Video Classification by Frequency Features based on Repeating Movements

Inaugural-Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von
Kahraman Ayyildiz
aus Bocholt

Düsseldorf, August 2015

Aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent: Prof. Dr. Stefan Conrad
Korreferent: Prof. Dr. Stefan Harmeling

Tag der mündlichen Prüfung: 11.12.2015

Eidesstattliche Erklärung

Ich versichere an Eides Statt, dass die Dissertation von mir selbständig und ohne unzulässige fremde Hilfe unter Beachtung der "Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Heinrich-Heine-Universität Düsseldorf" erstellt worden ist.

In .... am .... Kahraman Ayyildiz

Acknowledgments

Writing a PhD is like a long journey and at some point there has to be someone leading the way. Hereby I would like to express my special appreciation and thanks to my doctoral supervisor Prof. Dr. Stefan Conrad. You have been a tremendous mentor for me. Especially at the beginning of my PhD thesis you led my research into the right direction. You encouraged my creativity and gave me the space to grow as a scientist. It was a fruitful collaboration over the past years.
Moreover, I would like to thank my wife and my family. They offered moral support and motivated me to proceed in every phase of my PhD. A harmonic family life is the optimal balance for hard work.
Studying at the Heinrich Heine University was a pleasure for me.

Summary

This thesis at hand deals with a novel, content based video classification approach. Basically frequency features of repeating movements are utilized for the classification process. These frequency features play a central role for the whole classification system. But there are further parts of the system, which are analyzed in the context of repeating movements. For instance transforms, filters, classifiers and background subtraction models are explored in detail. Further, the robustness of the approach against different camera and environment settings is exposed.

We can divide this thesis into five main parts. Chapters 1 and 2 represent the first part. This part introduces into the topic and conveys basic techniques, that are used throughout the thesis. Part two consists of chapters 3 and 4. Here the extraction and utilization of frequency features from videos with repeating movements is explained. We measure the accuracy of the system based on these frequency features and discuss results. Chapter 5 represents the third part and shows how one-dimensional and two-dimensional repeating motion trajectories can be transformed. We underline experimentally that transforming two one-dimensional trajectories is more efficient than transforming one two-dimensional trajectory. Chapter 6 is considered as part four. Here the robustness of the presented approach to different settings is illustrated. Spatial camera rotation, scaling, translation, occlusion and time shift is analyzed. Chapters 7, 8, 9 and 10 represent part five of this thesis. Each of these chapters focuses on one stage or module of the system. Chapter 7 is about filters applied to one-dimensional input data. Filtering input data is a preprocess step, that aims to optimize the transformation result and therefore the clarity of the frequency domain. Chapter 8 deals with transforms applied to discrete, one-dimensional trajectory data. In chapter 9 different classifiers are studied, which assign videos by their frequency features to classes. In chapter 10 the focus is on background subtraction models, that are robust to camera motion.

The presented research on different modules of the classification system depends strongly on the extracted and assembled features by repeating motions.

Zusammenfassung

Die vorliegende Dissertation befasst sich mit einem neuen, inhaltsbasierten Konzept zur Video-Klassifikation. Es werden Frequenzeigenschaften wiederkehrender Bewegungen im Video zur Klassifikation herangezogen. Diese Frequenzeigenschaften spielen eine zentrale Rolle für das gesamte Klassifikationssystem. Jedoch gibt es eine Reihe weiterer Aspekte des Systems, welche im Kontext wiederkehrender Bewegungen und Frequenzen untersucht werden. Zum Beispiel werden Transformationen, Filter, Klassifikatoren und Hintergrund-Subtraktionsmodelle tiefgehend analysiert. Zudem stellen wir die Robustheit des Ansatzes gegenüber verschiedenen Kamera- und Umgebungseinstellungen heraus.
Die Dissertation kann in fünf Teile untergliedert werden. Kapitel 1 und 2 stellen den ersten Teil der Arbeit dar. Hier wird in das Themengebiet eingeführt und es werden grundsätzliche Techniken vermittelt, welche innerhalb der gesamten Arbeit immer wieder aufgegriffen und verwendet werden. Der zweite Teil besteht aus den Kapiteln 3 und 4. In diesem Teil der Arbeit wird erklärt, wie Frequenzeigenschaften aus Videos mit wiederkehrenden Bewegungen extrahiert und verwendet werden können. Basierend auf diesen Frequenzeigenschaften wird die Performanz des Systems gemessen und Ergebnisse werden diskutiert. Kapitel 5 repräsentiert den dritten Teil und zeigt auf, wie eindimensionale und zweidimensionale Verlaufsbahnen wiederkehrender Bewegungen transformiert werden. Es wird experimentell belegt, dass die Transformation von zwei eindimensionalen Verlaufsbahnen zu klareren Ergebnissen führt als die Transformation einer zweidimensionalen Verlaufsbahn. Kapitel 6 wird als vierter Teil der Arbeit angesehen und analysiert die Robustheit des Verfahrens gegenüber unterschiedlichen Randbedingungen. Es werden räumliche Kamerarotation, Skalierung, Translation, Okklusion und die zeitliche Verschiebung einer Aktion analysiert. Kapitel 7, 8, 9 und 10 stellen den fünften Teil der Dissertation dar. Jedes einzelne dieser Kapitel befasst sich mit einer Phase beziehungsweise einem Modul des Gesamtsystems. In Kapitel 7 werden verschiedene Filter auf eine Folge von eindimensionalen Eingabewerten angewandt, um so die Transformation in den Frequenzbereich zu optimieren. In Kapitel 8 werden Transformationen von diskreten, eindimensionalen Eingabewerten analysiert. In Kapitel 9 werden Klassifikatoren im Hinblick auf ihre Güte und Laufzeit untersucht. Als Klassifikationsbasis dienen dabei die Frequenzeigenschaften der Bewegungen im Video. Kapitel 10 legt den Fokus auf Hintergrund-Subtraktionsmodelle, die robust gegenüber Kamerabewegungen sind.
Die vorgelegten Forschungsergebnisse zu den verschiedenen Modulen des Klassifikationssystems hängen stark von den Frequenzeigenschaften ab, welche aus den wiederkehrenden Bewegungen extrahiert und zusammengestellt werden.

The author has enjoyed fruitful collaboration with his doctorate supervisor Prof. Dr. Stefan Conrad. A number of peer-reviewed papers, which constitute this thesis, are a result of this collaboration:

- **Chapter 3:** K. Ayyildiz and S. Conrad. Video classification by main frequencies of repeating movements. *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, 2011.
- **Chapter 4:** K. Ayyildiz and S. Conrad. Video classification by partitioned frequency spectra of repeating movements. *World Academy of Science, Engineering and Technology (WASET)*, pages 154–159, 2012.
- **Chapter 6:** K. Ayyildiz and S. Conrad. Analyzing invariance of frequency domain based features from videos with repeating motion. *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 659–666, 2012.
- **Chapter 7:** K. Ayyildiz and S. Conrad. Applying filters to repeating motion based trajectories for video classification. *International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 25-32, 2015.
- **Chapter 8:** K. Ayyildiz and S. Conrad. Analyzing transformation of 1d-functions for frequency domain based video classification. *World Academy of Science, Engineering and Technology (WASET)*, pages 921 – 927, 2012.
- **Chapter 9:** K. Ayyildiz and S. Conrad. Classifier comparison for repeating motion based video classification. *International Symposium on Visual Computing (ISVC)*, pages 725–736, 2013.
- **Chapter 10:** K. Ayyildiz and S. Conrad. Optimized background subtraction for moving camera recordings. *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2015.

The author wrote the papers for the most part independently. This includes developing ideas, experimental setups, structuring the work and formulating the texts, algorithms and equations. The doctorate supervisor Prof. Stefan Conrad was mainly responsible for the revision part. So texts, algorithms and equations were corrected by him. Moreover, he revealed mistakes concerning the structure and the content of the work. He also supported the author by additional ideas, aspects to consider and the choice of conferences.

Contents

Chapter 1

Introduction

This thesis is about content based video classification and action recognition. The presented research contributes novel and efficient approaches to this subject area. In this first chapter the motivation, purpose, and state of the art are outlined. Furthermore, the subject area is classified and related research fields are shown up.

1.1 Motivation

Video classification and action recognition plays a central role in computer vision, since video surveillance systems, human-computer interfaces or motion recognition software have a growing interest. Besides the World Wide Web with online video portals [68] and online video stores [61], there are many other branches and institutions with large video archives. Some examples are museums, the media branch, major corporations, or governmental institutions. All these areas need efficient algorithms for classifying the amount of existing clips or videos automatically.

Provider sided classification ensures the quality of the indexing process, but has high costs. On the other hand user sided classification has low costs without ensuring indexing quality. Both approaches usually annotate videos with general descriptions, which capture the main content. Beyond this main content videos contain activities, which are not captured by common annotations. An automatic indexing and annotating process can reduce costs, capture specific information and ensure quality. A bulk of work with regard to automatic video classification exists in literature. Nevertheless, video classification by frequencies of repeating movements is sparsely documented.

Repeating movements are part of many activities. For instance sports, playing musical instruments or home improvement are some areas with a high occurrence of repeating movements. These repeating movements can be considered as features of an activity, which can be utilized during an automatic classification process. This thesis now evaluates these features and contributes novel solutions to some subject areas of computer vision, which are depicted in the following section.

1.2 Research Area

Computer vision and information retrieval are the superior research areas of this thesis. Computer vision provides a series of techniques for action recognition. Whereas information retrieval uses these techniques in order to index and retrieve documents containing specific actions. Computer vision as part of computer science is concerned with image and video formation, preprocessing, analysis, understanding, and applications. Furthermore, motion analysis, tracking, and stereo vision are core research areas.
All of these areas play a role for the present thesis. Before the content analysis of a video its format and the format of its single frames have to be adapted. The content analysis phase applies segmentation and color analysis. This phase again depends on the motion detected for each frame. Motion and object tracking provides further information, that can be used for video understanding. For the present approach video understanding means primarily action recognition and content based indexing. Moreover, it means object detection and object localization.
Information retrieval is a subject of computer science, computational linguistics, and information science. It provides several retrieval models for retrieving documents in large databases. Originally retrieval models were applied to text document databases. But later on further document types were added. Some examples are digital photos, vector graphics, XML documents, HTML documents, GIF animations, MPEG-coded films, and MPEG-coded audio files. Information retrieval can be realized in different ways. One possible concept is textual annotation of documents without regard to its document type. In this case text retrieval supplies all relevant data. Another concept is content based retrieval. In this case all similar documents to an input document are retrieved. Further, a classification of documents allows users to pick out specific documents.
The present research work deals especially with action recognition, automatic video classification, and video retrieval. For video classification and video retrieval there are different video features, that can be utilized. The most popular method for classifying videos by content is to extract and match features from single video frames. Features can base on color, shape, texture and edge information. Also objects and their positions to each other can be utilized. A further method extracts text information from video frames. Many videos show up titles or names at the beginning, which are directly connected to the video content. Some texts are displayed at remarkable points and some throughout the whole video. The usage of optical character recognition (OCR) allows to extract these texts and to annotate videos. Besides key frames and text in frames, videos supply audio data. Audio data can represent sounds or speech. Especially speech is very interesting in the context of video retrieval, since speech can be converted to text, which again can be used for classification and annotation. Sounds can reveal the environment or topic of a video scene. The present research work has a focus on motion in videos. Motion is the fourth main source for features beside frames, texts and sounds. It is possible to classify an activity just by its motion pattern. A sprinter, for instance, has a different motion pattern than a long jumper, if we consider motion patterns from sports. With respect to periodical or repeating motion the human gait recognition is best investigated in literature. There

is a plenty of human activities containing repeating movements besides walking and running. Moreover, mechanical motions often repeat. Some examples are oil pumps, windmills, pendulum clocks or escalators. Repeating movements can also be found in other areas. The frequency of repeating movements is considered as a key feature in this research study. Next section 1.3 presents related work covering the main research area.

1.3 State of the Art

This section gives an overview of the research area by selected papers and introduces to the state of the art. Here general classification approaches are explained, that are strongly related to the approach presented in this work. More specific related work is outlined in the regarding chapters.
Videos reveal a huge amount of information. Hence, video annotation and classification can be realized in many different ways. A key-frame based approach, for example, is introduced by Pei and Chan in [70]. After detecting scene changes in videos key-frames are figured out for each scene. Key-frames lead to feature vectors for each frame and scene. Another approach is presented by Lienhart in [56]. Here videos are retrieved by texts within video shots. In [69] Patel and Sethi describe a method, which is able to detect dialog scenes in films by analyzing the audio signal.
The most research work related to motion recognition focuses on the gait [32], gestures [53] or face expressions [15] of humans. An approach for recognizing human motions in general is proposed by [12]. The authors utilize the flow and the strength of change of pixels as features. Some research concentrates on periodical motion in videos. In [85] the periodical motion of human body parts is captured by Moving Light Displays (MLD). Each MLD describes two discrete functions depending on time t: One function represents the x-coordinates and the other one the y-coordinates. Both functions are combined by an additional smoothing function and transformed to the frequency domain by fast Fourier transform. Then pieces of curves described by these MLDs, which fit best to the maximal frequency, are saved. These curve patterns serve as video features during classification phase. Davis and Cutler provide a method, which works with a single frequency [30]. They obtain the frequency domain by transforming measured self-similarity of motion as it evolves in time. Here one main frequency represents the video feature. Another method related to periodical motion recognition is proposed by He and Debrunner [44]. By calculating so-called *Hu moments* [48] for regions with motion in each frame, they count the number of frames until a Hu moment repeats and define this number as frequency. Afterwards, detected frequencies are used for classification by Hidden Markov Models (HMM). In image processing Hu moments are image moments with special properties. They are translation, scale and rotation invariant. A widely cited work on periodical movement recognition originates from Polana and Nelson [71]. They divide each frame of a clip into 16 parts of same size and store 6 frames for each repeating motion. Then pixel activities for these 6×16 parts are measured and summed up for every cyclic motion. The resulting 96-dimensional feature vector is used for classification. In [73] the authors explain a video classification system, which is able to subtract camera motion from a scene and to determine the video class by

the rate of foreground motion change. The rate of motion change is considered as a signal depending on time t.
The research work at hand focuses on repeating motion, which is also discussed in a similar way by [62] and [25]. [62] deals with repeating motion of human body parts tracked also by Moving Light Displays (MLD). Frequency peaks of Fourier transformed MLD curves are considered as features of repeating motion. In [25] Cheng et al. assign sports videos by using a neural network based classifier. They receive two main frequencies for each video by transforming series of vertical and horizontal pixel motion vectors. The transformation takes place by a modified fast Fourier transform. Cheng et al. define a set of main frequencies as a template for each sport or class. These templates again are used for assigning new videos. The authors assume that frequencies inside templates are Gaussian distributed. So a new video without any class information is assigned to a class, if its frequency features lie inside the standard deviation. A further repeating movements based approach is explained by Cutler and Turk [31]. Their system is able to recognize gestures in real time. First, the system detects motion areas by segmenting optical flow. Then characteristics of these motion areas as number, size, direction of movement and periodicity of movement are used for classification. If a motion area reveals repeating movement, its frequency is determined by the FFT of a series of motion area centroids.

1.4 Aim and Purpose

The previous section 1.3 introduced to relevant research papers concerned with motion frequency based action recognition. All of these papers provide valuable ideas, but at the same time there are disadvantages that have to be discussed. This section first reveals deficiencies of the introduced approaches and papers in order to underline the necessity of a novel and more complete concept. Then the steps for the study, that leads to a novel concept, are explained.
Polana and Nelson focus on patterns of repeating motion in specific frame areas and use these patterns to build up a feature vector [71]. The detected motion patterns are sensitive to translation, rotation and scaling. This means a feature comparison between videos requires exactly the same recording conditions for each video. The object's position and distance to the camera have to be the same. Moreover, the frequency of a motion is not part of the feature vector. In the end Polana and Nelson compare image patterns produced by repeating motion, but not the frequency of the repetition. This image pattern based approach is highly dependent on specific recording conditions and therefore hardly applicable to real world databases. He and Debrunner's approach has two disadvantages [44]. First, the way the authors use Hu moments leads to only one main frequency. Thus, the repeating motion yields only the minimal frequency information. Second, movements relative to the environment cannot be analyzed by this approach. For instance, the repeating change of direction of a table tennis ball gives always the frequency zero, because the shape of the ball does not change. The approach of Tsai et al. works with continuous motion in one direction [85]. As soon as the tracked person changes his direction, the MLD curve becomes a two dimensional drawing. Additionally, just as Meng et al. in [62] Tsai et al. use MLDs in their work for motion tracking. This means the presented approach is not applicable

to standard real world databases. Furthermore, the authors combine x- and y-axis motion data before transformation. This combination can affect the clarity of the frequency domain. Beyond this the usage of motion patterns is a weak method, since motion patterns of the same action can vary strongly. Cheng, Christmas and Kittler present the most convincing method [25]. They identify the frequency of a foreground motion by analyzing its centroid trajectory. As He and Debrunner the authors here again utilize only one main frequency, although repeating motion in general reveals several frequency peaks. In addition, Cheng et al. provide little experimental results realized by only 5 classes containing 1 to 5 videos. Cutler and Turk's gesture recognition system uses the frequency feature as one feature among multiple features [31]. The authors do not give a profound insight into the frequency feature analysis and application.
So concluding it can be stated that state of the art approaches need further analysis and improvement. Now the aim of the present research work is to analyze frequency features of repeating movements for video classification with respect to the benefits and disadvantages of the introduced related research. It is intended to find an optimal technique for frequency based video classification by extended theoretical and experimental analysis. The thesis has a focus on different stages of the classification process: One central issue is the construction of a feature vector based on frequency domains. Here different options are possible, that have to be evaluated. Furthermore, different image moments are available, which constitute the motion function for transformation. Image moments can influence the robustness of a classification system to different video properties. Moreover, the presented system has a modular structure, where each module can be replaced by a module with a different implementation. For instance, the transform module can apply the fast Fourier transform or the fast sine transform. The same principle holds for the filter module, classifier module and background subtraction module. So each of these modules needs profound analysis in order to find the most efficient ones. Efficiency means primarily high classification accuracy and low computation cost. The classification accuracy and the computation cost are not necessarily dependent on each other. It is possible that a module works highly accurate, needs little resources and has a short runtime at once. In cases, where the most accurate module and the fastest module differ, the most efficient module is a tradeoff between the two aspects.

