sections. The representation used is local features in a bag of words framework, because it is a promising direction for describing the more advanced datasets like UCF50. Supervised and unsupervised classification will be compared, to see how training data affects the recognition rate. There has not been many attempts to do unsupervised classification within this framework, so this will be explored in this thesis by a test of the clustering algorithm: affinity propagation that, to our knowledge, has not been used before in action recognition.

The framework consists of four main components: feature detector, feature descriptor, video representation and classification. The different parts of this framework can be replaced with different algorithms to form different combinations and thereby produce different results.

Feature detector

The first step is to detect interest points in the video, which are the positions where the feature descriptors are computed. These points should ideally be located at places in the video where the action is taking place.

Feature descriptor

The feature descriptor encodes the information in the area of an interest point into a representation suited for representing the action. The feature descriptor should ideally be invariant to changes like orientation, scale and illumination to be able to match features across different kinds of videos.

Video representation

The set of local feature descriptors in a video has to be combined into a representation that enables the comparison with other videos. The most popular method is the bag-of-features representation, where the spatial and temporal locations of the features are ignored. Other methods tries to take the relationship between the features into account.

Classification

The classification step can be: unsupervised, semi-supervised or supervised.

In the unsupervised approach, we assume that we do not know the labels of any of the videos. The videos are then grouped together based on their similarities. The number of groups used for unsupervised learning can be given as part of the problem, or can be dynamic where different partitions of videos corresponds to different semantically meanings.

In semi-supervised classification there is some prior knowledge about the videos, which can consist of a few labelled samples, or e. g. a constraint saying that sample a and b are from different classes without giving the label. Semi-supervised learning is not explored further in this thesis.

Supervised classification uses a large number of training samples to train a classifier.

2.5 Problem Statement

The analysis described different approaches to action recognition, and a specific framework was chosen to examine further. The rest of this thesis tries to answer the following problem statement:

- **What are the pros and cons of the different components in the bag-of-features framework, and which combination yields the best performance?**
  - How does unsupervised clustering algorithms compare to supervised classification?
  - How does the performance of the methods change when the complexity of the dataset change?
  - What is the best strategy for bag-of-features vocabulary building?

2.5.1 Scope of this thesis

This thesis compares methods within the beforementioned framework, and does then not consider all other possible action recognition frameworks beyond the brief introduction given in section 2.3. All video clips are presumed to contain exactly one action being performed, i. e. the described framework does not worry about action detection, multiple actors or multiple actions in the same clip.

2.6 Thesis outline

The rest of this thesis seeks to answer the questions asked in the above problem statement. Notable examples of each component of the bag-of-features framework is described. Section 3 and 4 presents feature detectors and descriptors, respectively. A comparison of detector and descriptor combinations is given in section 5. The bag of words representation is described in section 6. Supervised and unsupervised classification is presented in section 7 and 8, respectively. Finally the results of classification using various combinations of methods are presented in 9.

Chapter 3

Feature Detectors

This section describes in detail some of the more notable local feature detectors. A feature detector finds the points in the video where features are going to be extracted. These points are known as Spatio-Temporal Interest Points(STIPs). A STIP is a point in space and time (x, y, t) that has high saliency. High saliency means that there are high amounts of changes in the neighbourhood of the point. In the spatial domain this shows as large contrast changes, yielding a Spatial Interest Point(SIP). Saliency in the temporal domain occurs when a point changes over time, and when this change occurs at a SIP the point is then a STIP.

The motivation for locating areas in the video having high saliency, is that they must be the important areas for describing the action in the scene. This can be confirmed by observing a video of a person running. The difference in appearance between the person and the background will result in high spatial saliency all around the contour of the person. The fact that the person is running results in high temporal saliency at the same points, thus giving rise to STIPs.

Figure 3.1: Example of good and bad detected feature points. Video is class "running" from KTH dataset.

In the human action recognition, the detected STIPs should ideally be located on the actor performing the action that the video represents. The STIPs on a human body will represent action primitives such as: "moving leg" or "lifting arm", and it is the combinations of the STIPs detected in a video that discriminates one video from another.

To achieve good discrimination between classes, it is therefore desirable to maximize the amount

of good STIPs, i. e. STIPs on the actors, versus bad STIPs, i. e. STIPs on the background or on motion that does not represent the action in the video. Figure 3.1 shows a frame of detected STIPs of "person running" from the KTH dataset. The detector has detected three points on the background, which will not add any useful information about the action.

3.1 Scale space

In the more challenging datasets, there can be large intraclass differences between videos. Videos have different camera viewpoint, scene composition, resolution, etc. This results in that the same object, e. g. a human being, in one video can have the size of hundreds of pixels, while a human in another video only takes up about 50 pixels of space. To be able to detect similar features in these videos, STIPs are detected in different temporal and spatial scales. These scales are represented as a convolution with a Gaussian blur function, where higher values of the variances σ^2, τ^2, represents larger scales. This has intuitive meaning because the more the video is blurred the more small details are lost, leaving only larger scale details behind. A video $f(x, y, t)$ is then represented by the scale-space

$$L(x, y, t; \sigma^2, \tau^2) = f * g(\cdot; \sigma_l^2, \tau_l^2) \tag{3.1}$$

where g is the spatio-temporal Gaussian kernel

$$g(x, y, t; \sigma_l^2, \tau_l^2) = \frac{exp(-(x^2 + y^2)/2\sigma_l^2 - t^2/2\tau_l^2)}{\sqrt{(2\pi)^3 \sigma_l^4 \tau_l^2}} \tag{3.2}$$

having spatial variance σ_l^2 and temporal variance τ_l^2. Figure 3.2 shows the scale space of two different videos from the same class: "Horse race". In figure 3.2a, b and c the camera is close to the race, while in 3.2d, e and f it is further away. This means that the number of features found on the horse and rider will be larger, and represent smaller details in the close up video than the one with distance. Scale space representation can compensate for this as the blur removes the details. The features in 3.2c can then be comparable to the ones in 3.2d allowing the videos to represent the same class even though their view point differs.

(a) Original (b) $\sigma_l^2 = 4$ (c) $\sigma_l^2 = 16$

(d) Original (e) $\sigma_l^2 = 4$ (f) $\sigma_l^2 = 16$

Figure 3.2: Parts of scale space representation of 2 videos from class: "Horse race" from UCF50.

There are multiple approaches to find the best points across scales. The trivial method is to apply a multi-scale approach where the resulting points are the collection of all points from all scales. Another way is to try to estimate the best set of points across scales.

3.2 Harris

The Harris corner detector is a well-known algorithm for detection of salient regions in images [11]. The detector operates on a windowed second moment matrix

$$\mu = g(\cdot; \sigma_i^2) * \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix} \tag{3.3}$$

where L_ξ are derivatives of the image scale space

$$L_\xi(x, y; \sigma_l^2) = \frac{\partial L(x, y; \sigma_l^2)}{\partial \xi} \tag{3.4}$$

and σ_i is the variance used in the gaussian kernel that serves as a window function. It is also called the integration scale. The eigenvalues of μ determines if there is a corner, an edge or a "flat" area. If λ_1 and λ_2 both are large the point is a corner. If $\lambda_1 >> \lambda_2$ or $\lambda_2 >> \lambda_1$ the point is an edge, and if both eigenvalues are small the area is "flat".

Since the computation of eigenvalues is expensive, the corners are detected as local maxima of the corner function

$$H = det(\mu) - ktrace^2(\mu) = \lambda_1\lambda_2 - k(\lambda_1 + \lambda 2)^2 \tag{3.5}$$

3.3 Harris3D

Laptev [14] extended the idea of the Harris corner detector to three dimensions. The second-moment matrix is now spatio-temporal

$$\mu = g(\cdot; \sigma_i^2, \tau_i^2) * \begin{pmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{pmatrix} \tag{3.6}$$

and the gaussian window function is now spatio-temporal having temporal variance τ_i. The corner function becomes

$$H = det(\mu) - ktrace^3(\mu) = \lambda_1\lambda_2\lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3 \tag{3.7}$$

The implementation of the Laptev detector used in this thesis is the one provided on his webpage which also bundles the feature extractor for HOG/HOF features. The original algorithm [14] uses scale selection in space-time [17], but the newer implementation used in this thesis uses a multi-scale approach.

3.4 Cuboid detector

Dollár et al. [8] reports that the Harris3D detector in many cases does not detect enough features to perform well. The Cuboid detector is consequently tuned in a way that results in more STIPs than Harris3D for the same videos. It is sensitive to periodic motions which occur often in action videos. The response function is

$$R = (f * g(\cdot, \sigma_l^2) * h_{ev})^2 + (f * g(\cdot, \sigma_l^2) * h_{od})^2 \tag{3.8}$$

where h_{ev} and h_{od} are the quadrature pair of one dimensional Gabor Filters, which are applied temporally and g is two dimensional Gaussian function applied spatially.