Chapter 2

Application Overview and Basics

This chapter gives an overview of the presented and analyzed video classification system. Furthermore, some basic techniques are introduced, that play a role throughout the whole thesis. This includes techniques for motion area detection, image moment calculation and deriving motion functions. Moreover, a novel classifier is presented, that has turned out as very accurate during the experimental analysis. Main definitions of this chapter we also published in [5] and [8].

2.1 Overview of the Video Classification System

The flow diagram in figure 2.1 shows the different stages of the presented system. Its overall goal is to classify video sequences with repeating movements properly. Some examples for activity with repeating movements are jumping, playing tennis or hammering. The presented approach is not capable of classifying activities with *strongly* differing frequency spectra efficiently. For instance, riding a bicycle slowly or fast gives completely different pedaling frequencies. So the magnitude peaks of the frequency domain are shifted to the left or to the right depending on the riding speed. A large reference database, which encloses a wide range of frequencies, can improve the classification result. Also classifying periodic texture motion is difficult with our approach, since determining a main movement is often not feasible.

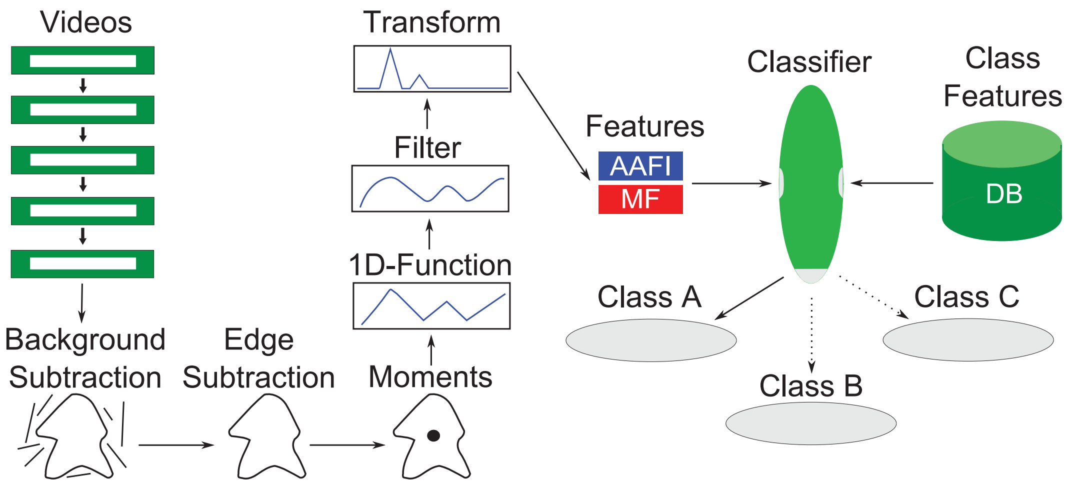

Figure 2.1: Flow diagram for the video classification system

As a first step the flow diagram shows background subtraction for each frame of an input video (see section 10.3). Background subtraction means foreground detection at the same time and represents a technique, which subtracts a background model from a frame in order to find the foreground object or foreground motion. In the case of input videos containing camera motion edge subtraction is applied (see section 10.4). Edge subtraction is needed, because camera motion leads to edges in the detected foreground mask, which belong to the background. As a next step image moments are calculated on the basis of the foreground mask (see section 2.2.2). Different types of image moments as raw moments, central moments, scale invariant moments and rotation invariant moments do exist. Image properties like the area, centroid, orientation or pixel variances can be derived by image moments. The chronological order of image moments or image properties derived by image moments is referenced as a *1D-function*, which again represents the main motion in a video sequence (see section 2.2.3). Moreover, applying a filter to 1D-functions can reduce noise or weight important parts (see chapter 7). Next the 1D-function is transformed by a transform algorithm as, for instance, the fast Fourier transform (see chapter 8). The resulting frequency spectrum describes the frequencies of repeating movements. At this point there are different options to build up a feature vector. One possible option is to use several *main frequencies* (referenced as MF, see chapter 3). Main frequencies are marked by significant amplitude peaks. Another option is the usage of *average amplitudes of frequency intervals* (referenced as AAFI, see chapter 4). By partitioning the frequency axis into intervals of same length, average amplitudes for each interval can be computed. Main frequencies or average amplitudes set up the final feature vector for each video. As a last step a classifier determines the next class to the video on the basis of this feature vector (see chapter 9).

2.2 Image Moments and 1D-functions

In order to compute frequency spectra for video scenes first the motion in each frame has to be localized. Once the motion is detected, image moments and resulting 1D-functions can be derived. The next sections define regions of motion and explain how these regions lead to 1D-functions.

2.2.1 Regions of Motion

Figure 2.2 shows sample video frames with a person troweling a wall. In this scene regions with movement are detected by measuring color differences between two sequential frames. If the color difference of a pixel exceeds a predefined threshold and if there are enough neighbor pixels with a color difference beyond the same threshold, this pixel is considered to be a part of a movement. Thus, a region with motion is represented by the entirety of pixels with motion.
This technique is called *Frame Differencing* and it is one of the simplest background subtraction techniques [77]. It works accurate for static camera recordings. A more detailed definition and explanation can be found in chapter 10.
Pixel differences of the two frames shown in figure 2.2 point out regions with movement, which are visualized by a monochrome image at the bottom. It is obvious that the most active areas are the trowel, the hand and the forearm.

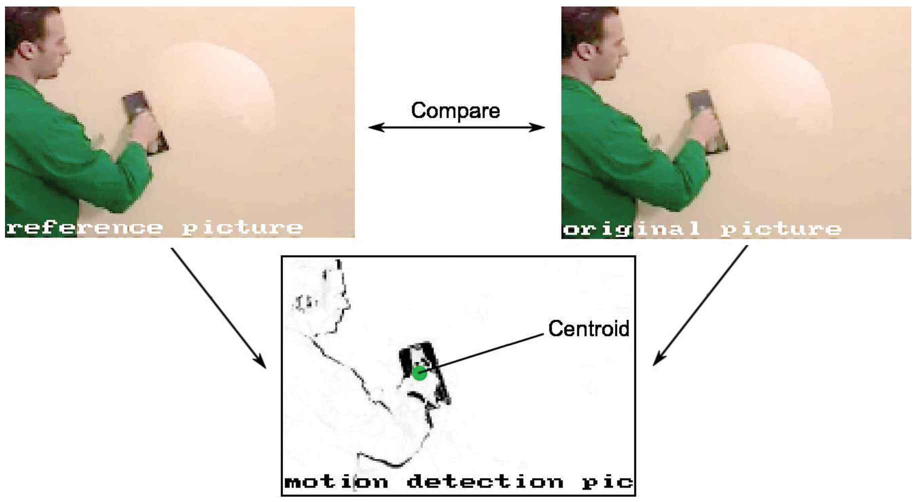

Figure 2.2: Regions with pixel activity and centroid

Therefore, the centroid of regions with motion follows exactly the right hand. As a result the troweling activity sets a specific motion trajectory.

2.2.2 Image Moments

An image moment is a weighted average of pixel intensities of a picture. Functions built up on image moments can describe properties of an image as the area, the orientation or the centroid. In literature image moments and image properties are often used synonymously. There are two main types of image moments: raw moments and central moments. The difference between these two moment types is, that raw moments are sensitive to translation and central moments are translational invariant. There are also further types of image moments, which are scale or rotation invariant. Scale invariant image moments are obtained by normalizing image moments [49]. Binary images, for instance, are normalized by the foreground area. Considering rotation invariant moments most frequently the so-called Hu moments are applied [48]. For a two dimensional monochrome image $b(x, y)$ and $i, j \in \mathbb{N}$ a raw moment M_{ij} is defined as follows [92]:

$$M_{ij} = \sum_x \sum_y x^i \cdot y^j \cdot b(x, y) \tag{2.1}$$

M_{ij} is always of the order $(i + j)$. For a given binary image function $b(x, y)$ the foreground area is determined by M_{00}.

$$(\bar{x}, \bar{y}) = (M_{10}/M_{00}, M_{01}/M_{00}) \tag{2.2}$$

Equation (2.2) defines the centroid of the foreground. Applying centroid coordinates central moments can be computed by equation (2.3) [92].

$$\mu_{ij} = \sum_x \sum_y (x - \bar{x})^i \cdot (y - \bar{y})^j \cdot b(x, y) \tag{2.3}$$

Here μ_{20} and μ_{02} represent the variances of pixels regarding to x respectively y coordinates.

2.2.3 Deriving 1D-functions

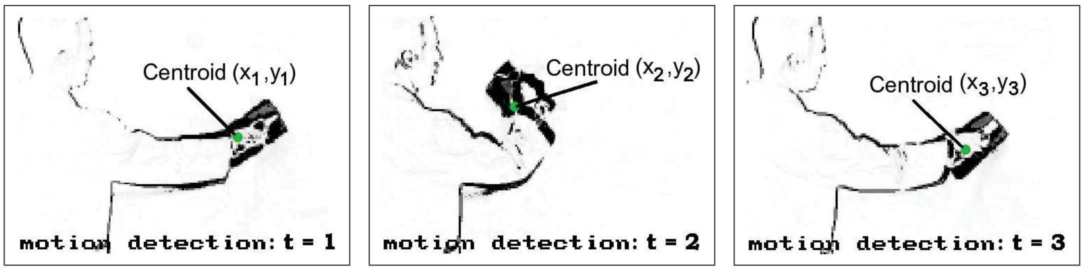

Figure 2.3: Sequence of centroid coordinates

Figure 2.3 illustrates for the troweling video sequence, that the centroid of the motion area is at a different position for each point of time t. For the whole video sequence these image moments describe the main motion and at the same time a 2D-function $f(t)$. Here the term 2D relates to the coordinates x and y depending on time t. Let $(\bar{x}_t, \bar{y}_t) = (M_{10_t}/M_{00_t}, M_{01_t}/M_{00_t})$ the coordinates of a centroid. Then for function $f_c(t) = (\bar{x}_t, \bar{y}_t)$ following applies:

$$f_{c_x}(t) = \bar{x}_t,\ \ f_{c_y}(t) = \bar{y}_t \tag{2.4}$$

In the next chapters $f_{c_x}(t)$ and $f_{c_y}(t)$ are used for experimental test series instead of $f_c(t)$, because transforming 1D-functions results in better accuracies than transforming 2D-functions (see chapter 5). For the same reason two separate 1D-functions of central moments are implemented and tested:

$$f_{v_x}(t) = \mu_{20_t},\ \ f_{v_y}(t) = \mu_{02_t} \tag{2.5}$$

For any 1D-function $f(t)$ we define the speed of an image moment at time t as follows:

$$f_s(t) = |f(t) - f(t-1)| \tag{2.6}$$

The direction of a moment at time t is defined by equation (2.7).

$$f_d(t) = \begin{cases} +1, & \texttt{if } f(t) - f(t-1) > 0 \\ 0, & \texttt{if } f(t) - f(t-1) = 0 \\ -1, & \texttt{if } f(t) - f(t-1) < 0 \end{cases} \tag{2.7}$$

2.3 Radius Based Classifier

A radius based classifier called *RBC* is the basis for the system's classification stage. We published extensive research on this novel classifier in [5] and [9]. In this section the main idea is illustrated and explained. A detailed formalization of the RBC process and complexity can be found in chapter 9.
To a given object o the RBC applies a radius ε in order to find out the number of objects belonging to class C_i within the radius. The normalized sum of all objects leads to a distance between tested object and class. For normalization the class size is used. After computing distances to all existing classes, the RBC assigns

o to the class with the minimal distance. The accuracy of the RBC depends basically on radius ϵ. Hence, this radius has to be adjusted at least for the first usage of the classifier.

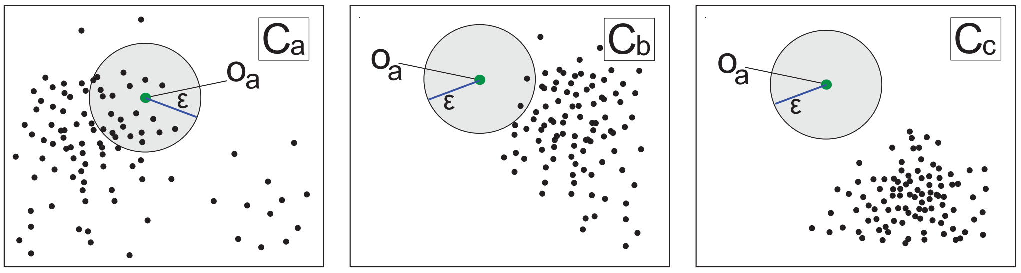

Figure 2.4: Classifying with RBC

Figure 2.4 illustrates how the RBC works: An object o_a of an unspecified class has to be classified. Therefore, it is assigned to each existing class in order to calculate the class with the minimal distance. Three different sample classes C_a, C_b and C_c are given and each class has its own typical object distribution. Assigning o_a to class C_a reveals that there are many objects within radius ε. In class C_b only 2 objects are present inside the given radius. Objects of class C_c are far away from o_a, so there is no object of this class within radius ε. According to these three classes, o_a fits best into class C_a, because it is part of the typical object distribution. At the same time this fact leads to a minimal distance.

Algorithm Radius Based Classifier

```
Input:   test_obj,
         classes,
         $\varepsilon$
Output: next_class_id

min_distance = 1
next_class_id = 0

 for each class ∈ classes  do
    count = 0

    for each class_obj ∈ class  do
       if distance(test_obj,class_obj) < $\varepsilon$
       then count++
    end for

    class_distance = 1 - count/class.size()

    if class_distance < min_distance
    then
      next_class_id = class.getID()
      min_distance = class_distance
 end for
```

The previous algorithm represents the steps of the RBC in more detail. Here the classifier requires three types of input data: first the unlabeled object, second all present classes and third the radius ε. In addition, the next class ID is set to 0 at the beginning of the algorithm. If there is no object in any class inside the ε-neighborhood, the next class ID remains 0 and the object stays unlabeled. Alternatively a next class can be chosen randomly.

2.4 Experimental Setup

Experiments done for this thesis focus mainly on the system's accuracy and runtime. So next we describe the basic experimental setup for these two aspects. Unless specified differently the setup in each chapter is as defined in the following two subsections.

2.4.1 System's Accuracy

Concerning the systems's accuracy own video data and external video data from the online video database YouTube [93] are analyzed. Tests with own video data are calculated by m-fold cross validation. 10 classes, where each class consists of 20 videos, are tested (total 200 videos). More precisely 20-fold cross validation is performed. We divide the video data set into 20 subsets, where each subset contains 10 videos from 10 different classes. So for each iteration the training data set consists of 190 videos and the test set of 10 videos.
External video data is tested by assigning videos to own classes, because cross validation was not possible due to classes with just few clips (total 102 videos). The 1D-function transformation is conducted via FFT. Feature classification takes place by the RBC. Moreover, each video shows one of the following 10 home improvement activities: filing, hammering, planing, sawing, screwing, using a paint roller, a paste brush, a putty knife, sandpaper and a wrench. Since each video is 512 frames long and the video frame rate is 10 fps (frames per second) the analyzed videos have a length of 51 seconds.
Although own videos were taken with varying lighting, slightly different camera positions and different clothing, the own video database is clearly more homogeneous than the external video database. Three major aspects influence the external database and therefore its classification accuracy: Frame quality, camera positioning and motion clearness. The frame quality includes the properties lighting, image sharpness, contrast and noise. Camera positioning means distance, angle and recorded image section. The motion clearness is affected, when hand-held cameras are used. In addition, the acting person himself influences the clearness and regularity of a motion.

2.4.2 System's Runtime

All runtime experiments are realized by a computer with a 2.7 GHz CPU, 250 GB solid-state drive (SSD) and 8 GB RAM. Analyzed videos have a length of 512 frames and each frame has a 320×240 resolution. The transformation of 1D-functions (512 data values) takes place via fast Fourier transform (FFT), hence the resulting frequency axis consists of 256 units. Moreover, 1D-functions are not

filtered as standard. The interval size for AAFIs is set to 4 by default. AAFIs are used by the RBC during classification phase. Furthermore, by default we consider 10 classes, where the class size is set to 20.
As the focus is on the runtime of different filters, transforms, classifiers and AAFI interval sizes, we skip the steps background subtraction, edge subtraction and image moment calculation (see figure 2.1). We start the classification process directly with 1D-functions stored in the database. Otherwise, a fixed additional runtime for each video has to be taken into account.

Chapter 3

Video Classification by Main Frequencies of Repeating Movements

Main frequencies of a repeating movement are the frequencies with the highest amplitudes. Because the more often a clear movement repeats, the higher becomes its amplitude at a particular frequency. Now this chapter describes how to build up and apply feature vectors based on main frequencies respectively amplitude maxima. Major parts of this chapter were published in a similar form by Ayyildiz and Conrad in [5].