3.5 Selective STIP

Selective STIP [3] tries to combine the best of existing approaches with a novel way of point suppression to strengthen the quality of detected STIPs. Instead of computing a response in space and time simultaneously, the algorithm starts out with a simple detection of Spatial Interest Points(SIPs) using Harris. For each interest point, an inhibition term is computed. The idea behind the inhibition term is that points belonging to the background follow a particular geometric pattern. For this purpose, a gradient weight factor is introduced

$$\Delta_{\Theta,\sigma}(x, y, x-u, y-v) = |cos(\Theta_\sigma(x, y) - \Theta_\sigma(x-u, y-v))| \tag{3.9}$$

where $\Delta_{\Theta,\sigma}(x, y)$ is the image gradient and u and v is the size of the mask. A suppression term is then defined as the weighted sum of gradient weights in the suppression surround of that point

$$t_\alpha(x, y) = \int\int_\Omega C_\sigma(x-u, y-v) \times \Delta_{\Theta,\sigma}(x, y, x-u, y-v)\mathrm{d}u\mathrm{d}v \tag{3.10}$$

where Ω denotes the image coordinate domain. The resulting measure of corner strength, is the difference between original Harris corner strength $C_\sigma(x, y)$ and the suppression term

$$C_\sigma(x, y) - \alpha t_\sigma(x, y) \tag{3.11}$$

where the factor α controls the amount of suppression.

Scale selection is then applied to the remaining points in the same way as the original Harris3D[17], and the n best SIPs are kept as the final set of suppressed SIPs.

The SIPs are then temporally constrained by point matching and Gabor filtering. The points are matched between two frames n and $n-1$, and matching points are removed since static points do not contribute to any motion information. The final points are selected based on the Gabor response at their locations. The resulting set of STIPs are usually sparser having a higher number of STIPs on the actors in the scene.

3.6 Dense Sampling

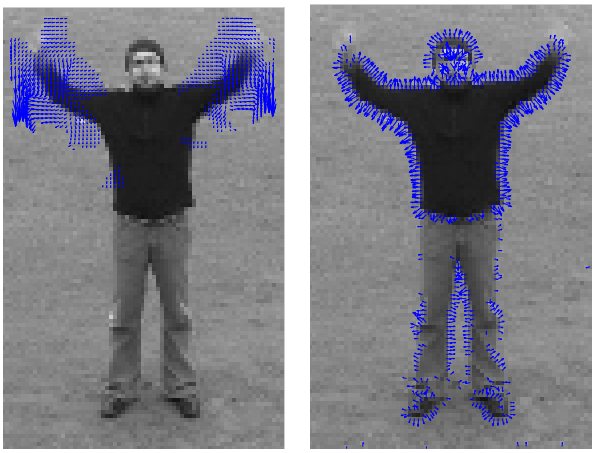
Instead of detecting feature points, feature points can be sampled uniformly over the whole frame at every time instant in the video. The points are often distributed in a way the results in a certain amount of both spatial and temporal overlap to be sure to capture all the information in the video. Dense sampling results in massive amounts of information and comes close to the raw pixel representation of the video. The amounts of bad features that dense sampling includes is outweighed by the fact that it does not miss any information.

Chapter 4

Feature Descriptors

Feature descriptors represents the underlying pixels in a way that maximizes classification performance and gives a sparse representation. The descriptors should ideally be invariant to operations like scale, rotation and illumination changes. This invariance enables descriptors to be matched across videos which have differences in these parameters.

The descriptors are local, meaning that they are extracted in a predefined area around the detected STIPs. This area is often proportional to the scale where the STIP was detected. The descriptors



(a) Optical Flow (b) Gradient

Figure 4.1: Most significant optical flow and gradient and gradient vectors. Action is "handwaving" from KTH dataset.

are often based on gradients or optical flow, because these representations emphasize the parts in the video where changes occur. Figure 4.1 shows an example of optical flow and spatial gradients in a action video. The highest values are located on the actor, and the optical flow is concentrated around the arms which makes sense because the action is "handwaving".

A popular way of constructing motion feature descriptors, is to extend an existing spatial feature descriptor to account for the temporal domain as well. Histograms of Oriented Gradients(HOG) is one example of this. The 2D descriptor HOG is extended to 3D in two different ways: HOG/HOF and HOG3D.

4.1 Histograms of Oriented Gradients

Histograms of Oriented Gradients [5] is a popular 2D descriptor originally developed for person detection. The important components of the detector are shown in figure 4.2. A HOG descriptor is computed using a block consisting of a grid of cells where each cell again consists of a grid of pixels. The number of pixels in a cell and number of cells in a block can be varied. The structure performing best according to the original paper is 3×3 cells with 6×6 pixels.

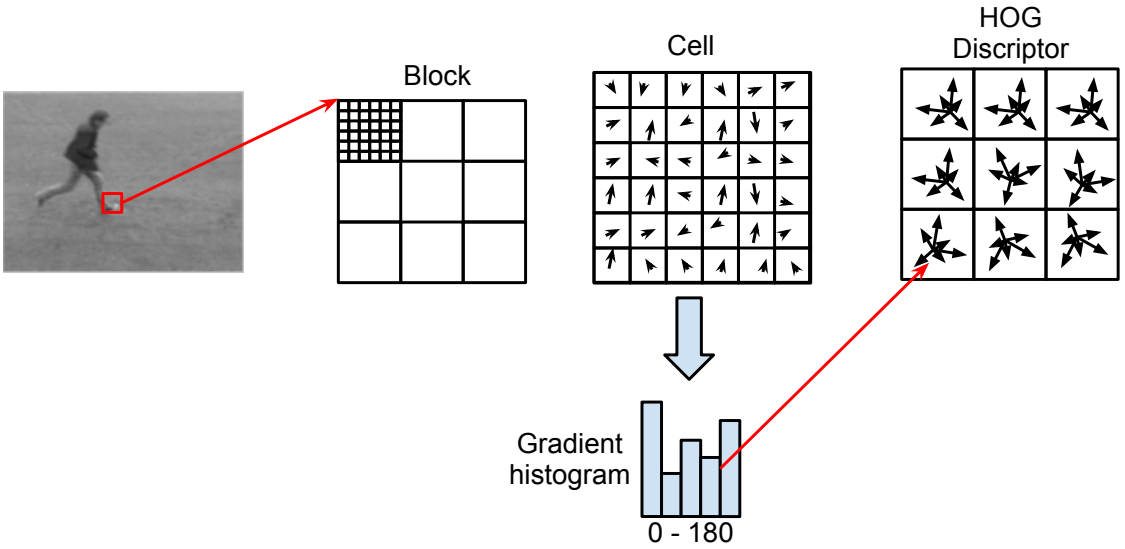


Figure 4.2: Block diagram of HOG method.

For each cell in the block, a histogram of the gradients in the pixels is computed. The histogram has 9 bins and a range of either 0-180°or 0-360°, where the former is known as unsigned and the latter as signed. Each gradient votes for the bin corresponding to the gradient direction, with a vote size corresponding to the gradient magnitude.

Finally, each block is concatenated into a vector v and normalized by its L_2 norm

$$v_{norm} = \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}} \tag{4.1}$$

where ϵ is a small constant to prevent division by zero.

The HOG descriptor is very similar to the descriptor used in SIFT [18]. The difference is that that the SIFT descriptor is rotated according to the orientation of the interest point.

4.2 HOG/HOF

HOG/HOF is the combination of histograms of oriented gradients with histograms of optical flow. HOF is computed the same way as HOG but with gradients replaced by optical flow. In [16], HOG is extended to include temporal information, by turning the 2D block in HOG into a 3D volume spanning (x, y, t). This volume is then divided into cuboids that corresponds to cells. The sizes of the volume is determined by the detection scale

$$\Delta_x, \Delta_y = 2k\sigma_l$$
$$\Delta_t = 2k\tau_l$$

making volumes larger for larger scales.

4.3 HOG3D

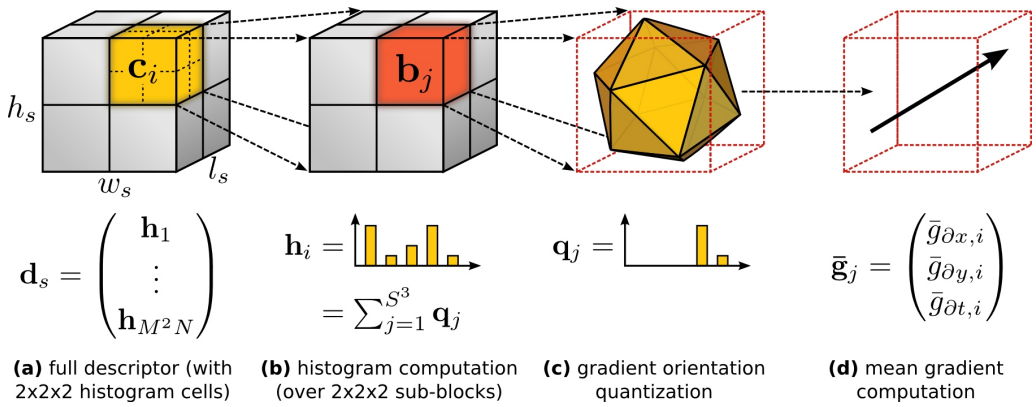


Figure 4.3: HOG3D Block diagram. [13]

The former approach of using HOG as a spatio-temporal feature involved computing gradients in cuboids instead of in cells, but the gradient directions are still two-dimensional. This makes them unable to capture the temporal information in the video, which is why the addition of HOF significantly improves the results.

HOG3D [13] changes the underlying approach by considering the three dimensional gradient instead of the 2D gradient. The gradient is computed in three dimensions, and histograms are quantized into polyhedrons. This elegant way of quantization is intuitive if we look at the standard way of quantizing 2D gradient orientations by approximation of a circle with a polygon. Each side of the polygon corresponds then to a histogram bin. By extending this to 3D the polygon becomes a polyhedron. The polyhedron used is an icosahedron which has 20 sides, thus resulting in a 20 bin histogram of 3d gradient directions. The final descriptor is obtained by concatenation of histograms into a vector, and normalization by the L_2 norm.

Chapter 5

Feature comparison

There are many different combinations of detector and descriptor that can be used in the framework, but which combination is the best? Wang et al. [35] tries to answer this question in a thorough comparison of detectors and descriptors. The detectors compared are: Harris3D, Cuboid, Hessian and Dense sampling. The descriptors used are HOG/HOF, HOG3D and ESURF. All different combinations of detector and descriptor are tested, to find the combination the yields the highest precision. The testing framework used, is the same as described in section 2.4, where the videos are represented using bag-of-features and classified using a non-linear support vector machine. Three different datasets are used: KTH, UCF sports and Hollywood2, to make sure that a results are not only achievable in a certain kind of videos.

5.1 Results

Table 5.1 shows the results for the KTH dataset. Harris3D is the best performing detector, and HOF is the best descriptor with the combination of the two gives the highest accuracy of 92.1%. The detectors outperform dense sampling in this dataset, and this could be explained by the simplicity of the dataset. The lack of camera movement and background clutter makes it less likely for the detectors to detect STIPs on the background, while the dense sampling will always include the background, which in KTH does not provide any valuable information about the action.

	HOG3D	HOG/HOF	HOG	HOF	Cuboids	ESURF
Harris3D	89.0%	**91.8%**	80.9%	**92.1%**	-	-
Cuboids	**90.0%**	88.7%	82.3%	88.2%	89.1%	-
Hessian	84.6%	88.7%	77.7%	88.6%	-	81.4%
Dense	85.3%	86.1%	79.0%	88.0%	-	-

Table 5.1: Average accuracy for various detector/descriptor combinations on the KTH dataset.

Table 5.2 shows the results for UCF Sports. In this more complex dataset, dense sampling performs better than all of the detectors. This can be explained the amounts of contextual information available in the background which is being captured by the dense sampling. In a sports dataset like UCF, the different actions will often have very similar background, e. g. stadium, pool or indoor sports arena. Capturing these backgrounds can provide better accuracy. The best performing descriptor is HOG3D and HOG/HOF the second-best.

	HOG3D	HOG/HOF	HOG	HOF	Cuboids	ESURF
Harris3D	79.7%	78.1%	71.4%	75.4%	-	-
Cuboids	**82.9%**	77.7%	72.7%	76.7%	76.6%	-
Hessian	79.0%	79.3%	66.0%	75.3%	-	77.3%
Dense	**85.6%**	81.6%	77.4%	**82.6%**	-	-

Table 5.2: Average accuracy for various detector/descriptor combinations on the UCF Sports dataset.

	HOG3D	HOG/HOF	HOG	HOF	Cuboids	ESURF
Harris3D	43.7%	45.2%	32.8%	43.3%	-	-
Cuboids	45.7%	**46.2%**	39.4%	42.9%	45.0%	-
Hessian	41.3%	**46.0%**	36.2%	43.0%	-	77.3%
Dense	45.3%	**47.4%**	39.4%	45.5	-	-

Table 5.3: Average accuracy for various detector/descriptor combinations on the Hollywood2 dataset.

The results for Hollywood2 is shown in table 5.3. This dataset is the most varied of the three, and this is reflected in the results. Dense is again performing better than the detectors, but this time the difference is not more than about 1% which could be explained by the fact that the intraclass variance in background and context is higher in Hollywood2 than in UCF Sports.

The best performing descriptor is HOG/HOF, and not HOG3D as in UCF Sports.It is hard to say why there is a difference between the performance of the descriptors across the different datasets, but some things can be concluded. Spatial Gradient information alone does not capture enough information as HOG is consistently outperformed by the other descriptors.

5.2 Conclusion

In the two realistic datasets tested in this survey, dense sampling outperforms the detectors. The detectors only perform slightly better in KTH because of the simple scene and background. Feature detectors still needs a lot of improvement to truly capture the important information in the video. Dense sampling does results in massive amounts of data, which can be time-consuming to process. Cuboids is generally the best performing of the detectors, but not significantly better than Harris3D.

The descriptor performing best is varying across datasets, but in general it seems to the descriptors containing temporal information that yields the highest accuracy. HOG3D includes spatial information by considering the gradient direction in both space and time and HOF includes optical flow which also encodes temporal changes in the video. HOG/HOF performs best in the most complicated dataset: Hollywood2, and generally performs good in the other datasets as well.

The combination selected for use in the experiments conducted in this thesis is Harris3D and HOG/HOF. This choice is made because it shows good results in all three datasets and although dense sampling gives a slightly better result in KTH and Hollywood, the more sparse representation of STIPs is preferred. The implementation used [16] also has the advantage of bundling both Harris3D and HOG/HOF.

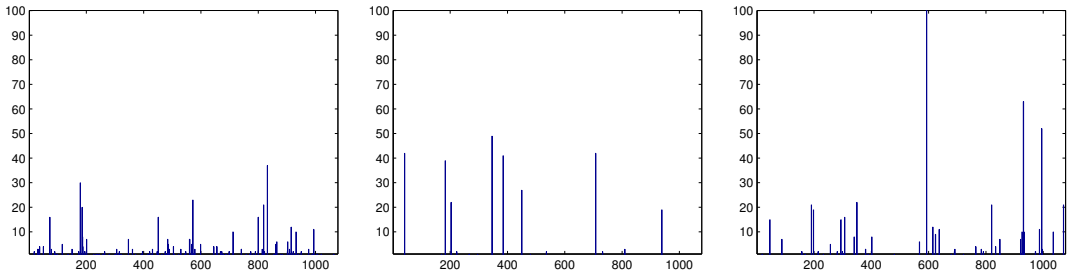
Chapter 6

Representation

This section describes the representation of videos in the framework. The features extracted from a video have to be represented in a way that serves well for classification. This generally results in a dimensionality reduction were the video often ends up being represented as a single vector, which is a useful input to a classifier.

6.1 Bag-of-features

Bag-of-features is derived from the bag of words representation that is a used in natural language processing to represent a text. The idea is that the text can be represented by the occurrences of words, disregarding the order in which they appear in the text, or to use the metaphor in the name, put all the words in a bag and draw them out without knowing the order in which they were put in. The result of the algorithm is a histogram of word occurrences, which can be more useful for comparing texts than a direct word-by-word comparison. The bins in the histogram are called the vocabulary, and can be the complete set of all kinds of words used in the text, or only a subset as it might be relevant to filter out common words like: the, be, to and of.



(a) Boxing 1 (b) Boxing 2 (c) Running

Figure 6.1: Three examples of word histograms from KTH with a vocabulary size of 1087 words.

The idea is extended to action recognition by Schuldt et al. [32]. Instead of a text, the histogram is now representing a video and instead of words consisting of letters, they are now features extracted from the videos. The vocabulary has to be defined and is most often generated by clustering a representative subset of all features in a dataset as seen in figure 6.2. There is no final answer to the number of words a vocabulary should consist of, but usually a more complex dataset requires a larger vocabulary. The bag-of-features histogram is computed by putting each feature into the bin of the most similar word, most often measured by euclidean distance. Finally the histogram

is normalized so that all feature vectors are comparable, even if there is a different number of features in the videos. Figure 6.1 shows 3 examples of word histograms, where two are of the same class. The histograms are not very meaningful to the human eye, and this lack of semantic meaning is one of the downsides to the bag-of-features representation.