3.1 Introduction

In chapter 2 the overall classification process was explained. Now in this chapter the focus is on elementary aspects of the approach. For instance, the feature extraction steps are defined and illustrated. Furthermore, the accuracy of different image moment types and 1D-functions is evaluated experimentally. On the one hand the evaluation aims to expose the overall accuracy of the presented system. On the other hand it aims to figure out the most effective image moment and 1D-function modules. Raw moments and central moments are compared, where raw moments represent centroids and central moments represent variances as defined in (2.2) and (2.3), respectively. The analyzed 1D-functions base on the location, direction or speed of these image moments.

3.2 Related Work

As introduced in section 1.3 strongly related work is given by research papers [62], [25] and [31]. Papers [25] and [31] also present techniques transforming 1D-functions based on centroid coordinates. But both approaches utilize only the main frequency of a frequency domain. Meng et al. present a technique in [62], which extracts seven features from the frequency domain. Since the amplitude maxima based technique introduced in this chapter extracts up to six features, there is a similarity between both methods. Meng et al. analyze activities like walking, running, jumping and skipping. They use a 3D-MLD system

to track multiple human body parts at the same time. Although this system provides motion data for three dimensions, the authors assess only the vertical movements for the reason of computation cost. The Fourier transform of these vertical movements results in a series of frequency domains. Each frequency domain corresponds to a particular body part movement. These frequency domains again are used to detect a gait cycle. Then the whole frequency axis for each frequency spectrum is normalized by this gait cycle. Henceforth, resulting frequencies are relative to the gait cycle and do not depend on the absolute motion speed anymore. Furthermore, the authors sum up the resulting frequency domain to three possible frequencies in order to keep the computation cost low. Along with these three frequencies additional Fourier components and various, summed up frequency magnitudes constitute the final feature vector.
Two steps of this approach bring along some disadvantages: Frequencies relative to the motion cycle cannot reveal information about the absolute repetition speed, which is a central motion feature. This aspect becomes even more important, when the analyzed dataset is not based on a multiple 3D-MLD system. In addition, condensing the frequency axis leads to further information loss.
The next section explains a feature extraction technique based on image moments, so the usage of MLDs is not necessary. Moreover, the motion speed along both the horizontal and the vertical axis is taken into account.

3.3 Main Frequencies as Feature Vectors

As mentioned before each 1D-function can be transformed to its frequency spectrum. In the following chapters mainly the fast Fourier transform (FFT) is applied, since this transform shows up high accuracy. A comparison to other transforms is presented in chapter 8. Frequency spectra commonly reveal several peaks, which correspond to the motion's repetition frequency in a video. These magnitude peaks mark the relevant frequency units. By extracting frequencies at magnitude peaks, a feature vector can be built up and used for classification. Experiments done for this research work have shown, that the extraction of up to three frequencies yields the best results. This is because the typical motion trajectory of the analyzed videos reveals one main frequency and several side frequencies in the most cases. Whereby frequencies at the fourth or fifth highest amplitude peak are not as significant as the frequencies at the first three amplitude peaks. Furthermore, main frequencies or amplitude maxima extraction reduces computation cost, because feature comparison takes places for few values and not for the whole frequency spectrum.
Figure 3.1 depicts a frequency domain with three amplitude maxima marked as horizontal, orange lines. The number of extracted main frequencies with maximal amplitudes has to be consistent for the whole classification process. Moreover, the optimal number depends on the database. It is possible to compute this number automatically by defining a lower bound as illustrated in figure 3.1. The number of frequency peaks above this lower bound can give rise to the optimal number of extracted frequencies. For instance, if all videos of one class have four to seven clear amplitude peaks above the lower bound, then the optimal number of extracted main frequencies would be four, since at least four significant features are available for each video.

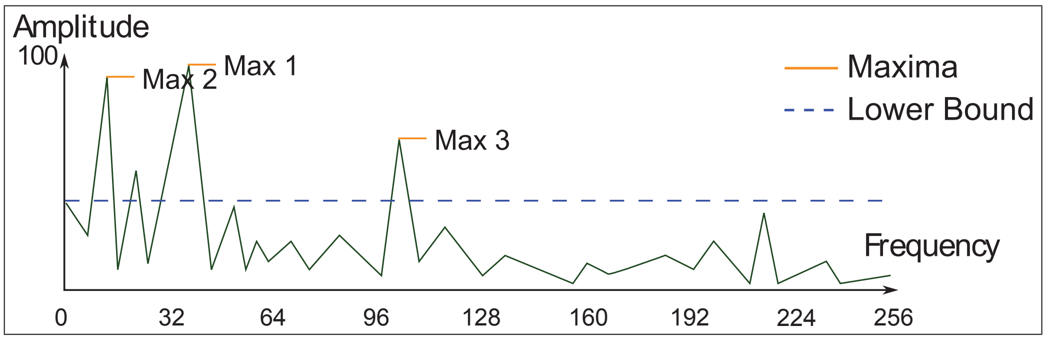

Figure 3.1: Main frequencies illustration

If we assume that the optimal number for a database is n frequencies, then we receive one n-dimensional feature vector for one 1D-function. When building up the feature vector, the order of the features is important. The frequency at the highest peak stands for the first feature, the second highest peak determines the second feature and so forth. Since videos produce two 1D-functions, each video is described by two n-dimensional feature vectors. Hence, the resulting feature vector for each video has $2 \cdot n$ dimensions. Additionally, we assume that the resulting video features have a Gaussian distribution, because repeating motion speed often varies. A classifier can handle these variations in speed as long as its referenced database has an adequate size. So this again means that the system is robust against a variable motion speed.

3.4 Experiments

Now we discuss test series based upon the presented idea of video classification. First, experiments regarding moment type, moment speed and moment direction are considered. Second, translation invariance of motion classification is analyzed and discussed. In both subsections firstly test series with own video data and secondly test series with external video data [93] is performed. Moreover, six- and four-dimensional feature vectors were utilized for own and external videos, respectively. For external videos we use less frequency peaks as features, because for external videos the third highest frequency peak is not as decisive as for own videos with clear recording conditions and movements.

3.4.1 Raw Moments and Central Moments

In figure 3.2 an example of a 1D-function and its transformation is illustrated. The upper plot shows a 1D-function of a clip with a person using a wrench. This function regards to the x-axis coordinate of centroids. One can see, that the centroid moves from left to right and vice versa, which corresponds to the movement of the person. Below this 1D-function its transformation to the frequency domain is plotted, where two maxima can be figured out. Both frequencies at these two maxima are used as feature vectors for this video clip during the classification process.
The following two bar diagrams in figure 3.3 focus on results of test series with raw and central moments. Moreover, results of tests with directional and speed

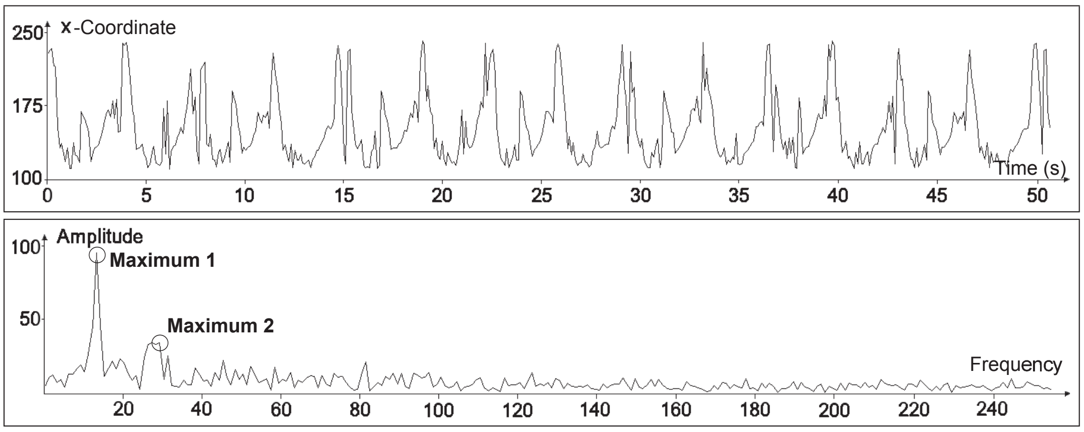

Figure 3.2: FFT of a 1D-function: above 1D-function of a person handling a wrench, bottom FFT of this action.

information of moments are included. Experiments need a tuning of parameter ε for the RBC since different moment types and data sources result in varying features. This tuning step can be performed automatically by approaching the ε with the maximal accuracy. Therefore, multiple test iterations are necessary. Parameter ε is fix for all classes of one test series. It changes only if the image moment type, the 1D-function type or the data source changes.

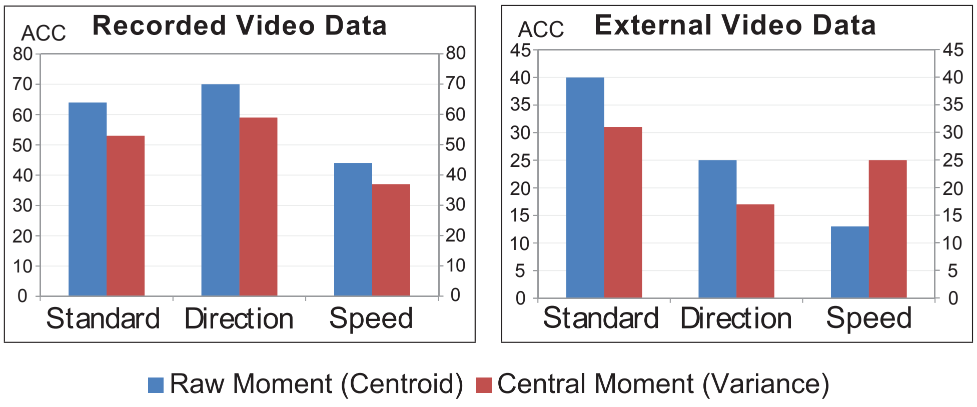

Figure 3.3: Accuracies of tests with raw and central moments

The left diagram in figure 3.3 refers to tests with video data, which was produced especially for the experiments. The right diagram relates to external video data from an internet database [93]. At first glance it becomes apparent, that the raw moments respectively centroids result in better accuracies in almost all cases. There is only one exception for speed information of central moments respectively variances with external data. Further, own videos can be classified much more efficient than external videos. This is associated with the fact, that the external videos have lower quality in the sense of regular movement, camera positioning and scaling. For recorded video data and directional information of centroids the approach achieves a maximal accuracy of 0.70. Experiments with external videos and centroid coordinates result in a maximal accuracy of 0.40. Here centroid coordinates achieve higher accuracies than centroid directions, because external videos contain more irregular movements. For both data sources the speed information of centroids is a weak feature for classification.

We also analyzed the combination of 1D-functions. As a result we detected that there is no further accuracy improvement, because the low accuracy of speed information of image moments decreases the overall accuracy of combined 1D-functions. The accuracies result from both selected features and RBC. In further test series, which can be found in chapter 9, the RBC is compared to other classifiers. Results show that the RBC works accurate. The chosen classifier does not affect the relative highs of feature accuracies.

3.4.2 Translation Invariance

Different positions of one activity in different videos have no effect on the classification process (translation invariance). Figure 3.4 shows how accuracies change, when motion areas are shifted within one video. The translation takes places for each classified clip frame by frame. Furthermore, tests with different shift velocities and shift directions are plotted. Again own and external video sources are integrated. Tests with own videos are performed via directional and tests with external videos are realized via standard information. According to the results of the previous subsection we apply 1D-functions with maximal accuracy. Thus, accuracy decreases caused by translation become much more apparent.

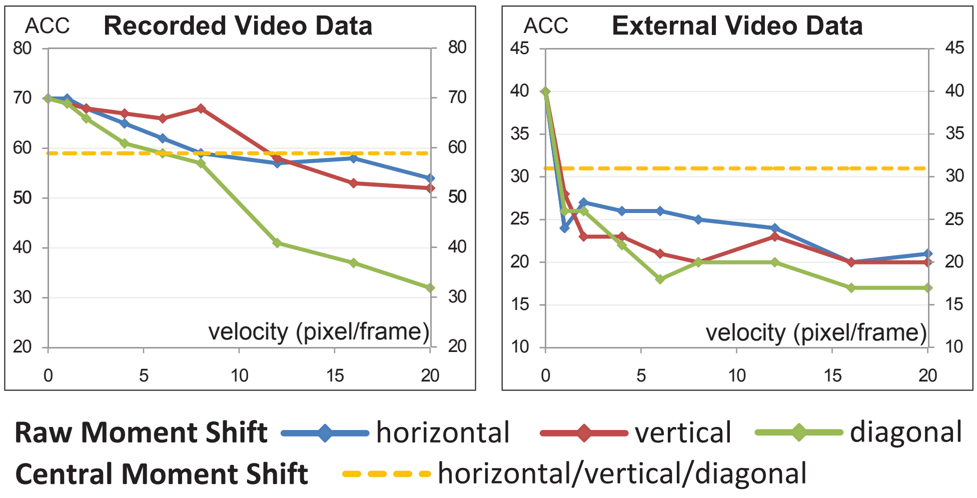

Figure 3.4: Accuracies for image moments with translation

For own videos and centroids the accuracy decreases constantly with increasing velocity of translation. Moreover, accuracies of a diagonal translation decrease much faster than accuracies of horizontal or vertical translation, because shifting a centroid along just one axis does modify just one coordinate. Unmodified coordinates result in unmodified feature vectors. The yellow line shows the accuracy for central moments (variance). For each translation type and velocity the accuracy stays constantly at 0.59. Considering test series with external data it becomes apparent, that accuracies react very sensitive on translation. At the beginning each curve falls rapidly and then decreases constantly. There are two reasons for this behavior: First, external videos depend much more on just one 1D-function and second tests with standard moments are more sensitive to translation than directional information of moments. On the other side here central moments lead to constant accuracies, too. For any translation type and velocity the accuracy is 0.31. According to these experiments it can be stated, that clips with moving objects or moving cameras can often be classified more accurate with central moments than with raw moments.

3.5 Conclusion

This chapter presented the analysis of video classification via main frequencies of repetitive movements. These frequencies can be figured out by transformed 1D-functions of image moments. The experimental stage exposed, that the approach works accurately for centroids as image moments. But for videos including translation of motion central moments work more efficient. Furthermore, a comparison of different 1D-functions showed, that directional information of image moments leads to the highest accuracies for clear camera recordings. If there is camera distortion, then the location information of image moments works accurate.

Chapter 4

Video Classification by Partitioned Frequency Spectra of Repeating Movements

By partitioning a frequency spectrum into intervals of same length average amplitudes for each frequency interval (AAFI) can be calculated. These AAFIs constitute a feature vector, which can be used for classification. The experimental section of this chapter provides distinct analysis of this feature vector. Moreover, central parts of this chapter were published by Ayyildiz and Conrad in [8].

4.1 Introduction

In chapter 3 video classification by main frequencies of repeating motion was analyzed. Now this chapter has an analogous structure and also presented experiments are conducted in a similar way. Here again accuracy analysis is realized for raw and central moments with respect to their location, direction and speed. The main difference is the feature extraction stage. In this chapter AAFIs constitute the feature vector. A detailed definition of this feature type can be found in section 4.3. The analogous structure of the two chapters has the advantage, that the accuracy differences can be compared directly. In addition, the experimental section presents runtime analysis with regard to AAFIs.
The central goal of this chapter is to expose the accuracy of a classification system, which is based on AAFIs as video features.

4.2 Related Work

In this chapter the related work stays almost the same as in the previous chapter about main frequencies. But at this point the experimental setup and results of related work need further discussion, because results offered in the experimental section of this chapter are representative for the whole approach. Most of the related work state high accuracies, but at the same time the size of analyzed datasets is insufficient.
Cheng et al. for instance state high accuracies from 0.85 to 1.0 for their main

frequency based technique [25]. The authors analyze five sports activities, but the average class size is three and therefore not convincing. Meng et al. achieve accuracies from 0.84 to 0.96 [62]. They apply relative frequencies derived from a 3D-MLD system. In total 9 activity classes were evaluated, but unfortunately the authors miss to state the size of tested dataset. Davis and Cutler's method utilizes a single frequency for each video obtained by transforming the self-similarity of motion [30]. Here experimental results depict an accuracy of 1.0. Three classes containing 89 short videos are tested. The length of time is about three seconds for each video. Also He and Debrunner's test series with Hu moments yields high accuracies from 0.87 to 1.0 [44]. But here again only three classes are tested, where each class consists of 16 samples.
Two further approaches, which do not use frequency features directly, are explained in [71] and [12]. Both approaches provide slightly wider datasets than the previous ones. In [71] Polana and Nelson experiment with 7 classes and 40 samples in total. Accuracies lie between 0.80 and 1.0. As mentioned before Babu and Ramakrishnan use motion history information for feature setup [12]. Here 7 different action classes and 51 test samples in total are analyzed. Accuracies range from 0.94 to 0.98 for different classifiers.
On closer examination of the test results and setups two facts become apparent: First, the accuracies are very high and range from 0.80 to 1.0. Second, in the most cases either the number of tested classes or the number of samples is small. So concluding it can be said that stated high accuracies are probably a result of the small datasets. But in real world databases there are often thousands of classes with hundreds of videos in each class [68; 61]. If the number of classes increases, the classification step becomes much more difficult, because the number of classes with similar features increases. Also the class size influences the classification quality. On the one hand big classes can supply further distinctive features, but on the other hand the number of overlapping class features may be increased. The experimental section of this chapter conducts experiments with larger datasets than the referenced papers in order to obtain more decisive results with respect to real world databases.