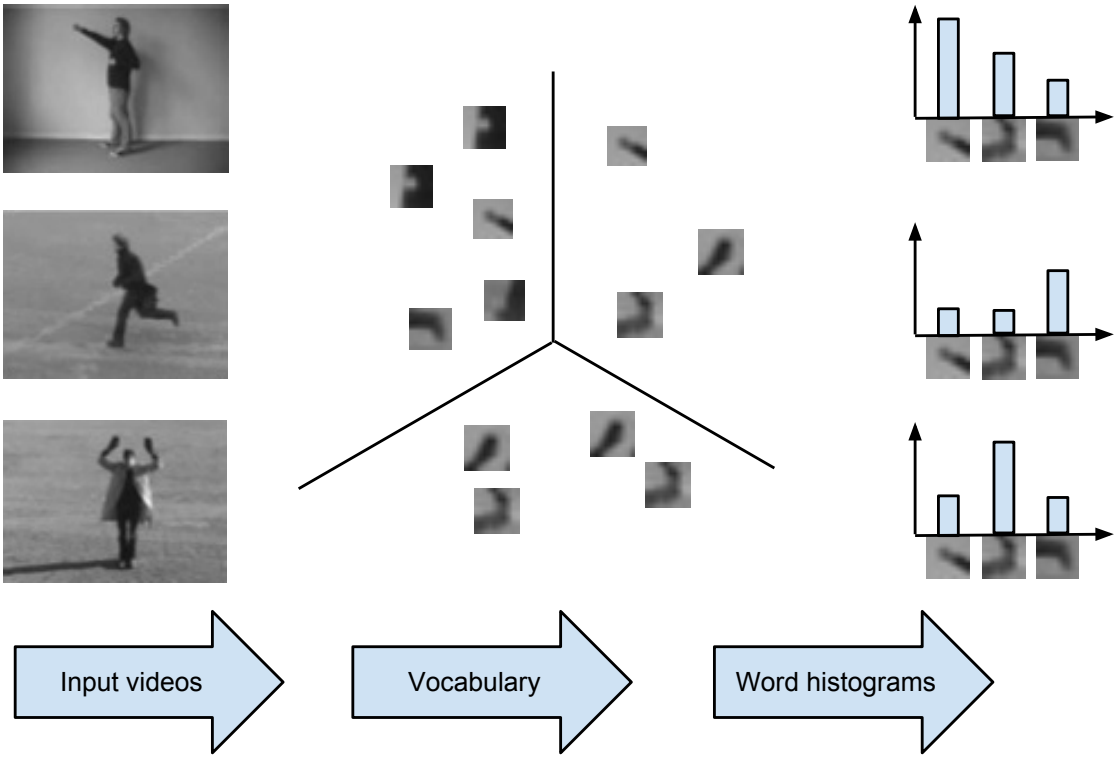


Figure 6.2: Bag-of-features representation.

Loosing the spatial and temporal information about the features is a strength and a weakness of bag-of-features. It makes it possible to classify datasets that have high intraclass variance and by that nature high differences in the spatio-temporal collocation of features. Throwing away the location information does, on the other hand, cause classification problems as confusion between similar classes.

There are ways to change the feature representation before bag-of-features to maintain the collocation information between the features. One example of this is a two step approach where features are grouped together based on their collocation to form new features. Yuan et al. [40] groups words together as phrases, where each phrase represents a combination of words. Chakraborty et al. [3] uses a pyramid division of features that divides them into groups based on their spatial location.

Chapter 7

Supervised Learning

Supervised learning uses training data to train a classifier that encodes the differences between the classes. The training data are supervised, i. e. the labels of the samples are known, so the classifier can learn the relationship between the features and the labels. Most of supervised classifiers requires significant amounts of training data, depending on the number of classes and the complexity of the dataset.

One challenge is not to overfit the classifier to the training data, resulting in that the classifier is only able to perform well on data very similar to the training data. In the opposite case, underfitting can cause the classifier to become too general, and the accuracy will suffer.

7.1 Support Vector Machine

A Support Vector Machine(SVM) is a linear binary classifier that seeks to maximize the distance between the points of two classes. The solution consists of a hyperplane that separates the two classes in the best way. It is possible to extend the SVM to achieve non-linear multi-class classification.

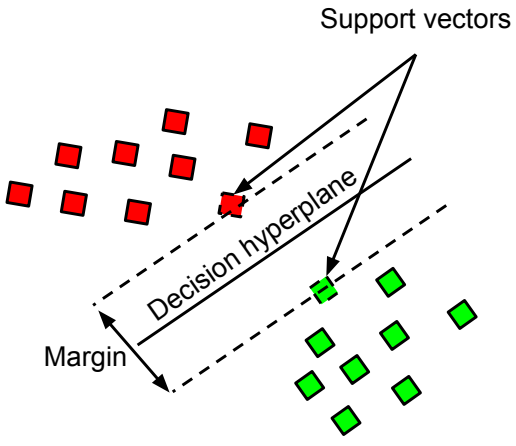


Figure 7.1: Support Vector Machine

7.1.1 Linear separable classes

In the simple case in fig 7.1, it is actually possible to obtain complete separation between two classes using a linear function. When the space is extended from two dimensions to n dimensions this linear function becomes a hyperplane. Many different hyperplanes could separate the points, but the desired hyperplane is the one in middle between the two point clouds, i. e. it maximizes the distance to the points. The points that influences the position of the hyperplane are the points closest to the empty space between the classes. These points are called support vectors.

The distance between the hyperplane and the support vectors is the margin, so the desired hyperplane is defined as the hyperplane that maximizes the margin.

We have L training points, where each input x_i is a D-dimensional feature vector, and is one of two classes $y_i \in \{-1, +1\}$. The hyperplane is defined as

$$w \cdot x + b = 0 \tag{7.1}$$

where w is the normal to the hyperplane. The objective of the SVM can then be described as finding w and b such that the two classes are separated

$$x_i \cdot w + b \geq +1 \qquad \text{for } y_i = +1 \tag{7.2}$$
$$x_i \cdot w + b \leq -1 \qquad \text{for } y_i = -1 \tag{7.3}$$

These equations can be combined into

$$y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall_i \tag{7.4}$$

The objective is to maximize the margin, which by simple vector geometry is found to be $\frac{1}{||w||}$. This means that the optimization problem can be formulated as a minimization of $||w||$ or rather

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||w||^2 \\ \text{subject to} \quad & y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall_i \end{aligned} \tag{7.5}$$

as this makes it possible to perform quadratic programming optimization later on. By introducing Lagrange multipliers α, the dual problem can eventually be reached

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\alpha^T H \alpha \\ \text{subject to} \quad & \alpha_i \geq 0 \; \forall_i, \\ & \sum_{i=1}^{L} \alpha_i y_i = 0 \end{aligned} \tag{7.6}$$

This problem is solved using a quadratic programming solver, yielding a solution for α and thereby w and b. Points can now be classified using the decision function

$$y' = sgn(w \cdot x' + b) \tag{7.7}$$

7.1.2 Non-separable points

In the formulation used above, it is assumed that the two classes are completely separable by a hyperplane, but this is often not the case. The points are often entangled in a way that makes it impossible to separate them, but it is still desired to have a classifier that makes as few mistakes as possible.

This is achieved by introducing a positive slack variable $\xi_i, i = 1 \ldots L$. This slack allows the points

to be located on the "wrong" side of the hyperplane as seen in the modified expressions

$$x_i \cdot w+b \geq +1 - \xi_i \qquad \text{for } \; y_i = +1 \tag{7.8}$$
$$x_i \cdot w+b \leq -1 + \xi_i \qquad \text{for } \; y_i = -1 \tag{7.9}$$
$$\xi_i \geq 0 \forall i \tag{7.10}$$

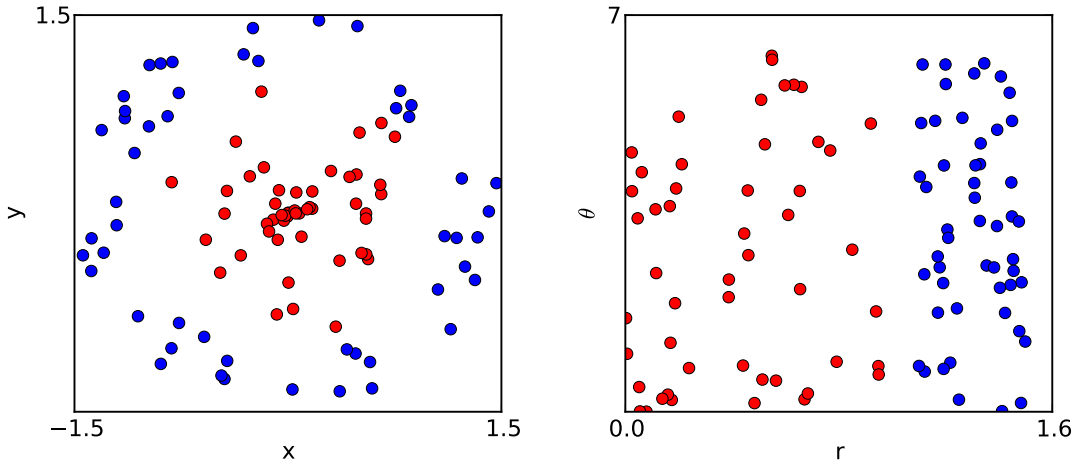
The minimization problem is reformulated as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{L}\xi_i \\ \text{subject to} \quad & y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0 \; \forall_i \end{aligned} \tag{7.11}$$

where the parameter C controls how much the slack variables are punished.

7.1.3 Extending to non-linear

In the examples above, a linear decision function was considered, but in many cases, the data are non-linear. A way to classify the non-linear data is to map the data onto another space where the data are separable. In the example shown in figure 7.2a, a linear classifier would not yield a good result, but by mapping the data to polar coordinates as shown in 7.2b, the data can be completely separated by a linear classifier.



(a) Cartesian coordinates (b) Polar coordinates

Figure 7.2: Data points shown in cartesian and polar coordinates.

More formally, this can be written as finding a mapping $x \rightarrow \phi(x)$, which for the example in figure 7.2 is given by

$$\phi : \Re^2 \rightarrow \Re^2 \tag{7.12}$$
$$(x, y) \rightarrow (r, \theta) := \left(\sqrt{x^2 + y^2}, \arctan\frac{y}{x}\right) \tag{7.13}$$

This can also be acheived in a simpler way, by doing the mapping implicitly in the optimization. This can be done if there exists a funtion k for which it holds that

$$k(x, y) = \phi(x) \cdot \phi(y) = \phi(x)^T\phi(y) \tag{7.14}$$

i. e. k is equal to the dot product of the vectors. K is called a kernel-function, and it enables the problem to be solved in complexity of the original space, while having the classification accuracy of the potentialy higher-order space.

Kernel	**Expression**
Linear	$k(x, y) = x^T y + c$
Polynomial	$k(x, y) = \left(ax^T y + c\right)^d$
Radial basis function(RBF)	$k(x, y) = \exp\left(-\frac{\|\|x-y\|\|^2}{2\sigma^2}\right)$
Intersection	$k(x, y) = \sum_{i=1}^{n} min(x_i, y_i)$
χ-squared	$k(x, y) = 1 - \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)}$

Table 7.1: Different examples of kernel functions.

Table 7.1 shows some popular choices of SVM kernels. The intersection and χ-squared kernels are especially useful when it comes to bag-of-features classification because they provide a good way to capture the difference between histograms which is the common way to represent a video in bag-of-features.

7.1.4 Multi-class approach

The above-mentioned SVM classifiers have all considered the binary case of labelling a video as either one class or the other, but this can of course be extended to the general case of n classes. Two common approaches are: one-vs-all and one-vs-one.

In one-vs-all, each classifier is trained using training data from one class, vs. the training data from all other classes resulting in n classifiers for n classes. In classification, a point is assigned to class that has the highest output of the descision function. The classifiers have to be trained in a way that results in the same range of their output to make the descision functions comparable.

In one-vs-one a classifier is constructed for each combination of two classes which results in $n(n-1)/2$ classifiers. In the classification step, each point casts a vote for one of the classes in each of the classifiers and in the end picks the class having the highest number of votes. In the event of a tie, the class being assigned is usually arbitrary.

The implementation used in this work, libSVM[4], uses one-vs-one for multi-class approach.

Chapter 8

Unsupervised Learning

In unsupervised Learning, there is no prior knowledge of labels or constraints between the data, and it is up to the algorithm to cluster the videos based on their features. This clustering can have a fixed goal of a number of clusters, or the number can be variable making it part of the problem to figure out how many clusters to choose.

One question is how to evaluate the results of an unsupervised algorithm. If the results has a cluster that consists of 4 videos, where 2 videos are of class a and 2 of class b, is this evaluated as 2 positive results for class a and 2 negative for class b, or the opposite? When there are no supervised data involved, a cluster does not have a prior label, and if the algorithm does not output a fixed number of clusters, there can be more than one cluster per label.

It could also be the case, that class a and b are actually similar in a way different from the problem formulation. If e. g. class a is running, class b is boxing, but all videos in the cluster takes place in a baseball stadium, the cluster could be representing baseball stadium and not running or boxing.

When the features used have no specific semantic meaning, the clusters found by an unsupervised method can not be guaranteed to have the semantic meaning defined by the prior class definitions.

8.1 K-means clustering

K-meas clustering is a clustering method that divides n points into k clusters, where each cluster belongs to a mean value of the cluster. Formally this is defined as minimizing the sum of squares within each cluster

$$\underset{S}{argmin} \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2 \tag{8.1}$$

where x_j is a point and μ_i is the mean belonging to cluster i. Solving this problem is NP-hard, but there are efficient heuristic algorithms to reach an approximation.

8.1.1 Standard algorithm

1. **Initialization**
   The algorithm is usually initialized by setting all the means to a random value, but there are also various heuristics that can choose starting means based on the data.
2. **Assignment step**
   In the assignment step, each point is assigned to the nearest mean based on the distance between the point and the means

$$S_i = \{x_p : ||x_p - \mu_i|| \leq ||x_p - \mu_j|| \, \forall \, j = 1 \dots k\} \tag{8.2}$$

This distance is usually the euclidean distance, but other distance measures can be used. When each point has been assigned to one mean, the result is k temporary clusters.

3. **Update step**
   A new mean is calculated in each cluster as the centroid of all the points in the cluster, i. e. the average of the point coordinates in a cluster.

$$\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j \tag{8.3}$$

4. Repeat step 2 and 3 until convergence.

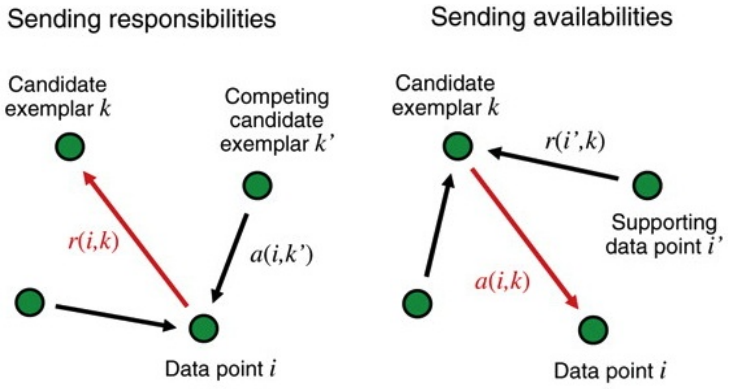
The algorithm is simple and efficient, but it also has a number of drawbacks. The solution found will almost always only be a local minimum. Picking the best out of multiple runs with different initializations is a way of improving the solution, but a guaranteed global minimum can of course not be found.

K-means has a tendency to find clusters that are similar in size. This does not make it suitable for data where there are significant differences in the sizes of the clusters.

Because that the distance measure used is the euclidean distance, the result is spherical clusters which is not very well suited for certain distributions.

8.2 Affinity Propagation

Affinity Propagation [10] is a message passing algorithm. This means that each point in the clustering exchanges messages with all other points to gain an understanding of the composition of the data. The result of affinity propagation is a number of exemplars, which are points selected from the input points. Each cluster has exactly one exemplar, which can be seen as the point the best represents the points in the cluster. Because there can be only one exemplar in a cluster, and every point has to choose an exemplar, each exemplar is its own exemplar. The number of



(a) Responsibilities (b) Availabilities

Figure 8.1: The two types of messages exchanged in affinity propagation. [10]

exemplars are not constrained to a fixed number like in e. g. k-means. Instead each point has a number called the preference $p(i)$. Points having a higher preference are more likely to become exemplars. The preference is usually equal for all points, and the author suggest setting it to the median of the similarities. When collective preferences determine the number of resulting clusters.

Exemplars are selected based on the messages which are passed between the points. The input to the algorithm are pairwise similarities between all the points $s(i, k)$. Any similarity measure

can be used, and similarities do not have to be symmetric: i. e. $s(i,k) = s(k,i)$ is always true, or satisfy the triangle equality: $s(i,k) < s(i,j) + s(j,k)$.

There are two types of messages that are being exchanged between the points: responsibility and availability, as shown in figure 8.1. Responsibility $r(i,k)$, sent from point i to candidate exemplar point k, is the evidence of how much i wants k to be it's exemplar. This evidence is based on how i views its other options of exemplars.

Availability $a(i,k)$, sent from candidate exemplar point k to i, is the evidence of how much k thinks that it should be i's exemplar. The availability from k is based on the responsibilities sent to k, i. e. if other points like k as an exemplar, k must be a good candidate exemplar and will then send a higher availability to i.