4.3 AAFIs as Feature Vectors

In the previous chapter we used up to 6 main frequencies for each video as feature vector. Now the whole frequency spectrum is described by AAFIs and feature vectors reveal much more information about the motion type. Each significant frequency high or low has an influence on the concerning AAFI. Here amplitude values have an effect on the classification process, whereas the MF based method applies only the frequency positions.
As mentioned before each 1D-function can be transformed to its frequency spectrum. By partitioning this spectrum into intervals of same length an average amplitude for each interval can be stated. Figure 4.1 depicts this idea by partitioning a frequency spectrum with a length of $m = 256$ units to $n = 8$ intervals. Using the fast Fourier transform, for instance, variables m and n have to be a power of 2, where $m \geq n$. Further, the horizontal, orange lines mark the average amplitude of each interval. Hence, with regard to figure 4.1 one 1D-function leads to 8 average amplitudes respectively to one 8-dimensional feature vector.

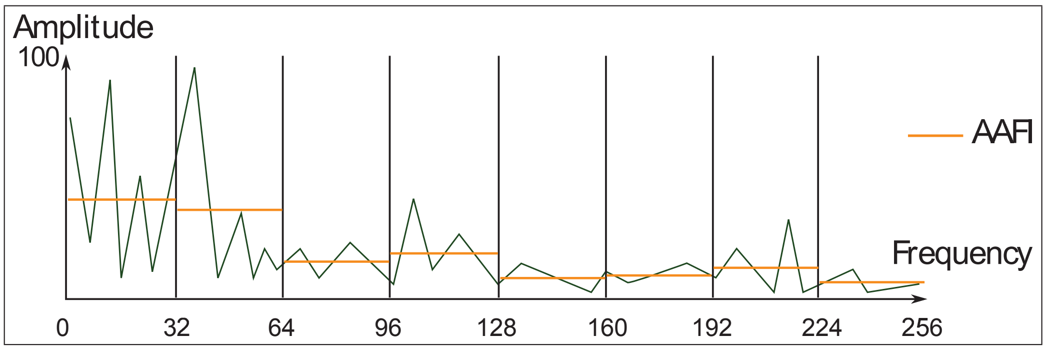

Figure 4.1: Average amplitudes of frequency intervals (AAFIs)

Due to the fact, that videos produce two 1D-functions, each video is described by two 8-dimensional feature vectors in this example. Thus, a partitioning of the frequency spectrum into n intervals results in a $(2 \cdot n)$-dimensional feature vector for each video.
If we compare the AAFI method to a method, which uses every single frequency value, then two advantages become apparent. First, AFFIs reduce the computation cost during classification phase, because the feature vectors are smaller. Second, frequency distortions are compensated by averaging. AAFIs and MFs have a similar runtime, because finding main frequencies takes more time than averaging frequency amplitudes, but at the same time MF feature vectors are smaller.

4.4 Experiments

The AAFI based classification system is analyzed in three steps. First, experiments with regard to moment type, moment speed and moment direction are considered. In addition, these experiments include different interval sizes. Second, translation invariance of motion classification is analyzed and discussed. Third, the runtime of the approach is evaluated. Here class sizes, the amount of classified videos and again interval sizes play an important role.

4.4.1 Raw Moments and Central Moments

Figure 4.2 depicts an example 1D-function, which relates to a person's motion while using a wrench. Particularly the figure plots x-axis coordinates of centroids and captures the main motion. It is obvious that the 1D-function corresponds to the left-right and right-left movements. Transforming this 1D-function by fast Fourier transform (FFT) results in a frequency domain with peaks at 13 and 27. The second plot below depicts the transform to the frequency domain. A partitioning of the frequency axis into $m = 32$ intervals leads to 32 AAFIs. Moreover, the entirety of all AAFIs captures the main information of the frequency domain without considering every single unit. Significant frequency highs or lows have an influence on the concerning AAFI. Moreover, wide ranges with constantly high or low amplitudes are all captured by AAFIs and resulting feature vectors.
The two line charts in figure 4.3 focus on results of test series with raw moments (centroids) regarding interval sizes. Further, results of tests with directional and

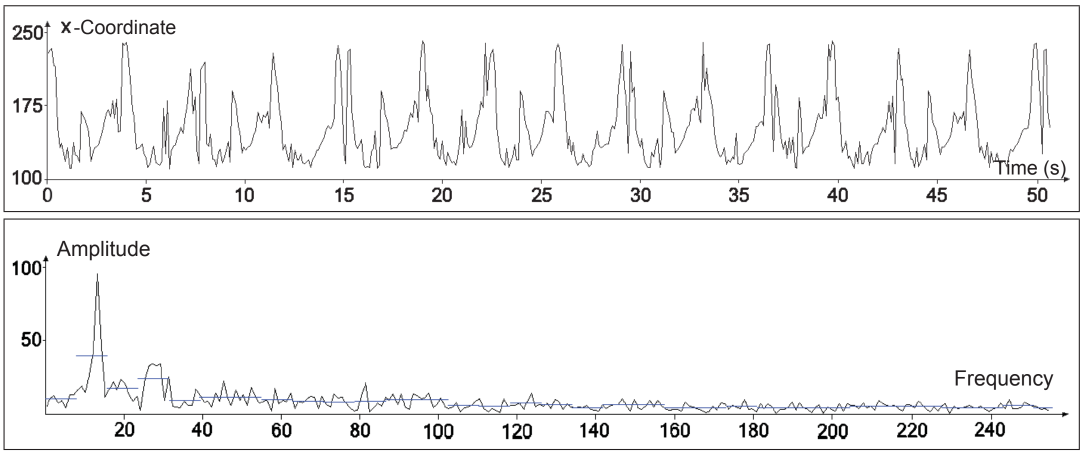

Figure 4.2: FFT of a 1D-function: above 1D-function of a person handling a wrench, bottom FFT of this activity.

speed information of moments are presented. The left chart refers to tests with video data, which was produced especially for the experiments. The right chart relates to external video data from an internet database [93]. At first glance it becomes apparent, that recorded video data results in better accuracies than external video data. This is associated with the fact, that the external videos have lower quality in the sense of regular movement, camera positioning and scaling. Furthermore, for both data sources the optimal interval size lies between 4 and 32.

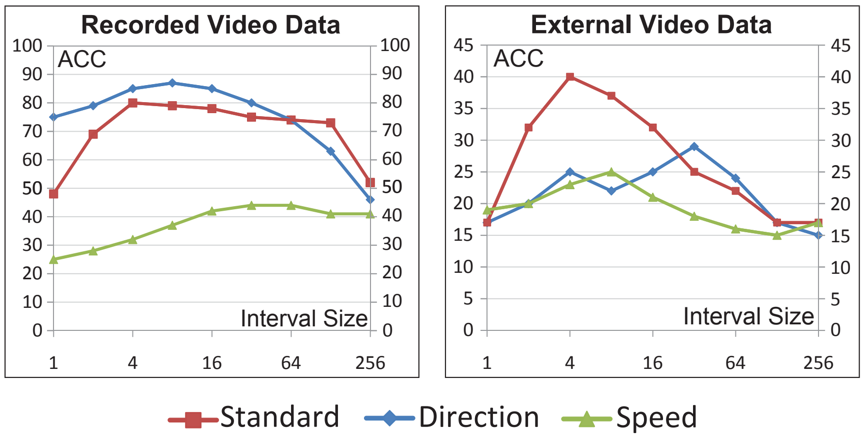

Figure 4.3: Accuracies of tests with raw moments

For recorded video data and directional information of centroids the presented approach achieves a maximal accuracy of 0.87 at an interval size of 8. This means 174 videos of 200 videos are assigned properly. Experiments with external videos and centroid coordinates result in a maximal accuracy of 0.40 at an interval size of 4. Here 41 of 102 clips are assigned correctly. In the case of external videos centroid coordinates achieve higher accuracies than centroid directions, because external videos contain more irregular movements. For the two data sources the speed information of centroids is a weak feature for classifying.
Next we see in figure 4.4 results of tests with central moments (variances). Here own videos result in higher accuracies than external videos, too. But on the

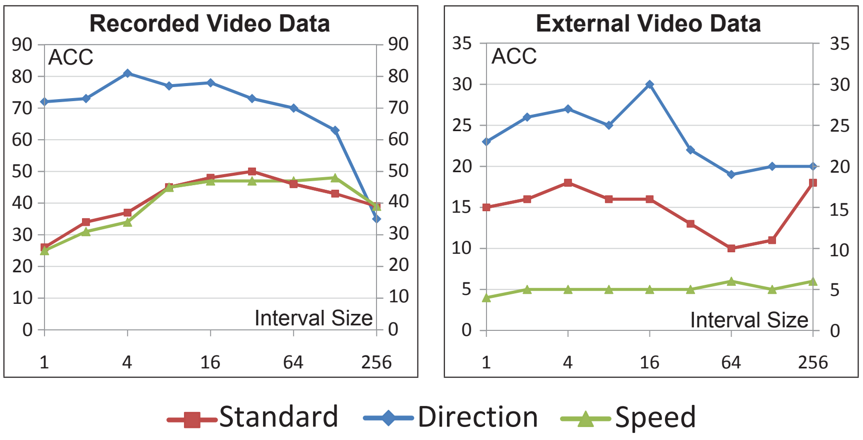

Figure 4.4: Accuracies of tests with central moments

whole accuracies for the two data sources decrease comparing with results of raw moments. For own video data and directional information of central moments a maximal accuracy of 0.81 is achieved at an interval size of 4. In contrast to figure 4.3 standard and speed information have similar line progressions here. The two lines do not exceed an accuracy of 0.50. Compared with centroids utilizing variances for spectral analysis is a weak approach, because motion inside a scene cannot be tracked properly by variances. However, directional information of variances (increase or decrease) is almost as effective as directional information of centroids, because repeating movements of one class can produce very different variances, but the directional information of these variances is mostly similar.
Considering external video data there is an accuracy peak at an interval size of 16 for directional information. The accuracy amounts to 0.30. Standard central moments reveal accuracies between 0.10 and 0.18, speed information of central moments leads to accuracies between 0.04 and 0.06.

4.4.2 Comparison of AAFIs and MFs

Figure 4.5 illustrates a comparison between AAFI and MF accuracies. It becomes apparent, that the AAFI based technique leads to higher accuracies on the whole. It seems that the usage of amplitude highs and more frequency amplitudes improves the results. For instance, considering own videos, raw moments and directional information AAFIs achieve an accuracy of 0.87 and MFs 0.70, respectively. For raw moments and location information AAFIs yield an accuracy of 0.80 and MFs 0.64. Further major accuracy gaps can be stated for central moments combined with directional or speed based 1D-functions. AFFIs lead to an accuracy of 0.81 for directional information and MFs are at 0.59. This is the maximal difference of 0.22. The moment speed based approach results in 0.48 for AAFIs and 0.37 for MFs.
An additional major accuracy gap can be stated for external videos classified via speed information of central moments. In this case the accuracy for the MF based method is more accurate with 0.25, whereas the AAFI based method results in an accuracy of 0.06. Speed information in general is sensitive to distortion of motion. Hence, resulting frequency spectra for the same activity type often vary strongly. For central moments this behavior becomes even more apparent for the

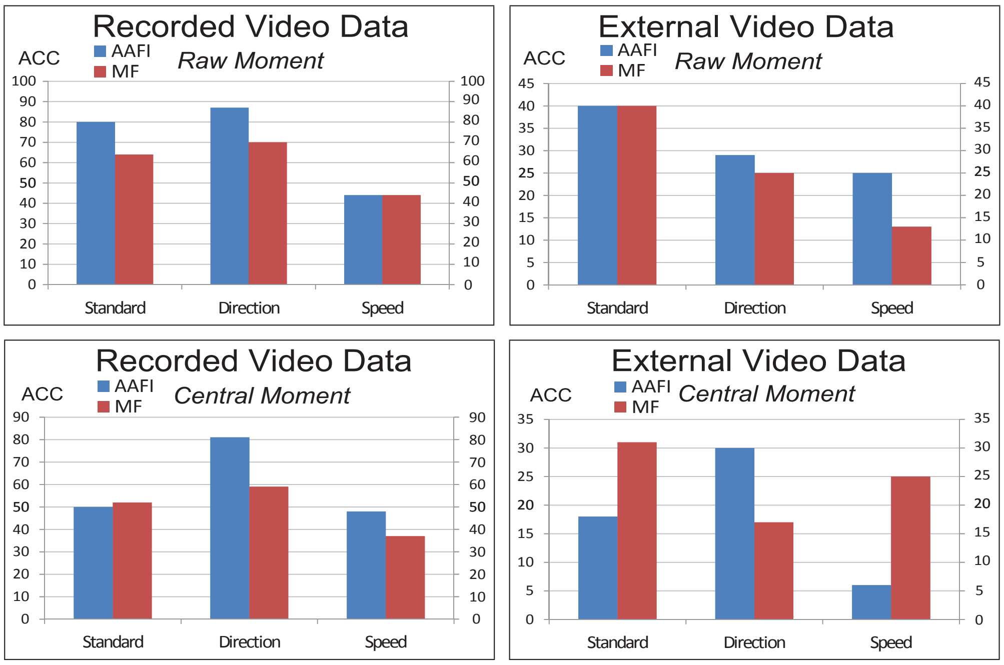

Figure 4.5: Accuracies for AAFI and MF based approaches

most part of frequency areas. But nevertheless frequency peaks stay often at the same position, so for this special case MF based feature vectors work more accurate.

4.4.3 Translation Invariance

This section conducts an analogous translation invariance analysis as presented in section 3.4.2. Main insights stay the same, only the accuracy levels differ. Motion areas are shifted within one video, where the influence of this shift on classification accuracy is depicted in figure 4.6. Again translation takes place for each classified video framewise. Additionally tests with different shift velocities and shift directions are plotted for own and external video sources. For the analysis of own videos directional and of external videos location information of moments is applied.
For own videos and centroids the accuracy decreases slightly by increasing the velocity of translation, if horizontal or vertical shift of motion is realized. Here accuracy starts at 0.87 and ends at 0.75 for vertical respectively 0.72 for horizontal shift. Moreover, accuracies of a diagonal translation decrease rapidly. Starting at an accuracy of 0.87 the accuracy ends up at 0.16. The yellow line shows the accuracy for central moments (variance). For each translation type and velocity the accuracy stays constantly at 0.81. Considering test series with external data, it becomes clear, that accuracies react very sensitive on translation. At the beginning each curve falls abruptly. Then the curves for horizontal and vertical translation stay constantly at 0.27 and 0.23. The accuracy curve for diagonal shift ends at 0.17. On the other side here central moments lead to constant accuracies, too. For any translation type and velocity the accuracy is 0.30.
Just as in the previous chapter test results show again, that moving objects or moving camera recordings can often be classified more accurate with central

moments than with raw moments.

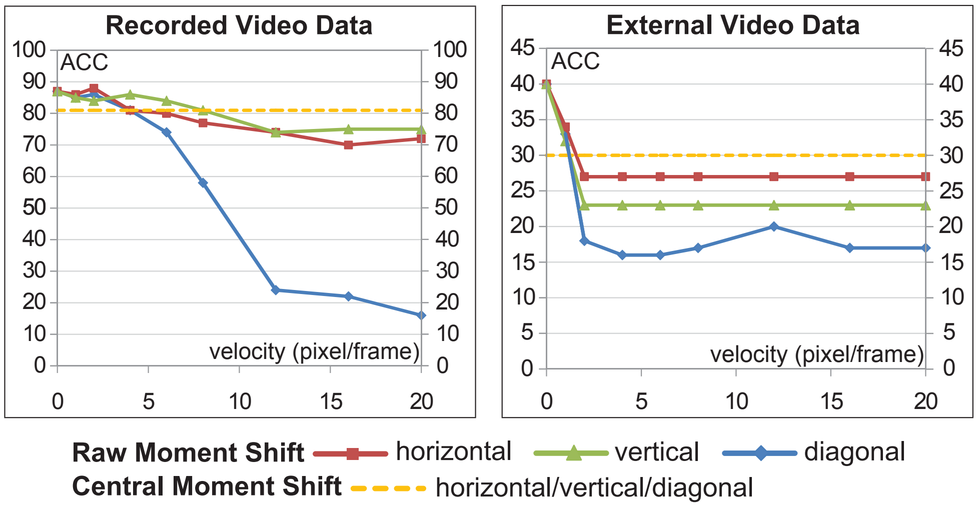

Figure 4.6: Accuracies for moments with translation

4.4.4 Runtime Analysis

After analyzing the accuracy of the system we focus on runtime performance. Therefore, tests with respect to referenced class size, interval size of AAFIs and the number of classified videos are performed.

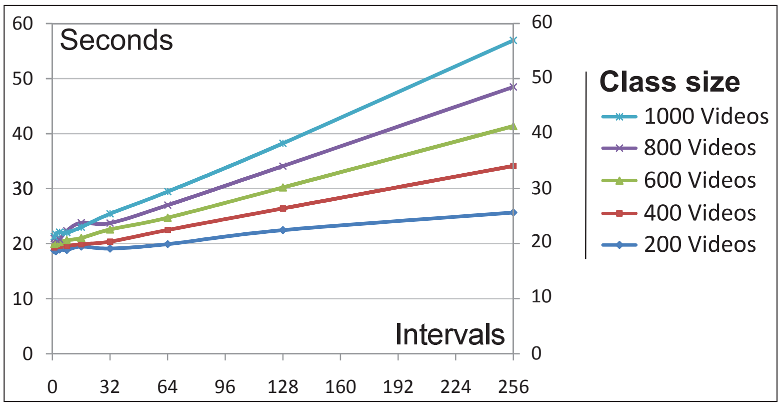

Figure 4.7: Runtime with regard to the number of intervals and class sizes

Figure 4.7 shows the runtime for several class sizes and interval sizes. When we analyze videos with a length of 512 frames via FFT, the frequency axis consists of 256 units. Hence, an interval size of 1 unit results in 256 intervals and an interval size of 256 units results in 1 interval. All line charts in figure 4.7 illustrate runtime in seconds for 200 classified clips. There is an increase for each line, when the number of intervals is rising. At the same time this increase becomes clearer, if the class size of referenced classes increases. This relates to the fact, that the distance of each classified clip to a class is calculated by involving distances to each class object. Hence, big class sizes combined with big interval sizes have just little effect on runtime. But big class sizes combined with small interval sizes do have a clear effect on runtime.

For a referenced class size of 200 videos and utilizing one interval the runtime is 19 seconds. Moreover, for 256 intervals the runtime is 26 seconds. A referenced class size of 1000 videos combined with just one interval results in 21 seconds. In addition, 57 seconds are needed for 256 intervals. There is a linear growth for the increasing number of intervals.

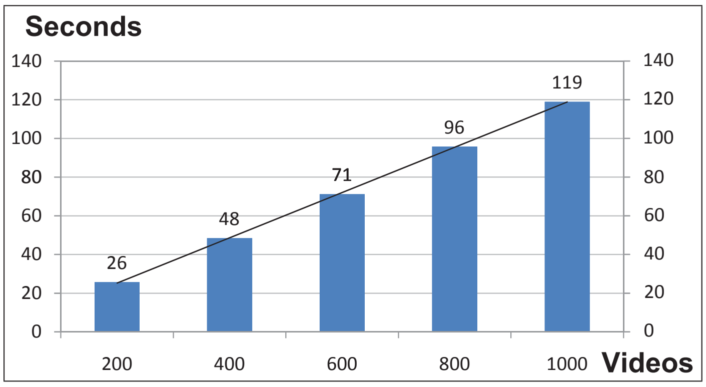

Figure 4.8: Runtime with regard to the number of classified videos

In figure 4.8 we see also a linear growth of runtime. The bar diagram regards to test series with an increasing number of classified clips. In each test the number of intervals is 256 and the number of videos in referenced classes is 200. Beginning at 26 seconds for 200 classified clips the runtime peaks at 119 seconds for 1000 clips. The average time needed for classifying one video is 0.119 seconds. We see, a high number of classified clips has a stronger impact on runtime than a high number of clips in referenced classes.

4.5 Conclusion

The experimental stage exposed, that the AAFI approach works accurately for centroids as image moments. A maximal accuracy of 0.87 could be measured for recorded video data. For external videos the maximal accuracy was 0.40. On the other side for videos including translation of motion translation invariant central moments work more efficient. Here the highs at 0.81 for own videos and at 0.30 for external videos stay constantly at the same level for each shift velocity of motion. Considering interval sizes experiments have shown that an interval size between 4 and 16 for AAFIs gives the best results. Furthermore, runtime tests with big classes combined with small intervals increase the runtime apparently.
With respect to 1D-function or image moment types both the AAFI and the MF based method yield the same insights. But accuracy highs of both approaches differ clearly as shown in section 4.4.2. On the whole AAFIs lead to more accurate video assignments.

Chapter 5

Comparison between 1D-Function and 2D-Function Transforms

Motion trajectories for videos are two dimensional. The transform of a trajectory to its frequency domain can be performed for the two dimensions at once or for each dimension separately. Therefore, this chapter analyzes differences between the two approaches.

5.1 Introduction

In the following sections trajectory functions depending on time t are referenced as 2D-functions. When x- and y-coordinates are considered separately, trajectory functions are referenced as 1D-functions. The transformation of 1D- or 2D-functions is performed via fast Fourier transform. The major purpose of this chapter is, to find out if 1D-function transforms are more accurate than 2D-function transforms. On the one hand it is possible, that 2D-function transforms result in more specific frequency domains. On the other hand a separate consideration of the two dimensions can lead to clear frequency domains for both dimensions. 1D-functions and 2D-functions are transformed via 1D-FFTs and 2D-FFTs, respectively.
The following section 5.2 introduces related work, which analyzes and compares feature extraction by applying 1D-FFTs and 2D-FFTs. Section 5.3 defines 1D- and 2D-FFTs formally. Afterwards, in section 5.4 experimental results for 1D- and 2D-FFTs are discussed. Additionally spatio-temporal trajectories are visualized for illustration purposes.

5.2 Related Work

In the previous chapters we already explained papers [25] and [62]. The two approaches presented by these research papers base on 1D-FFTs. In [25] Cheng et al. consider the series of vertical and horizontal pixel motion vectors for sports videos separately. So after applying two 1D-FFTs they receive two main frequencies for each video sequence. [62] deals with repeating motion of human body

parts tracked by Moving Light Displays (MLD). Meng et al. consider frequency peaks of Fourier transformed MLD curves as features of repeating motion. Here the MLD curves describe 3D motion trajectories, but only the vertical movements are considered for motion transformation. Thus, again a 1D-FFT is performed for each MLD curve. A further cyclic motion based approach is introduced by Allmen und Dyer in [3]. Here the explained method is capable of detecting cyclic motion via spatio-temporal object surfaces. The surface of an object in a two-dimensional video scene yields a three-dimensional pattern as it moves over time. Curves along this resulting pattern are cyclic, if the object motion is cyclic. In this work the authors consider each curve frequency for a set of curves along the 3D-surface separately.
Weinland et al. discuss a free viewpoint action recognition approach [91]. By applying multiple camera perspectives the authors compute 3D motion history volumes for various activities. Motion history volumes describe the flow and the strength of change of pixels. The feature vector for video scenes is set up by Fourier transformed 3D motion history volumes. Here the authors combine features extracted by multiple 1D-FFTs and one 3D-FFT for each activity. Weinland et al. do not consider 1D-FFTs and 3D-FFTs separately, because combining both transforms gives more accurate results. Another technique from the field of pattern recognition and computer vision is presented by Estrozi et al. [35]. This work deals with the analysis of 2D shapes. In more detail Fourier-based curvature estimation is discussed. Numeric techniques for curvature estimation based on 1D-FFT and 2D-FFT are analyzed and compared to each other. The authors state similar estimation errors for both 1D-FFT and 2D-FFT based techniques.
In [79] Singh et al. consider a FFT based OFDM system. OFDM is a multicarrier modulation technique, which is used for wireless data transfer. The approach analyzed in this work first encodes an input signal via IFFT. Second, this signal is transmitted through an Additive White Gaussian Noise (AWGN) channel. As a third step the modified signal is decoded via FFT. In the experimental part of their work the authors state, that the bit error rate (BER) for the 2D-FFT based OFDM system is lower than the bit error rate for the 1D-FFT based system.
Papers [51; 78] belong to the field of earth sciences. The two papers explore large scale gravity and geoid prediction techniques based on 1D-FFT and 2D-FFT. Both papers come to the result, that 1D-FFT based techniques work more accurate than 2D-FFT based techniques. In [51] Hwang applies the inverse Vening Meinesz formula and the deflection-geoid formula by the usage of the FFT. Moreover, in [78] Sideris and She evaluate the discrete Stokes integral by applying the FFT.
As the results of the different research papers show, the accuracies of 1D-FFT and 2D-FFT based techniques depend on the application type. Hence, results of the introduced papers cannot be generalized. Since there is only sparse research done on this topic in computer vision, we aim to analyze the differences between the two approaches in the context of cyclic motion.

5.3 Formalizing 1D and 2D Fourier transforms

In science the Fourier transform finds broad application. Test results in section 8 confirm the efficiency of this type of transform. Now in this section first the

1D Fourier transform is explained and then the 2D Fourier transform, because the 2D transform is based on the 1D transform. For transformation we assume a time discrete and periodic input signal.

5.3.1 1D Fourier transform

Since 1D-functions derived by a series of image moments are finite and periodic, we utilize the discrete Fourier transform (DFT) [40]. Let $\hat{f} = (\hat{f}_0, ..., \hat{f}_{N-1}) \in \mathbb{C}^N$ the discrete Fourier transform of a complex vector $x = (x_0, ..., x_{N-1}) \in \mathbb{C}^N$, which is composed by the interpolation of input values. We set the size of the transform to $2n$. This size plays a role, when the Fourier transform is adapted to the fast Fourier transform. These definitions lead to formula (5.1) with $m = 0, \ldots, 2n-1$. Here $\hat{f}_m$ represents the Fourier coefficient or Fourier component.

$$\hat{f}_m = \sum_{k=0}^{2n-1} x_k \, e^{-\frac{2\pi i}{2n} mk} \tag{5.1}$$

5.3.2 2D Fourier transform

The 2D Fourier transform is simply the 1D Fourier transform applied first to each row and then to each column of a two-dimensional data array [47]. So for this case we consider R rows and consequently a complex vector $x = (x_0, ..., x_{(N-1)\cdot(R-1)}) \in \mathbb{C}^{N\cdot R}$. Thus, discrete Fourier transform $\hat{f} = (\hat{f}_0, ..., \hat{f}_{(N-1)\cdot(R-1)}) \in \mathbb{C}^{N\cdot R}$ follows. The size of the transform is defined as $2n \cdot 2r$. For $m = 0, \ldots, 2n-1$ and $q = 0, \ldots, 2r-1$ formula (5.2) results. Here $\hat{f}_{m,q}$ again represents the Fourier coefficient.

$$\hat{f}_{m,q} = \sum_{k=0}^{2n-1} \sum_{j=0}^{2r-1} x_{k,j} \, e^{-2\pi i (\frac{mk}{2n} + \frac{qj}{2r})} \tag{5.2}$$

5.3.3 Fast Fourier transform

By applying the algorithm of Cooley and Tukey we receive the 1D and the 2D fast Fourier transform (FFT) [27]. FFT works with DFTs of half length and consequently needs the quarter of complex computations. Further on, the FFT implements a divide-and-conquer method. Depending on the length of the measured value sequence the method can be realized multiple times, whereby the runtime amounts to $\mathcal{O}(n \cdot log(n))$ for 1D transforms and to $\mathcal{O}(n \cdot r \cdot log(n \cdot r))$ for 2D transforms.

5.4 Experiments

This section is concerned with the visualization and the accuracy of 1D- and 2D-function transforms. As the previous chapter exposed, AAFIs as frequency features work more accurate than MFs. So here again AAFIs are applied for experimental purposes.

5.4.1 Transform by 1D-FFT and 2D-FFT

Figure 5.1 depicts the x- and y-coordinate trajectories for a hammering video with a 20 fps frame rate. We see clear up and down movements along the y-axis and slight right and left movements along the x-axis.

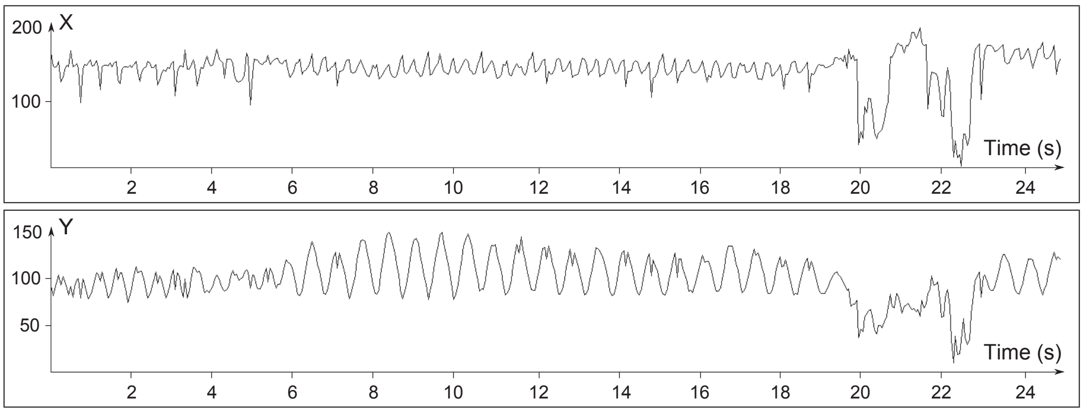

Figure 5.1: Top: centroid's x-coordinate trajectory. Bottom: centroid's y-coordinate trajectory.

The corresponding 1D-FFTs are presented in figure 5.2. For the y-coordinate's trajectory an amplitude peak around frequency 38 can be stated. For the x-coordinate's trajectory transform there are three interesting amplitude peaks. The first amplitude peak around frequency unit 10 results from an overall volatility along the x-axis. The second peak around 38 corresponds to the movements along the y-axis. As figure 5.1 illustrates, one up and down movement of the hammer leads to two slight centroid movements along the x-axis, thus a third frequency peak around 76 appears.

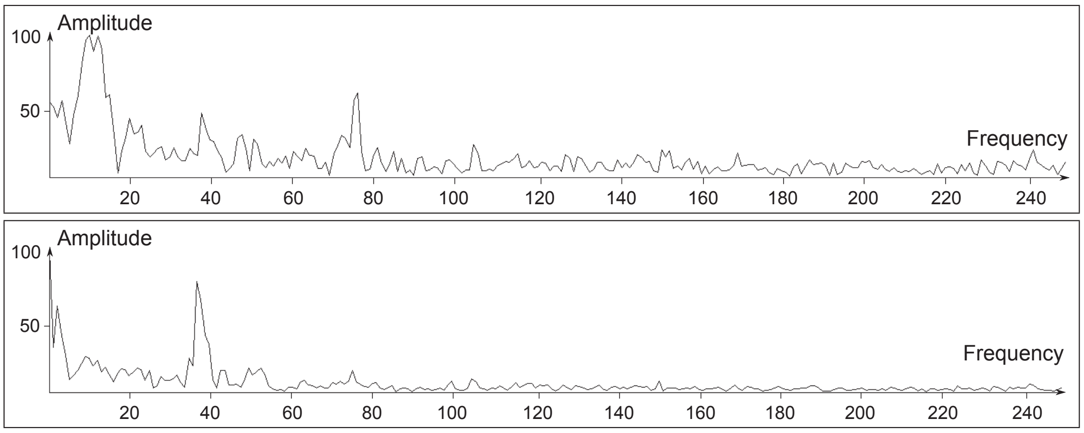

Figure 5.2: Top: 1D-FFT of the x-coordinate trajectory. Bottom: 1D-FFT of the y-coordinate trajectory.

Next figure 5.3 plots the same hammering motion as a three-dimensional spatio-temporal trajectory. In this representation the vertical and the corresponding double time horizontal movements become clearly apparent.

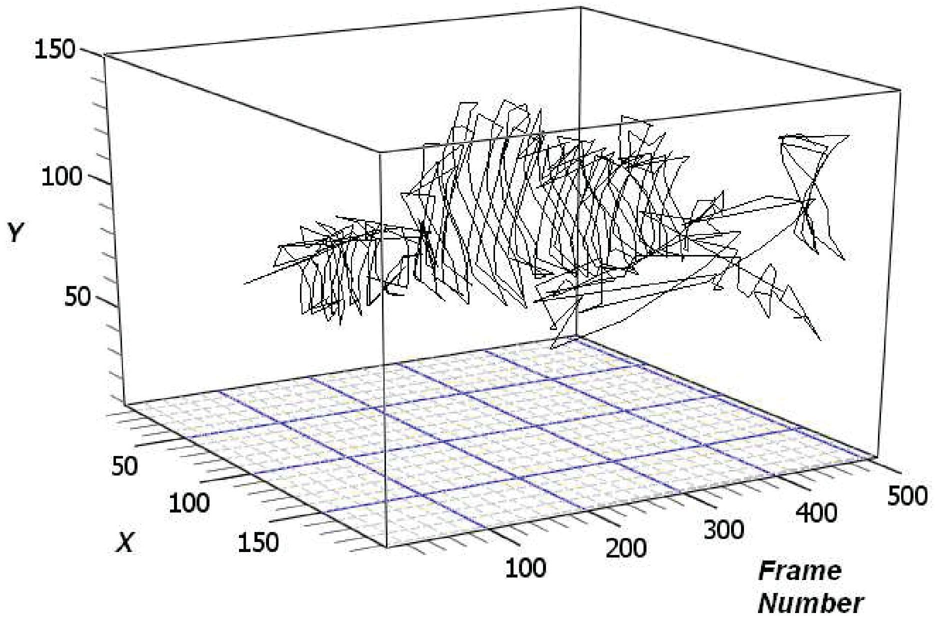

Figure 5.3: Spatio-temporal centroid trajectory

By transforming this 2D-function depending on time t or depending on the frame number, we receive a different frequency domain as illustrated by figures 5.4 and 5.5. This frequency domain shows up two dependent rows. Hence, not only the horizontal frequency rows, but also the vertical frequency columns reveal frequency information.