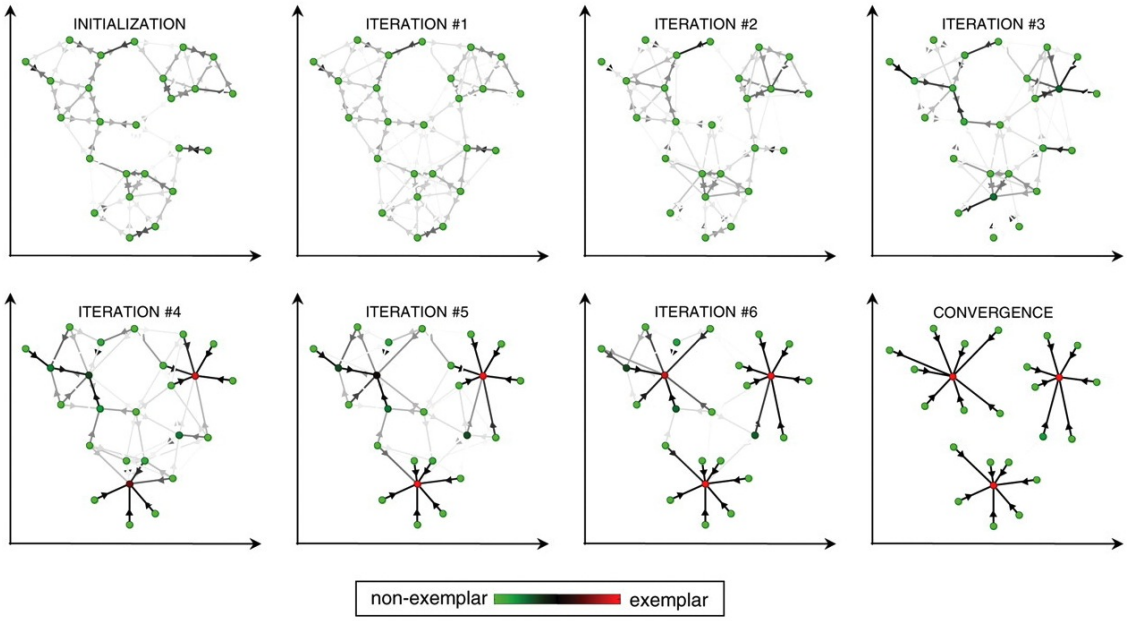


Figure 8.2: Visualization of iterations of 2D affinity propagation. [10]

The algorithm is initialized by setting all availabilities to zero: $a(i,k) = 0$. The responsibilities are then computed using the update rule

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\} \tag{8.4}$$

In the beginning when the availability is zero, r(i,k) will be equal to the similarity between i and k, minus the maximum of the similarities between i and all other points. This means that the responsibility is proportional to how similar i and k are to each other compared to other points, which is in accordance to the description of responsibility given above. The special case of $i = k$: r(k,k) is set to the input preference $p(k)$.

$$a(i,k) \leftarrow min \left\{ 0, r(k,k) + \sum_{i' \notin \{i,k\}} max \{0, r(i',k)\} \right\} \tag{8.5}$$

The expression for the availability is also intuitive. The availability of k, send to i, is determined by k's preference plus the sum of how well all other points besides i, like k as an exemplar.

The self-availability $a(k,k)$ is a measure of the accumulated evidence that k is an exemplar and is computed differently

$$a(k,k) \leftarrow \sum_{i' \neq k} max\left\{0, r(i',k)\right\} \tag{8.6}$$

For each iteration of computing availabilities and responsibilities, the temporary exemplars are determined by

$$\text{exemplar}(i) = \underset{k}{arg\,max}\ a(i,k) + r(i,k) \tag{8.7}$$

which means that if $k = i$, i is itself an exemplar, and else k is the exemplar of i. When the exemplar choices does not change for a number of iterations, convergence is reached and the algorithm terminates.

Figure 8.2 shows an illustration of the different iterations of affinity propagation. The messages starts out being almost equal between all points, but as the number of iterations increase the messages to some points increases and finally the algorithm converges with three exemplars.

Chapter 9

Results

This chapter presents the experimental results and comparison of the chosen subset of methods described in the previous sections. Section 9.1 describes the algorithms and parameters used in the tests. In section 9.2, the supervised results are presented for each of the datasets: KTH and UCF50. The results from the unsupervised algorithms are presented in section 9.3, and a comparison of supervised and unsupervised algorithms is given. Finally the parameters for bag-of-features vocabulary generation are evaluated.

9.1 Experimental Setup

The framework used in all experiments is the one described in section 2.4. The two first steps in the framework: feature detector and feature descriptor are fixed in all the experiments, while the last two: video representation and classification are changed, to compare the results.

The feature detector used is Harris3D which is combined with the HOG/HOF descriptor. The implementation used is the updated version of the one described in [16], obtained from the authors web page. The default settings are used for parameters like number of temporal and spatial scales and detector sensitivity. SVM is used for supervised classification and affinity propagation and k-means clustering are used for unsupervised classification.

For each of the datasets, a number of different vocabularies are built to find the best strategy for vocabulary generation. Each individual experiment utilizes the vocabulary that provides the best result for that specific experiment.

The measure used for comparison is mean average precision. Average precision is the average of all true positive percentages across classes. When cross validation is used, the test is run multiple times resulting in different average precisions. The mean of all these runs is reported as the result of the cross validation.

9.2 Supervised

This section presents the results of the supervised classifier. The classifier used is a support vector machine(SVM) and the implementation used is libsvm [4], with a χ-squared kernel. Because libsvm does not have native support for the χ-squared kernel, the kernel is precomputed using an external function. The default option of one-vs-one approach for multi-class classification is selected.

9.2.1 KTH

The 600 videos in KTH are partitioned according to the directions given in the dataset [32], with one exception: the video files are not split up into the sequences described in the directions, but kept together meaning that one video file contains the same subject performing an action multiple times. This does not seem to affect the recognition rate particularly, even when compared to Wang et al. [35], where the mirrored versions of the clips also are used to gain more data.

Figure 9.1 shows the confusion matrix for the result of the SVM classification of KTH. The average precision of 89.4% is close to the precision of 91.8% reported by [35], even though there are differences in methodology.

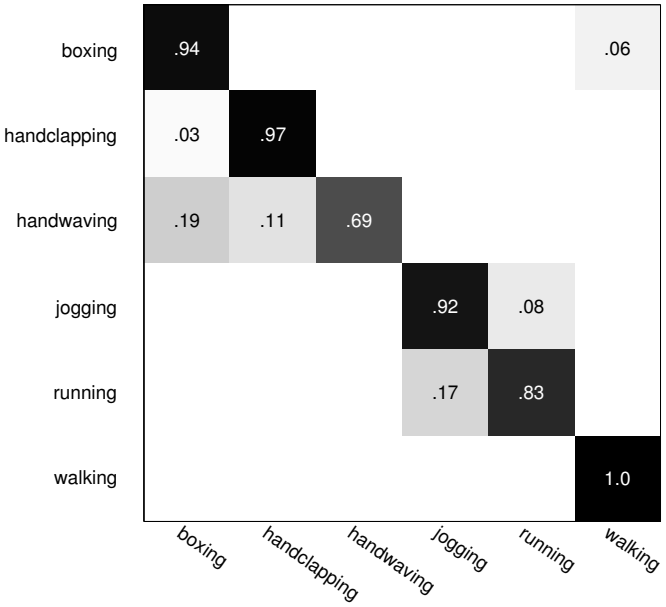


Figure 9.1: Confusion matrix of the best SVM classification of the KTH dataset. Avg. precision: 89.4%

There is some confusion among the arm classes and among the leg classes but not between the two. This is expected, as a video involving movement of the arms should be more likely to be confused with a class the involves arm movement. The highest error is found in handwaving, which is confused as boxing almost 20% of the time.

9.2.2 UCF50

UCF50 is classified using 10-fold cross-validation, i. e. the videos are divided into 10 groups and for each test run the classifier is trained using 1 group and tested on 9 remaining groups. The reported precision and confusion matrix is the average over the 10 runs.

Figure 9.2 shows the confusion matrix for the best SVM classification. There are large differences between the precision of the different classes, ranging from 98% in class 5: “Billiards” to 0% in class 44: “TaiChi”. Class 45: “TennisSwing” has many false positives from other classes.

9.3 Unsupervised

The unsupervised algorithms use the same word histograms for input as the SVM. K-means uses the histograms directly while affinity propagation uses their pairwise χ-squared distances. The

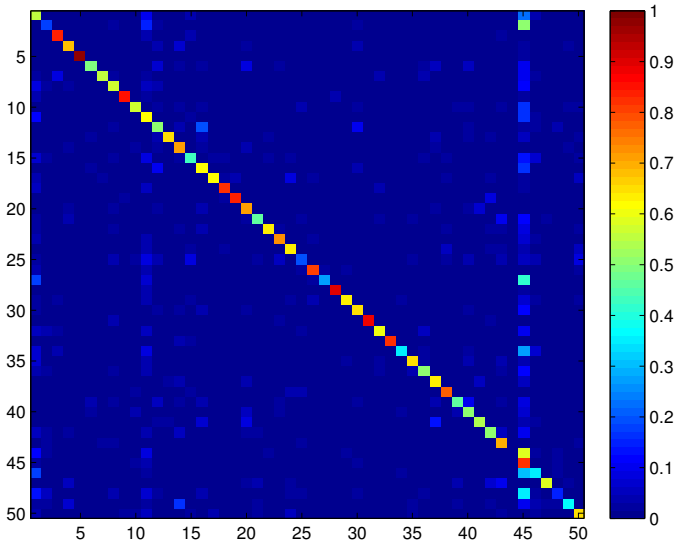


Figure 9.2: Confusion matrix of the best SVM classification of the UCF50 dataset. Avg. precision: 60.6%

outcome of affinity propagation is deterministic, but the k-means algorithm uses random starting means and will have a different solution each time. To partly overcome this difference, the best of 10 k-means runs is used.