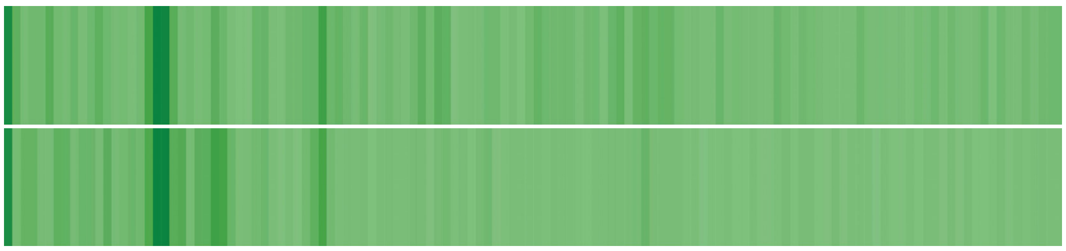

Figure 5.4: Visualization of the 2D-FFT frequency spectrum

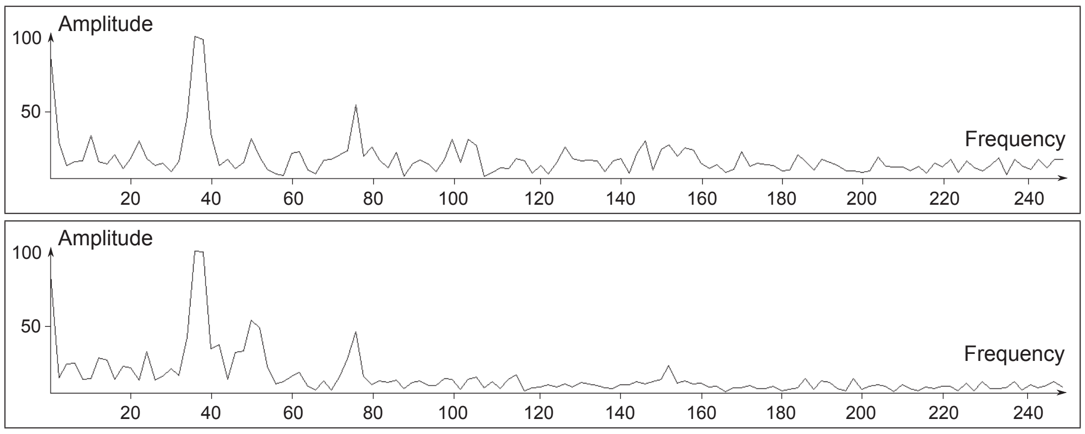

Figure 5.5: Top: first row of the 2D-FFT. Bottom: second row of the 2D-FFT.

Nevertheless, the resulting horizontal frequency domains are similar to the frequency domains of the 1D-FFTs in figure 5.2. But there are three important differences for this instance: First, the peak around frequency unit 10 for the x-axis trajectory in figure 5.2 is eliminated. This peak is more a side effect than a meaningful feature for the classification process. Moreover, the two correct frequency peaks around 38 and 76 are depicted. In addition, the frequency amplitude around unit 48 is emphasized and becomes apparent. Although both 1D-FFTs hint at this point, only the 2D-FFT brings this frequency position out.

5.4.2 Accuracy of 1D-FFTs and 2D-FFTs

As explained and illustrated in the previous sections 1D- and 2D-FFTs create different frequency domains. This again leads to varying frequency features and therefore to varying accuracies. Now by figures 5.6 and 5.7 we expose accuracies for both approaches.
Figure 5.6 shows accuracies for own videos classified via directional, location and speed information of centroids. For each type of input function 1D-FFTs perform more accurate than 2D-FFTs. The accuracy for 1D-FFTs and 2D-FFTs based on centroid directions is 0.87 and 0.76, respectively. So for this case the accuracy difference is 0.11. For own videos accuracy differences range from 0.11 for input functions based on centroid directions to 0.18 for input functions based on speed information of centroids. Hence, a significant accuracy decrease for 2D-FFTs can be stated.

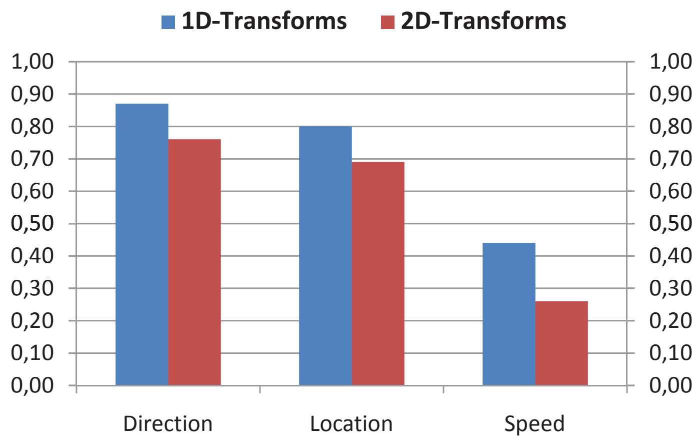

Figure 5.6: Classification accuracy for own video dataset with 1D- and 2D-transforms

Next in figure 5.7 accuracies for external videos are depicted. Here we detect similar absolute accuracy decreases. If we consider the relative decreases, then differences become even more apparent. For external videos differences range from 0.09 for centroid speed based functions to 0.15 for centroid position based functions.
Results of the presented test series illustrate, that the transform of two 1D-functions yields a more accurate feature vector than the transform of one 2D-function. There is one main reason for this behavior: The usage of 2D-FFTs leads to dependent frequency domains. So motion along the x-axis affects both

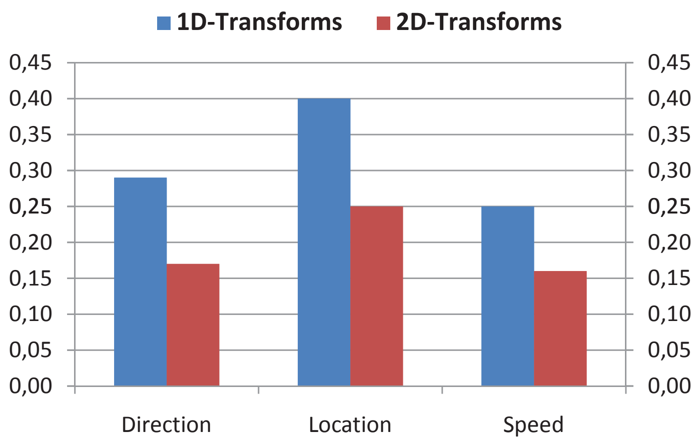

Figure 5.7: Classification accuracy for external video dataset with 1D- and 2D-transforms

frequency domains and motion along the y-axis also does. After the transformation process the heritage of amplitude peaks cannot be distinguished and therefore a central information for the classification process is lost. In videos motion and as a consequence frequencies often can be assigned to a specific direction.
To be more precise, let $x(t)$ and $y(t)$ the x- and y-trajectories. Then $X(\omega)$ and $Y(\omega)$ are the related Fourier transforms with $|X(\omega)|$ and $|Y(\omega)|$ as their one-dimensional power spectra. Consequently, we can define the two-dimensional power spectrum by $|X(\omega)+Y(\omega)|$ and $|X(\omega)-Y(\omega)|$, because the second dimension covers only two components. This definition reveals that the two horizontal lines of the two-dimensional power spectrum contain similar information. For instance, by this notation it becomes obvious, that clear peaks for motion along one axis will often be visible inside both lines of the two-dimensional power spectrum. This is the basic reason, why the heritage of amplitude peaks cannot be assigned to one specific axis. And this information loss leads at the time same to the accuracy decrease for 2D-FFTs.

5.5 Conclusion

This chapter explained the difference between 1D-function and 2D-function transforms. Since the two types of transforms lead to useable feature vectors, the experimental stage exposed benefits and disadvantages of the two approaches. Although 2D-FFTs lead to more specific frequency domains and eliminate false amplitude peaks, the motion direction is poorly considered here. Hence, a separate analysis of motion along both axes via 1D-functions and 1D-FFTs turns out clearly higher accuracies.

Chapter 6

Analyzing Invariance of Frequency Domain based Features

Features based on the frequency domain work accurate, when the system detects clear and repeating movements. If there is rotation, reflection, scaling, translation or time shift in a video scene, the detected main motion varies. In this chapter the robustness of frequency features against these variations is analyzed. Core parts of this chapter were published by Ayyildiz and Conrad in [6].

6.1 Introduction

Previous chapters primarily evaluated a certain module of the classification system. Now the application as a whole is not modified during the experiments. We only take a closer look at the aspect of invariance. Since video databases ordinarily contain videos with different camera angles, zooming factors or object positions, this aspect plays an important part. Hence, this chapter will assess experimentally, how robust the presented approach works with varying camera settings.

6.2 Related Work

Translation invariant methods for human action classification can be found in [36; 19; 65], where approaches of Fanti et al. [36] and Bobick et al. [19] also fulfill scale invariance. A bulk of literature refers to rotation invariant motion classification. In [24] and [16] rotation invariant methods for motion trajectory recognition are presented, where [24] provides only planar rotation invariance. Results in [16] resemble presented test results of this chapter, but the maximal number of classes is set to 5 and the maximal angle is 60°.
Further on, some research work provides methods with rotation and scale invariance at the same time: Papers [91; 72] are based on motion history volumes respectively motion trajectories, whereas [1] utilizes self-similarity plots resulting from periodic motion. Unfortunately the research work of Weinland et al. [91] and Rao et al. [72] does not analyze rotation invariance satisfactorily. The formal

definitions and experimental results show, that the methods are rotation invariant. But the effect of rotation on the system's accuracy is not evaluated. The approach of Abdelkader et al. [1] achieves high accuracies for a wide range of different camera angles. For a 1-nearest neighbor classifier and using normalized cross correlation of foreground images seven out of eight angles have an accuracy above 0.60. A comparison of this work to our work is not possible due to the fact, that Abdelkader et al. consider only one class for their classification process.
As explained in previous chapters He and Debrunner compute Hu moments for regions with motion in each frame [44]. Afterwards their system counts the number of frames until a Hu Moment repeats and define this number as the motion's frequency. Hu moments are invariant for translation, planar rotation, reflection and scaling. Here the periodic trajectory of an object cannot be ascertained. In paper [62], which is explained and discussed profoundly in the previous chapters, Meng et al. state that their 3D-MLD and frequency domain based technique is time shift invariant. In addition, the authors show experimentally that their technique is robust to trajectory distortions.

6.3 Experiments

This section evaluates the presented video classification system with respect to the robustness against varying camera settings. Varying camera or recording settings primarily mean aspects of invariance in this context: First, rotation invariance with 9 different camera angles is analyzed. Then translation invariance is considered by shifting objects with repeating movements. Third, scale invariance with different zooming factors is tested. The fourth major analysis deals with invariance with regard to time shift of an activity.
Test series are performed by own and by external video data. Experiments with own videos are conducted by the direction information of centroids. For external videos centroid locations are the basis for the 1D-functions. AAFIs constitute the feature vectors. Further, in section 6.3.3 240 self recorded video sequences from different camera angles are classified. Here again each video belongs to one out of 10 home improvement classes.

6.3.1 Reflection Invariance

Reflection inside video frames changes the centroid positions of an activity. Considering figure 4.2 for instance a horizontal reflection of the wrench handling activity would lead to a vertically reflected plot. But this resulting plot reflection has no effect on the resulting frequency domain, because the frequency of the motion does not change. This applies not only to horizontal reflection of video frames, but also to vertical and diagonal reflection. Hence, frequency domain based AAFI features are reflection invariant.

6.3.2 Planar Rotation Invariance

Without detailed experimental analysis this subsection explains the effect of planar rotation on the system's accuracy. Planar rotation has a much stronger influence on the classification process than spatial rotation, since planar rotation

changes image moment values along both axes at the same time. A planar rotation of 90° for instance completely swaps image moment values along the x- and y-axis. So a classification by the presented raw and central moments is not possible for this case. Only small variations of the planar rotation angle can be handled accurately.
Alternative planar rotation invariant image moments are for instance Hu moments [48], which are also used by He and Debrunner [44]. The usage of Hu moments needs a reference database for the classifier, which is built up on features derived by Hu moments. Hence, a planar rotation invariant version of the application is feasible, if the image moment extraction module is exchanged.

6.3.3 Spatial Rotation Invariance

The next three test series focus on rotation invariance of the presented classification method. For each test series raw moments (centroids) are utilized. Videos from 9 different camera angles are classified. Except videos recorded from a frontal point of view 30 videos for each angle are assigned to one of ten classes. Each of these classes consists of 20 videos recorded from a frontal view (total 440 videos in database). Frontal view videos are assigned by m-fold cross validation.

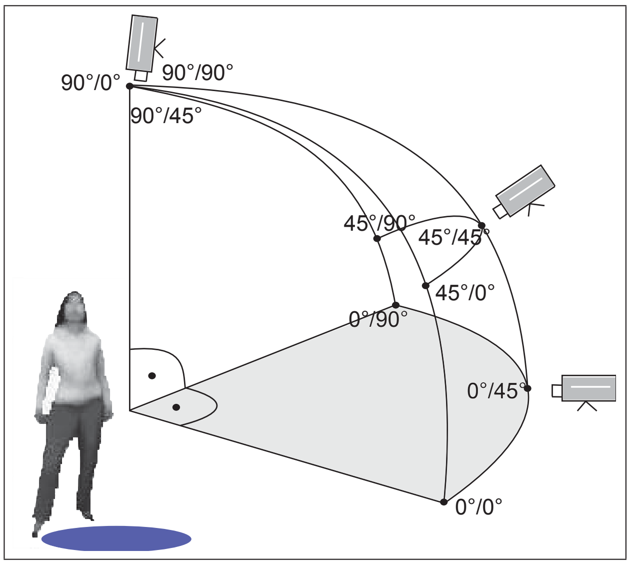

Figure 6.1: Illustrating camera angles

The bar chart in figure 6.2 depicts experimental results for especially recorded video data using directional information of image moments. In addition, AAFI interval sizes are set to 8.
Videos recorded from a frontal point of view achieve a maximal accuracy at 0.87. The higher the horizontal and vertical camera angle, the lower the accuracies. The lowest accuracy is marked at 0.23 for a 90°/90° angle. This behavior is related to the fact, that the referenced classes contain only videos with a frontal camera position. In addition, a frontal point of view gives clearer motion. Nonetheless, 7 out of 9 camera angles achieve at least an accuracy of 0.40 and the average accuracy is 0.48.
This means the presented approach works, even if the point of view is rotated. There are two main reasons for this observation: First, if the angle is enlarged

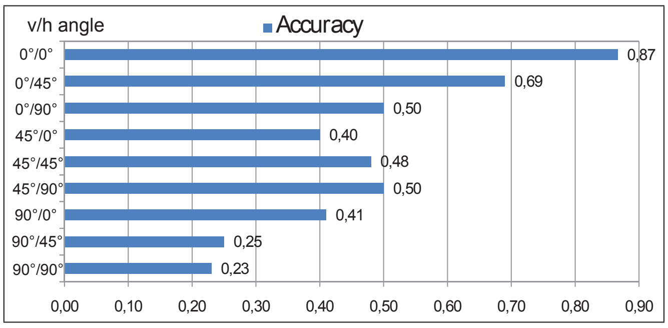

Figure 6.2: Accuracies of tests with raw moments and directional data

along just one axis, motion along the other axis stays almost unchanged. So for the motion feature vector of one axis there are little changes. Second, even if the camera angle changes, the frequency of a movement stays the same. Only the clearness of the motion direction descends.
Figure 6.3 shows experimental results for own video data using position information of image moments. Here AAFI interval sizes are set to 4. Frontally recorded videos result in a maximal accuracy of 0.80. Accuracies for each angle are varying strongly and the overall accuracy falls to 0.43. The lowest accuracy measured is 0.21 for a 90°/90° angle. Altogether 5 out of 9 camera angles give an accuracy of at least 0.40. In contrast to directional information of moments the position of moments is much more sensitive to camera angles, because the range of a movement affects directly the average amplitude of each frequency interval.

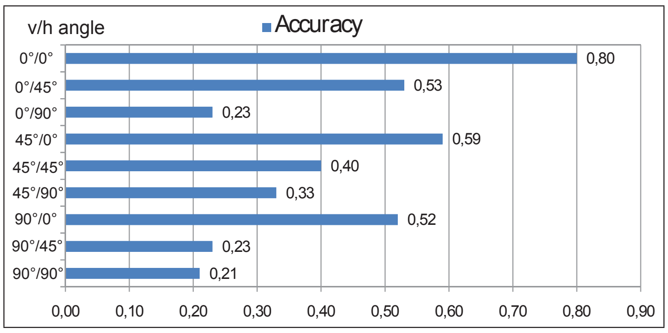

Figure 6.3: Accuracies of tests with raw moments and positional data

Settings for tests regarding figure 6.4 are the same as for figure 6.3. The only difference is that we normalize here frequency values for classified clips as well as for referenced clips. Normalization is realized by dividing each frequency by the amplitude maximum of the whole frequency spectrum. Thereby AAFIs of classified and referenced clips stay at the same level, even if the camera angle changes. Here experimental results do not vary as strong as in figure 6.3 and the average accuracy ascends to 0.45. 6 out of 9 camera angles yield an accuracy about 0.40.

6.3.4 Translation Invariance

As explained in previous chapters varying positions of one activity in different videos do not influence the classification process (translation invariance). But

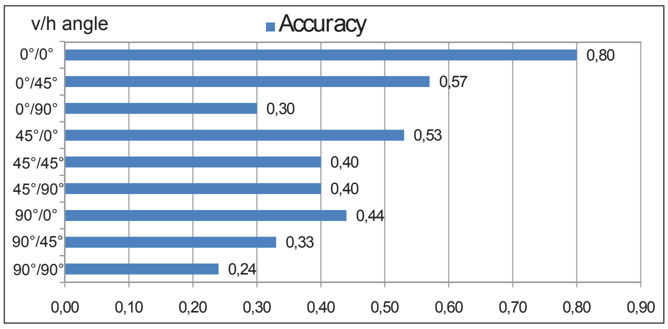

Figure 6.4: Accuracies of tests with raw moments, positional data and normalized frequency domain

shifting motion areas within one video influences the classification process. Next figure 6.5 illustrates how accuracies change in this case. For each classified clip translation takes place frame by frame. Further, figure 6.5 shows test results for different shift directions and shift velocities. Tests with own videos use directional information of moments and tests with external videos are performed by position information.

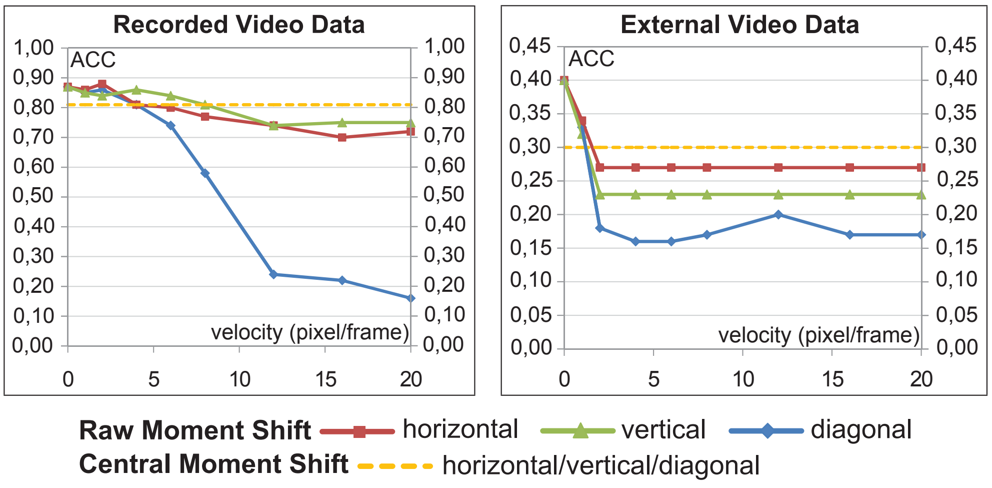

Figure 6.5: Accuracies for moments with translation

A detailed discussion of the experimental results depicted by figure 6.5 can be found in section 4.4.3. As a main insight we can state, that video sequences with moving objects or moving cameras can often be classified more accurately with central moments than with raw moments. It should be taken into account that sequences recorded with a moving camera need edge subtraction.

6.3.5 Scale Invariance

Next two line charts present experimental results for scale invariance of the presented approach. The first line chart shows results for tests with own videos and the second line chart regards to external videos. Both internal and external video sequences are analyzed via raw moment positions (centroids), since moment directions are always scale invariant. This is related to the fact that a direction can only be -1, 0 or 1 (see (2.7)). Scaling has no effect on these values. Also the usage of scale invariant image moments would lead to constant accuracy values. Experiments are conducted for 10 different scale factors beginning at 0.25 and ending at 4.0.

Figure 6.6 illustrates how accuracies decline when the scaling factor decreases or increases. By decreasing the zooming factor accuracies fall faster than by zooming in, because the clearness of a motion depends on the range, too. A zooming factor of 1.5 achieves an accuracy of 0.74 and a factor of 0.67 achieves 0.59. For zooming factors 0.5 and 2.5 accuracies stay above 0.30. So the approach works even for position information of raw moments as far as the zooming factor is not too high or too low. Normalizing frequency spectra by maxima or averages of each spectrum leads to constant accuracies 0.80 and 0.74, respectively. Raw moment directions give a maximal constant accuracy for all scaling factors at 0.87.

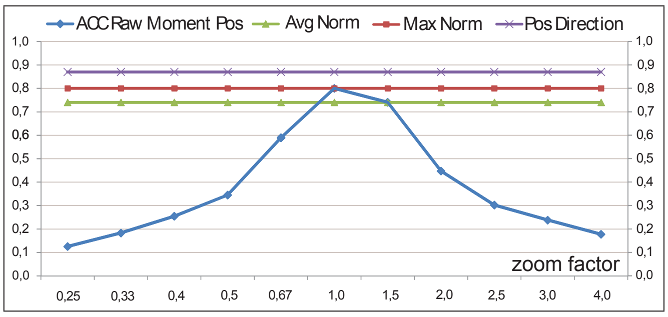

Figure 6.6: Accuracies for internal videos and different zooming factors

Now in figure 6.7 we see the same effect as in figure 6.6. Accuracies decline when scaling factor decreases or increases. There is just one exception for scaling factor 1.5. For this factor accuracy increases from 0.40 to 0.42. This behavior is associated with the referenced classes. External videos are assigned to own video classes, where distances between camera and moving object in external clips are bigger than in own clips. Hence, a zoom in aligns external and referenced AAFIs. A normalization of frequency spectra by maxima or averages of each spectrum results in constant accuracies 0.32 and 0.28. By utilizing moment directions the accuracy for each test series stays at 0.30.

Once again we can see that a zoom out has a stronger effect on the accuracy descend than a zoom in, because motion ranges become smaller. Here this effect becomes even more apparent than in figure 6.6, because external videos reveal more irregular motions.

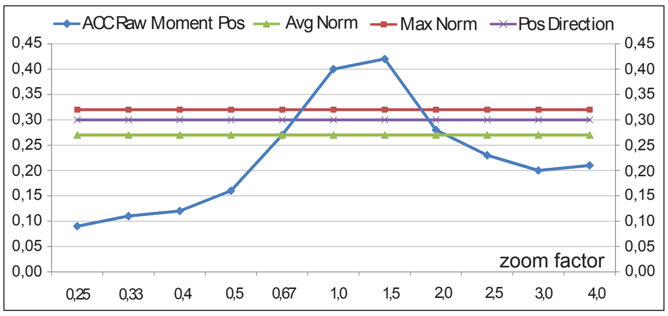

Figure 6.7: Accuracies for external videos and different zooming factors

6.3.6 Occlusion Invariance

In computer vision occlusion is a known research area, since occlusion of foreground objects can be detected in many real world databases. Occlusion can be caused by any object between the camera and the foreground object. As a result of occlusion only parts of the original motion areas can be detected. This again leads to different feature vectors of a specific activity.
Chapter 7 provides experimental results for foreground object occlusion combined with distortion filtering. Here home improvement activities are occluded by horizontal or vertical bars. The orientation of a bar depends on the motion's direction and the thickness depends on the motion's range. Each bar aims to occlude the main motion area essentially. Results show that the presented frequency based approach can handle distortions caused by occlusion. Maximal accuracies of 0.72 for own videos and 0.37 for external videos are stated. Distortion filtering raises the accuracy for own videos even to 0.81.
These results are not restricted to occlusion by bars. Also other patterns as a net, a grid or dots spread across each frame can be handled accurately, since the overall motion is always detectable. This is a further advantage of the frequency based technique towards other techniques. For instance, key frame based or motion pattern based approaches are not able to handle occlusion as standard.

6.3.7 Data Distortion Invariance

Depending on the video quality it is possible that image moment trajectories are distorted or that input data is missing. For those cases Meng et al. analyze simulated data distortion in [62]. They show that missing data can be effectively replaced via interpolation. For experimental purposes they cut randomly data from MLD motion trajectories. Then several interpolation methods such as linear, cubic spline, cubic Hermite polynomial interpolation are applied. Cubic spline interpolation as illustrated in figure 6.8 shows up the best results.

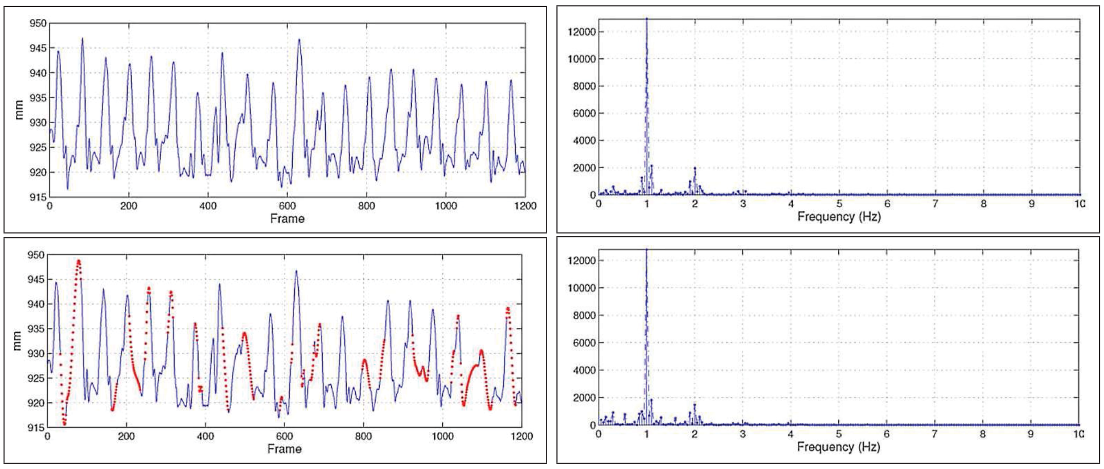

Figure 6.8: Interpolation of simulated data distortion and the effect on the resulting frequency domain by Meng et al. [62]

The upper left plot in figure 6.8 shows 20 hip motion cycles. Below the same trajectory is plotted with 30 randomly generated gaps, which again are interpolated by cubic spline. Interpolation is marked by the red dots. As the frequency domains for the standard and the interpolated trajectory show, the data distortion has only little effect on the frequencies. Hence, the authors depict in their work, that a frequency domain based approach is robust against missing data or data distortion.

6.3.8 Time Shift Invariance

Now we focus on time shift invariance of AAFIs. In the present context time shift means, that the analyzed video starts at different points of time. In order to obtain regular shifts, we use sliding windows with a window size of 256 frames. The full length of a video is 512 frames. The window is shifted along the time axis stepwise. After each shift the action inside the sliding window is classified. Figure 6.9 illustrates this idea for a 512 frames long video sequence.

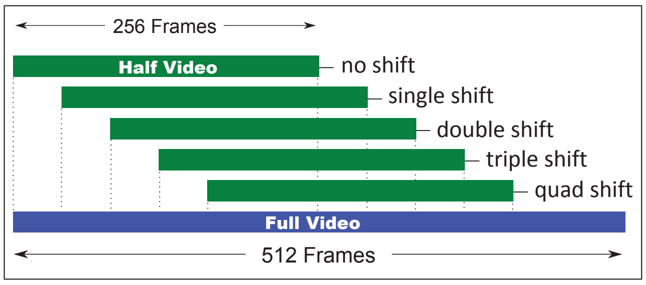

Figure 6.9: Time shift illustration

This technique is used for internal and external videos during classification stage. Again internal videos are classified via raw moment directions and external videos are assigned by raw moment locations. In figure 6.10 it becomes obvious, that the starting point of a repeating movement has only a slight effect on frequency spectra and resulting feature vectors. Here 256 frames of 512 frames long clips are shifted along the time axis and classified. Each shift has a length of 10 frames. Own videos stay for each shift around an accuracy level of 0.80 and external videos stay around 0.30. Hence, the approach is almost invariant with regard to time shift.

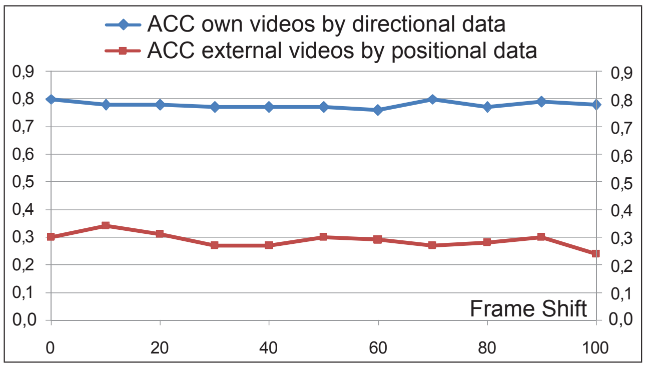

Figure 6.10: Accuracies for videos with different starting times

6.4 Conclusion

This chapter analyzed reflection, view, scale, translation and time shift invariance for the presented video classification approach. Experimental results have shown that the approach is most widely robust against these aspects of invariance. If we compare the presented method to the related work introduced in section 6.2, it becomes apparent that other methods do not comprise all different types of invariance as entirely as the presented one.

Chapter 7

Applying Filters to Trajectories based on Repeating Motion

Filters can be applied to any input signal in order to smoothen or emphasize particular features. In audio processing the input signal is the actual audio signal, in image processing the whole image and in the case of motion the motion trajectory can be considered as an input signal. Since motion trajectories can be distorted by camera jitter, resolution, occlusion or illumination, there has to be an opportunity to compensate these distortions. At this point the application of a filter can reduce noise and emphasize significant parts. Main results and insights of this chapter can be found in research paper [10] by Ayyildiz and Conrad.

7.1 Introduction

This chapter aims to find out, if a filtered centroid trajectory respectively 1D-function leads to a more accurate frequency spectrum after transformation. For this purpose five filters are analyzed and compared to each other. The introduced filters are commonly known and implemented in the area of signal and image processing. A typical task in signal processing is filtering a specific bandwidth. In image processing a filter is often applied to smoothen the image or detect edges.
So in the experimental section filters are added to the classification process. The focus of the test series is on three aspects: First, the effect of filters on the standard test results is assessed. Second, trajectories of occluded motions are filtered and evaluated, since occlusion is one type of signal distortion. As a third aspect the runtime of the system without and with filters is analyzed.

7.2 Related Work

As mentioned before filters considered in this chapter are particularly applied in image processing. Research in [88] and [60] shows that the Lee filter performs better than the average or median filter, when it comes to noise reduction for images. Alsultanny and Shilbayeh analyze a series of filters by applying them to satellite images [4]. Here median, average and low-pass filter lead to similar results. Concerning edge detection filters the so-called *Prewitt filter* works more

accurate than the Laplace filter.
Furthermore, Fanti et al. propose a hybrid model for human action recognition, which is robust to occlusion [36]. This model is based on position, velocity and appearance of body parts.

7.3 Filters for 1D-functions

In real world videos motions of the same activity are never exactly the same and motion trajectories differ from ideal mathematical functions. Unexpected motions, occluded motion or low recording quality can reduce the clarity of 1D-functions and therefore the system's accuracy. In order to improve the clarity various filters can be applied. Filters can reduce noise, smoothen trajectories or emphasize edges, which mean the change of direction in the case of 1D-functions.

7.3.1 Maximum Filter

A maximum filter substitutes each value of a data sequence by a maximum value inside a predefined radius. Let sequence (a_i) with $a_i \in \mathbb{N}$, $i = 0, \ldots, n$ and let radius $r \in \mathbb{N}$. Further on, we define $N_r(i)$ as the set of neighborhood indices of sequence element a_i:

$$N_r(i) = \{x \mid 0 \leq x \leq n \ \wedge \ i - r \leq x \leq i + r\} \tag{7.1}$$

By these definitions we can compute the maximum value around a_i:

$$max_r(a_i) = \max_{x \in N_r(i)} a_x \tag{7.2}$$

Now applying the maximum filter the new sequence $(q_{i_{max}})$ gives:

$$(q_{i_{max}}) = (max_r(a_0), max_r(a_1), \ldots, max_r(a_n)) \tag{7.3}$$

7.3.2 Median Filter

The median filter substitutes each value of a sequence by a medium value inside a given radius. Again we consider sequence (a_i) with $a_i \in \mathbb{N}$ and $i = 0, \ldots, n$, radius $r \in \mathbb{N}$ and $N_r(i)$. For each value a_i we compute a sorted subsequence $(s_j) = (s_1, s_2, \ldots, s_m)$ inside radius r, where again $N_r(i)$ determines the indices neighborhood. For m as the length of (s_j) we define:

$$med_r(a_i) = \begin{cases} \frac{1}{2}\left(s_{\frac{m}{2}} + s_{\frac{m}{2}+1}\right), & \texttt{if m even} \\ s_{\frac{m+1}{2}}, & \texttt{if m odd} \end{cases} \tag{7.4}$$

For (a_i) the usage of a median filter results in $(q_{i_{med}})$:

$$(q_{i_{med}}) = (med_r(a_0), med_r(a_1), \ldots, med_r(a_n)) \tag{7.5}$$

7.3.3 Average Filter

By applying the average filter each value of a sequence is replaced by the average of all values inside radius $r \in \mathbb{N}$. For sequence (a_i) and $N_r(i)$ as the indices neighborhood we replace each value a_i as follows:

$$avg_r(a_i) = \frac{\sum_{x \in N_r(i)} a_x}{|N_r(i)|} \tag{7.6}$$

Hence, we formulate sequence $(q_{i_{avg}})$ as:

$$(q_{i_{avg}}) = (avg_r(a_0), avg_r(a_1), \ldots, avg_r(a_n)) \tag{7.7}$$

7.3.4 Lee Filter

J. S. Lee proposes a statistical filter for digital images [54]. Lee assumes that each image contains natural noise, which can be removed pixelwise. Let σ^2 the variance inside radius r, δ a predefined noise energy and $\sigma^2 < \delta$, then a pixel is replaced by the average inside r. For $\sigma^2 > \delta$ the original value is replaced by another functional value: A high variance σ^2 means that the original value stays almost the same, because it is significant. Lee's filter can also be applied to 1D-functions. For sequence (a_i), radius r and $\beta = max(\frac{\sigma^2 - \delta}{\sigma^2}, 0)$ with $\beta \in \mathbb{R}^+$ we define the Lee filter as:

$$lee_r(a_i) = \beta \cdot a_i + (1 - \beta) \cdot avg_r(a_i) \tag{7.8}$$

So for the new, filtered sequence $(q_{i_{lee}})$ we receive:

$$(q_{i_{lee}}) = (lee_r(a_0), lee_r(a_1), \ldots, lee_r(a_n)) \tag{7.9}$$

7.3.5 Laplace Filter

A Laplace filter is usually utilized for signal and image processing in order to emphasize edges [89]. It is based on the *Laplace operator*, which simply means the second derivative in the context of 1D-functions. Hence, 0 as the second derivate points to a local minimum or maximum. This again gives a hint for an edge inside a signal or an image. So the discretization of the second partial derivative results in:

$$\begin{aligned} \Delta f(i) &= \frac{\partial^2 f(i)}{\partial i^2} \\ &\approx \frac{\partial (f(i+1) - f(i))}{\partial i} \\ &\approx f(i+1) - f(i) - (f(i) - f(i-1)) \\ &= f(i+1) - 2 \cdot f(i) + f(i-1) \end{aligned} \tag{7.10}$$

Consequently, the Laplace operator can be described as a convolution matrix.

$$D_i^2 = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \tag{7.11}$$

An extension of equation (7.10) allows determining edges with varying properties.

$$\Delta f_{r,t}(i) = (f(i+r) - 2 \cdot f(i) + f(i-r))^t \tag{7.12}$$

Variable $r \in \mathbb{N}$ extends or reduces the radius for the local minimum and maximum search. Parameter $t \in \mathbb{N}$ has a further influence on the filtering process. For instance, $t = 2$ leads to only positive results.
Let (a_i) with $a_i \in \mathbb{N}$, $i = 0, \ldots, n$ and $f(i) = a_i$, where $\Delta f_{r,t}(i)$ is undefined for $(i-r) < 0$ or $(i+r) > n$. Now by these preconditions a Laplace filtered sequence $(q_{i_{lpc}})$ based on equation (7.12) can be determined:

$$(q_{i_{lpc}}) = (\Delta f_{r,t}(0), \Delta f_{r,t}(1), \ldots, \Delta f_{r,t}(n)) \tag{7.13}$$

7.4 Experiments

This section focuses on accuracy and runtime performance of the presented system with respect to the filters introduced in section 7.3.