The unsupervised algorithms are presented as a function of the number of clusters. A cluster is assigned the label of the most frequent occurring class in the cluster, which will make the precision go towards 100% with increasing number of clusters.

9.3.1 KTH

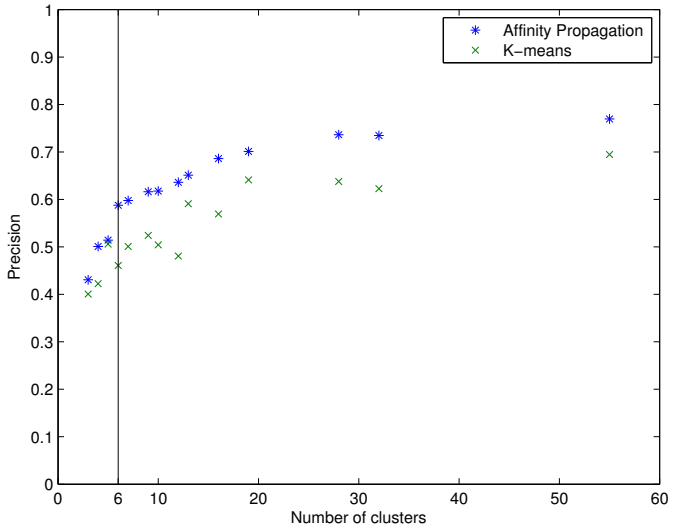


Figure 9.3: Comparison of Affinity propagation and K-means clustering for unsupervised clustering of KTH. The vertical line is located where the number of clusters is equal to the number of classes.

Figure 9.3 shows the results of the clustering. Affinity propagation is consistently better than k-means across different number of clusters. The recognition rate at 6 clusters is notable, because that is the number of classes in KTH. Affinity propagation obtains 58.8% while k-means only obtains 46.0%.

Both algorithms improve their precision as the number of clusters increases. The graph could be extended all the way to the number of videos in KTH, which is 600. At this point, the precision would be 1, because each video would be in a cluster by itself and by the prior definition a true positive.

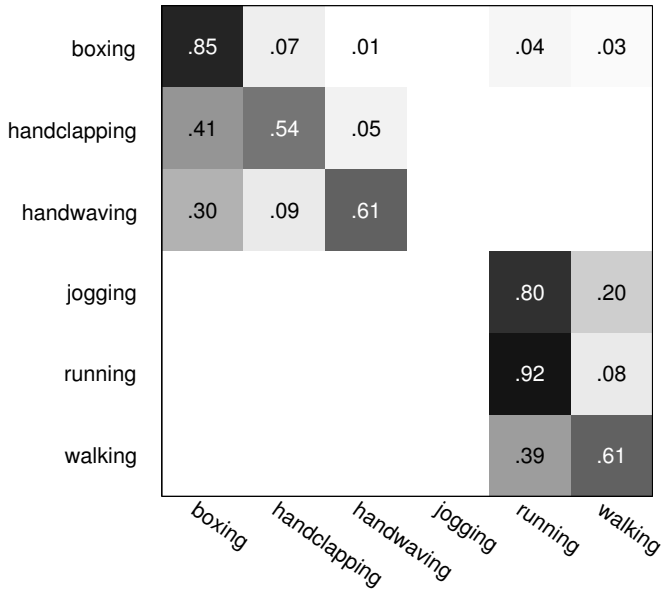


Figure 9.4: Confusion matrix of Affinity Propagation clustering of KTH. Average precision: 58.8%

Figure 9.4 shows the confusion matrix for the affinity propagation run that resulted in 6 clusters on KTH. One of the problems with the used cluster labelling method, is that every class does not have to be represented by a cluster, which is what happened in this case. There was no cluster in the result labelled as jogging because 80% of jogging videos ended up together in a cluster with 92% of the running videos. The cluster was then labelled as running, resulting in a large quantity of miss-labelled jogging samples.

Some classes are more greedy than others. Boxing not only includes 85% of its own points, but also 41% of handclapping and 30% of handwaving. The same is true for running, that also captures many of the videos from the two other leg actions. This could be due to the histograms for boxing and running videos being too general and not discriminative enough and thereby representing not only their own class, but also the others.

One reason why affinity propagation is performing better than k-means could be that the distance used for the similarity measure in affinity propagation is χ-squared distance, while k-means uses euclidean distance. According to experiments conducted in [3] with the type of SVM kernel used, the χ-squared kernel is the best performing kernel. This suggests that the χ-squared distance is a good measure for the distance between bag-of-features histograms.

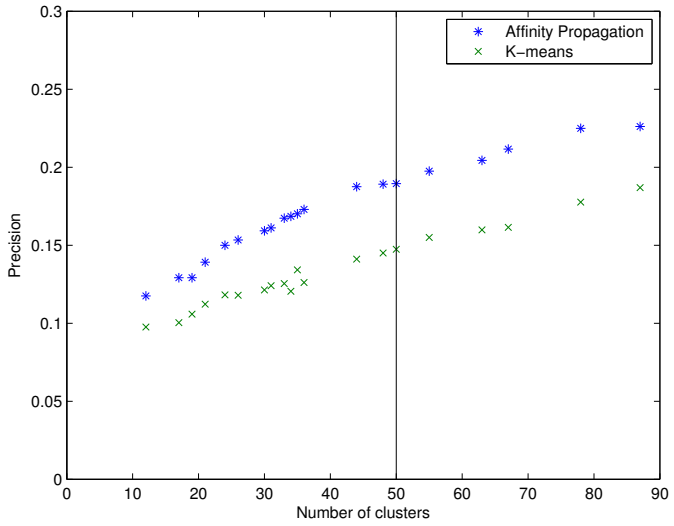


Figure 9.5: Comparison of affinity propagation and K-means clustering for unsupervised clustering of UCF50. The vertical line is located where the number of clusters is equal to the number of classes.

9.3.2 UCF50

The unsupervised results for UCF50 are seen in figure 9.5. Affinity propagation performs better than k-means over all number of clusters, and obtains a precision of 18.0% at 50 clusters. The gap between AP and k-means is smaller than it is for KTH.

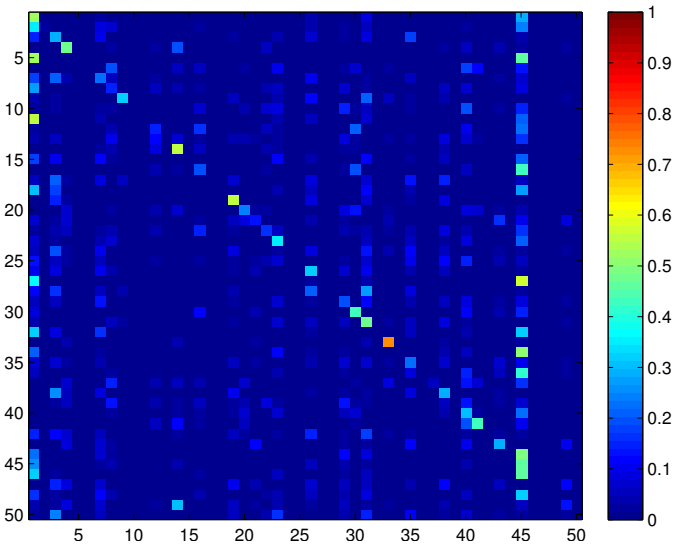


Figure 9.6: Confusion matrix of affinity propagation clustering of UCF50. Average precision: 18.9%

Figure 9.6 shows the confusion matrix for the best affinity propagation result. There is significant confusion, what mostly seems random, in almost every class apart from a few exceptions. Classes: 33: “Punch”, 14: “HorseRiding” and 19: “JumpingJack”, are the three classes having the highest

precision. Class 1:“BaseballPitch” and 45: “TennisSwing” are attracting many false positives from many other classes.

9.4 Comparison of supervised and unsupervised

This section compares the precision of the SVM classification with the precision of the unsupervised algorithms.

	SVM	AP	K-means
KTH	89.4%	58.8%	46.0%
UCF50	50%	18.9%	14.8%

Table 9.1: Comparison of average precision of supervised and unsupervised algorithms.

Table 9.1 shows the average precisions from supervised and unsupervised classification. The unsupervised precisions used, are from the clustering runs that has number of clusters equal to the number of classes, so that the results are comparable. SVM classification is over 30% more precise in KTH, and more than double as precise for UCF50 compared to affinity propagation.

9.5 Vocabulary Building Strategy

	Number of words		
	1078	2160	4856
AP 20.000	87.0%	85.6%	87.0%
KM 20.000	88.4%	82.4%	87.9%
KM 50.000	84.2%	85.6%	**89.4%**
KM 100.000	88.4%	85.6%	88.4%

Table 9.2: Comparison of different parameters for vocabulary building for KTH. AP is affinity propagation and KM is K-means.

The vocabulary is an important part of the bag-of-features framework. This makes it interesting to explore the different parameters used for vocabulary building to see which values that yields the best performance. The vocabulary is the result of clustering where the input is a sample of features from the dataset. Affinity propagation and k-means are compared to see if the used clustering algorithm makes any difference.

Three different vocabulary sizes are tested for each dataset. Affinity propagation does not output a fixed number of clusters, so by trying different p values, 3 different number of words were found for each dataset. The number of words output by AP is then used as the input k to k-means to get a comparable number of clusters. The number of input samples to the clustering algorithms used are: 20,000, 50,000 and 100,000. To get a representative sample from the set of all features, the same number of random features is sampled from each of the videos in the dataset to form the collective set used for clustering. This is only done for k-means because the implementation of AP has too high complexity in memory for running on large sample sizes, at least for the computational power available in this project.

The different vocabularies have been tested only using the supervised algorithm.

Table 9.2 shows the results for KTH. There are not a huge difference between the different strategies. K-means and AP perform similarly on 20.000 input features. There is a little gain in

	Number of words		
	591	2839	4327
AP 20,000	50.4%	55.7%	58.3%
KM 20,000	49.5%	56.4%	58.4%
KM 50,000	51.2%	54.4%	59.3
KM 100,000	51.1%	57.9%	**60.6%**

Table 9.3: Comparison of different parameters for vocabulary building for UCF50. AP is affinity propagation and KM is K-means.

performance by using more input points. K-means with 50.000 points performs better than k-means with 100.000 for the highest number of words, but this could be explained by the inherent random outcome of k-means.

Table 9.3 shows the results for UCF50. Here, the performance increases significantly with the number of words, which is not surprising as a diverse dataset is expected to require a diverse vocabulary to represent it. The performance also increases a little with the number of input samples, but not as much as expected.

Chapter 10

Conclusion

This thesis has explored the pros and cons of methods within the bag-of-features framework. The best methods within each component category were compared, and a few combinations selected for experimental tests. The Harris3D detector with HOG/HOF descriptor was used in a bag-of-features representation as a base for classification. Supervised and unsupervised classifiers were compared on the simple KTH dataset and the more complicated UCF50 dataset.

The results of the supervised SVM classification was an average precision of 89.4% in KTH. This was expected as this is simillar to what others achieved on KTH. This shows that KTH is getting to simple for comparison of state-of-the art supervised algorithms, as there is not enough room for future improvement. The mean average precision for SVM classification of UCF50 was 60.6% in UCF50, with many higly varying performance of the different classes.

The two unsupervised algorithms tested were: affinity propagation(AP) and k-means. Affinity propagation outperforms k-means for both KTH and UCF50, but the difference is more significant in KTH, which could mean that affinity propagation works better for small problems. The average precision of affinity propagation is 60% for KTH and 18.9% for UCF50.

One of the goals of this thesis was to compare supervised and unsupervised classification. SVM outperforms AP by about 30% in KTH and 20% in UCF50. The training data used in SVM provides a lot of information that the unsupervised methods do not have. UCF50 is especially a problem for the unsupervised algorithms. Even though the unsupervised algorithms do not achieve the best performance, they obtained their results without the use of training data which is a huge advantage. Further development of unsupervised methods will make classification less dependant of large amounts of training data. Semisupervised methods are also gaining attentention for their ability to combine the advantages of supervised and unsupervised algorithms.

Two different algorithms were tested for vocabulary building: affinity propagation and k-means. The clustering was done with a different number of words and a different number of input features to see which combination that yielded the best performance for supervised classification. The average precision of the SVM increases with the number of words in the vocabulary, having the best vocabulary at 4856 words in KTH and 4327 in UCF50. One explanation could be that SVM is good at dealing with many-dimensional points, and therefore benefits from having the extra information provided by a larger vocabulary. Affinity propagation does not seem to provide any advantage over k-means for vocabulary building as the two algorithms have almost the same precision in the comparable case of 20,000 input samples. Increasing the number of sampled features for clustering does not improve the vocabulary much above 50.000 features. The number of samples is more important for UCF50 than for KTH which can be expected because UCF50 is a much more diverse dataset.

Bibliography

[1] 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, 2005.

[2] A.F. Bobick and J.W. Davis. The recognition of human movement using temporal templates. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 23(3):257–267, 2001.

[3] B. Chakraborty, M.B. Holte, T.B. Moeslund, and J. Gonzàlez. Selective spatio-temporal interest points. Computer Vision and Image Understanding, 2011.

[4] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. Ieee, 2005.

[6] S. Danafar and N. Gheissari. Action recognition for surveillance applications using optic flow and svm. In Proceedings of the 8th Asian conference on Computer vision-Volume Part II, pages 457–466. Springer-Verlag, 2007.

[7] V. Delaitre, I. Laptev, and J. Sivic. Recognizing human actions in still images: a study of bag-of-features and part-based representations. IEEE PAMI, 31(10):1775–1789, 2009.

[8] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on, pages 65–72. IEEE, 2005.

[9] D. Dueck and B.J. Frey. Non-metric affinity propagation for unsupervised image categorization. In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1–8. IEEE, 2007.

[10] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. Science, 315:972–976, 2007. URL `www.psi.toronto.edu/affinitypropagation`.

[11] C. Harris and M. Stephens. A combined corner and edge detector. In Alvey vision conference, volume 15, page 50. Manchester, UK, 1988.

[12] G. Johansson. Visual perception of biological motion and a model for its analysis. Attention, Perception, & Psychophysics, 14(2):201–211, 1973.

[13] Alexander Kläser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In British Machine Vision Conference, pages 995–1004, 2008.

[14] I. Laptev. On space-time interest points. International Journal of Computer Vision, 64(2): 107–123, 2005.

[15] I. Laptev, B. Caputo, C. Schüldt, and T. Lindeberg. Local velocity-adapted motion events for spatio-temporal recognition. Computer Vision and Image Understanding, 108(3):207–229, 2007.

[16] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008.

[17] T. Lindeberg. Feature detection with automatic scale selection. International journal of computer vision, 30(2):79–116, 1998.

[18] D.G. Lowe. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2):91–110, 2004.

[19] D. Marr and H.K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. Proceedings of the Royal Society of London. Series B. Biological Sciences, 200(1140):269–294, 1978.

[20] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 2929–2936. IEEE, 2009.

[21] Microsoft. Kinect for xbox 360, 2012. URL `http://www.xbox.com/da-DK/Kinect`.

[22] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. Computer Vision and Image Understanding, 81(3):231 – 268, 2001.

[23] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. Computer Vision and Image Understanding, 104 (2–3):90–126, 2006.

[24] J.C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. International Journal of Computer Vision, 79(3):299–318, 2008.

[25] University of Central Florida. Ucf50 dataset. URL `http://vision.eecs.ucf.edu/datasetsActions.html`.

[26] University of Minnesota. Unusual crowd activity dataset. URL `http://mha.cs.umn.edu/proj_events.shtml`.

[27] R. Poppe. A survey on vision-based human action recognition. Image and Vision Computing, 28(6):976–990, 2010.

[28] D.K. Ramanan and D. Forsyth. Automatic annotation of everyday movements. Computer Science Division, University of California, 2003.

[29] K. Rapantzikos, Y. Avrithis, and S. Kollias. Spatiotemporal saliency for event detection and representation in the 3d wavelet domain: potential in human action recognition. In Proceedings of the 6th ACM international conference on Image and video retrieval, pages 294–301. ACM, 2007.

[30] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. 2005.

[31] S. Savarese, A. DelPozo, J.C. Niebles, and L. Fei-Fei. Spatial-temporal correlatons for unsupervised action classification. In Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on, pages 1–8. IEEE, 2008.

[32] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 3, pages 32–36. IEEE, 2004.

[33] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In Proceedings of the 15th international conference on Multimedia, pages 357–360. ACM, 2007.

[34] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing gaussian mixture models with em using equivalence constraints. Advances in neural information processing systems, 16:465–472, 2004.

[35] H. Wang, M.M. Ullah, A. Klaser, I. Laptev, C. Schmid, et al. Evaluation of local spatio-temporal features for action recognition. BMVC, 2009.

[36] S. Wang, Z. Liu, S. Lv, Y. Lv, G. Wu, P. Peng, F. Chen, and X. Wang. A natural visible and infrared facial expression database for expression recognition and emotion inference. Multimedia, IEEE Transactions on, 12(7):682–691, 2010.

[37] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. Computer Vision and Image Understanding, 115 (2):224–241, 2011.

[38] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. Computer Vision–ECCV 2008, pages 650–663, 2008.

[39] Youtube.com. Youtube statistics, 2012. URL `http://www.youtube.com/t/press_statistics`.

[40] J. Yuan, Y. Wu, and M. Yang. Discovery of collocation patterns: from visual words to visual phrases. In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pages 1–8. Ieee, 2007.

[41] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.