7.4.1 Motion Trajectory Filtering and Transformation

Figure 7.1 shows filtered example 1D-functions on the left and corresponding transforms on the right side. Moreover, the basic 1D-function stems from a person's motion while using a wrench. Particularly the charts in figure 7.1 plot x-axis coordinates of centroids and capture the main motion. It is obvious that the 1D-functions correspond to the left-right and right-left movements. Transforming these 1D-functions by fast Fourier transform (FFT) results in a frequency domain with peaks at 13 and 27. The first amplitude peak at 13 corresponds to the number of left-right movements. In addition, the second peak at 27 arises from a slight centroid movement along the x-axis between two repetitions. This typical centroid movement results from the overall body motion.
Without a filter the spatio-temporal motion trajectory has many highs and lows inside a small time frame. If we consider maximum, median or average filter, these highs and lows disappear and the original chart appears smoothed. In addition, maximum and medium filter lead on to edged charts. For each filter the corresponding high frequency domain has lower amplitudes than the original high frequency domain without filter usage. Especially the average filter reduces amplitudes of the high frequency ranges. However, Lee filter smoothens only parts of the 1D-function, which are below a predefined noise level. Other parts with strong movements even inside small time frames stay nearly unmodified. So only high frequency amplitudes belonging to noisy parts are reduced.
The last chart in figure 7.1 shows the transform for Laplace filter. Frequency 27 is emphasized strongly, because corresponding edges in the 1D-function are emphasized. By using a small or large radius it is even possible to focus on high frequency or low frequency domains, respectively.

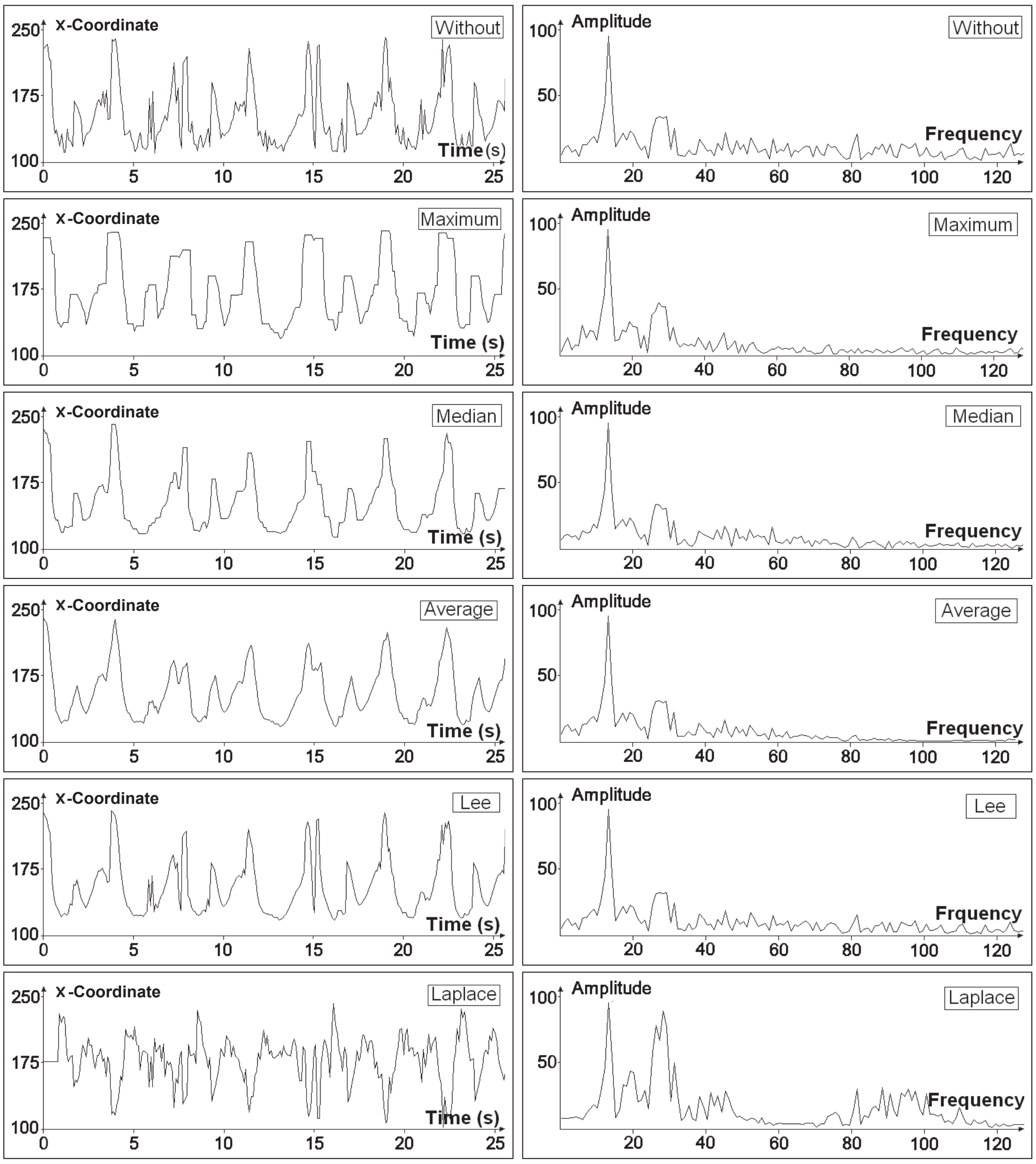

Figure 7.1: Left: wrench handling 1D-function with or without filtering. Right: corresponding transform.

7.4.2 Motion Occlusion

Figure 7.2 illustrates how occlusion changes motion detection pictures for video scenes. A planing video, with the main motion taking place along the horizontal axis, is occluded by a vertical bar. The occluded motion area is not visible inside the motion detection picture and therefore its image moment and depending 1D-functions change. We adjust the alignment and the width of the bar manually for each class in order to achieve a maximal distraction of the motion centroid. This means the bar has always a relative thickness to the main motion area as shown in figure 7.2 and furthermore that this bar is always in the middle of the motion.

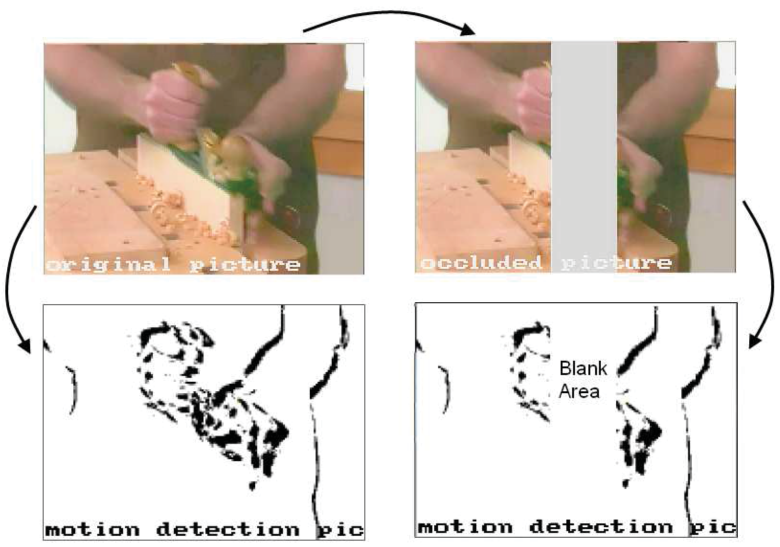

Figure 7.2: Regions with movement for an occluded planing video

7.4.3 Filter Accuracies

Table 7.1 shows resulting accuracies for different 1D-functions and filters. Additionally, we check the same video classes with occlusion. Here the purpose is to find out how occlusion affects the classification process and how far filter can balance out irregularities caused by occlusion.

Filter	**None**	**Maximum**	**Median**	**Average**	**Lee**	**Laplace**
Own Videos						
Direction	0.89	0.87	**0.86**	**0.92**	0.89	**0.86**
Location	0.81	**0.72**	0.73	0.73	0.81	**0.84**
Speed	0.48	0.45	**0.37**	**0.49**	0.47	0.44
Own Videos with Occlusion						
Direction	0.70	0.71	0.73	0.75	**0.81**	0.71
Location	0.72	**0.67**	0.69	**0.67**	**0.73**	0.68
Speed	0.35	0.39	0.35	**0.43**	0.36	**0.33**
External Videos						
Direction	0.28	0.22	0.27	**0.20**	0.27	0.26
Location	0.37	0.29	0.36	0.33	**0.39**	**0.26**
Speed	0.23	**0.17**	0.22	0.21	0.23	0.22
External Videos with Occlusion						
Direction	0.25	0.21	0.24	**0.16**	0.18	0.25
Location	0.37	0.32	0.34	0.33	0.37	**0.26**
Speed	0.21	**0.25**	0.21	0.21	**0.25**	**0.18**

Table 7.1: Overall accuracies for different filter types and 1D-functions

The experimental results in table 7.1 depict that occlusion decreases accuracies, but on the whole the system is still able to classify videos properly. Furthermore, for each 1D-function of own videos there is at least one filter type, that increases the accuracy. Especially for occluded videos classified by directional motion data

we measure a significant accuracy increase. In this case Lee filter raises the accuracy from 0.70 to 0.81.
For occluded videos and 1D-functions derived by the speed of image moments there is a further significant increase. Here the average filter increases the accuracy from 0.35 to 0.43. With respect to external video data there are only three cases with an accuracy improvement. External videos contain more irregular motions, which again means that for instance the maximum filter substitutes values by maximal noise values and increases therefore the number of false classifications. Moreover, the Laplace filter emphasizes noise and the average filter reduces important high frequency amplitudes, which are typical for some external videos.
An overall comparison of all filters leads to the result that the Lee filter is the most accurate filter for repeating motion based video classification. Accuracy increases can be strong and decreases are slight. Here the selective noise reduction seems to be effective. On the other hand the Laplace filter tends to increase noise. Hence, almost all experimental results show up accuracy decreases. Besides, the average filter works only for videos containing clear and smooth motion.

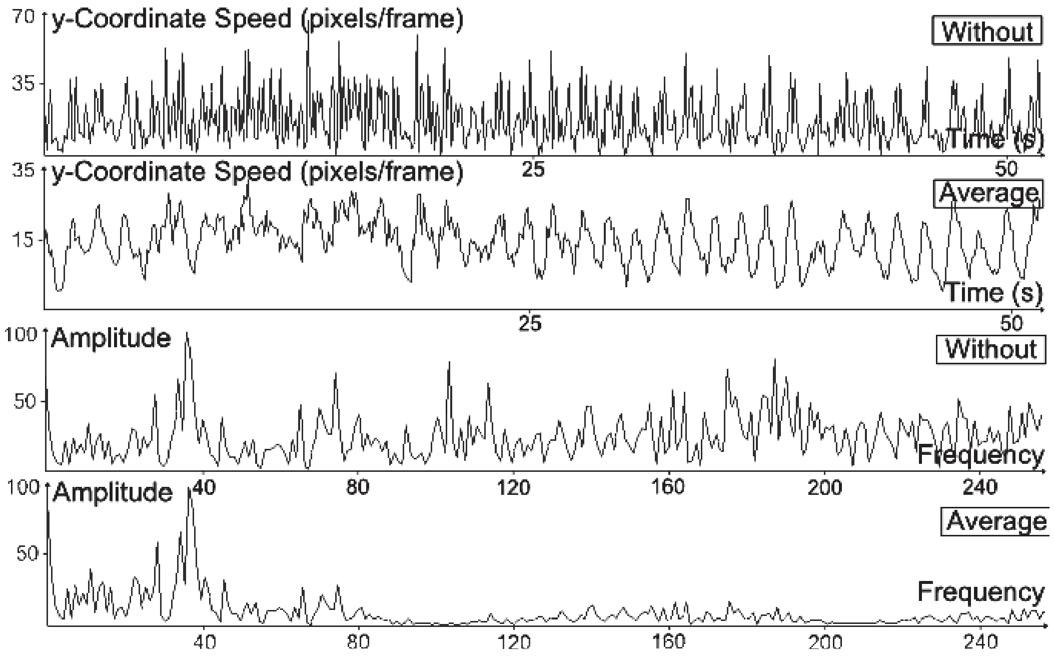

Figure 7.3: Average filter applied to a 1D-function for a hammering video. The foreground motion is occluded and the 1D-function is based on the centroid's speed.

Table 7.1 shows that the Lee filter raises accuracy by 0.11 for directional centroid data of own and occluded videos. Average filter raises accuracy by 0.08 for 1D-functions based on the centroid's speed. By contrast 1D-functions based on the centroid's location do not show any remarkable accuracy raise by applying filters. The reason is that an occlusion influences location based 1D-functions in various ways. Different parts of the frequency domain can be emphasized or declined, whereby filters cannot compensate these changes. Beyond that the location based 1D-functions are the most robust ones, because an occlusion has a minor effect on the overall motion trajectory. As a result there is less space for

accuracy improvement by filtering.
By adding occlusion to video frames the centroid's speed is often raised, when it passes the occluded area. But at the same time the 1D-function becomes noisier. Thus, on the one hand we detect clear highs and lows, on the other hand already noisy 1D-functions based on the centroid's speed become even noisier. In this special case the average filter can reduce the noise as shown in figure 7.3. The transform of a 1D-function based on the centroid's speed for an occluded hammering video gives a noisy frequency domain. There is a large number of amplitude peaks along the whole frequency axis. Determining main frequencies is not possible. Once we smooth the 1D-function by the average filter, the hammering motion becomes much clearer and at the same time high frequency noise disappears. As a consequence, the main frequencies become apparent and can be used for feature selection.
Occlusion weakens the clarity of motion; consequently the centroid direction becomes also noisy. Most often this noise stays below a certain amplitude value, so that the Lee filter can remove exactly this specific noise type. This improvement becomes even more apparent, if the original movement without occlusion was wide and clear. In figure 7.4 classes paint roller, plane and wrench confirm this behavior. Since we consider 10 classes with 20 videos, the maximal number of proper classifications is 20 for each class.

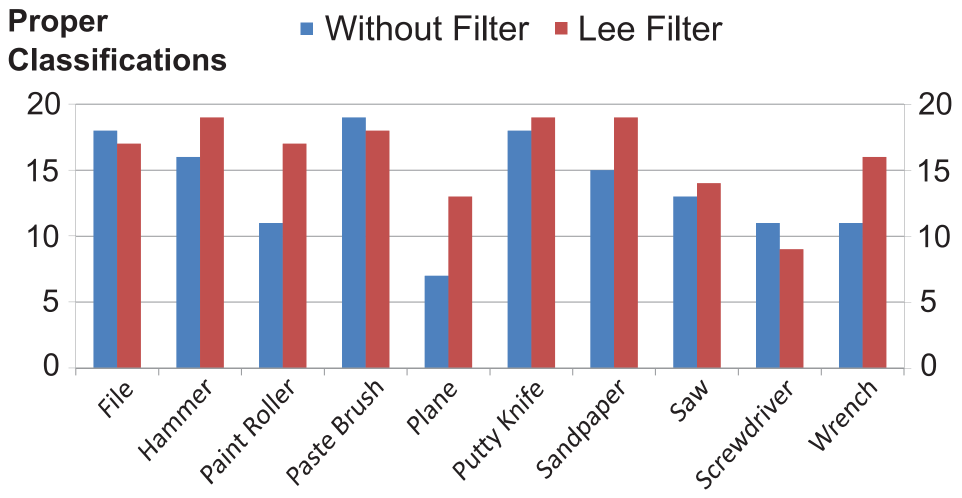

Figure 7.4: Number of proper classifications for own videos with occlusion

Concluding we can state that filtering 1D-functions can improve accuracy in some cases, but on the whole filters reduce the system's accuracy. They reduce the information content or emphasize noise for motion trajectories, so that the resulting feature vectors cannot be assigned properly.

7.4.4 Runtime Analysis

Figure 7.5 shows runtime results for each introduced filter. The system assigns 1000 videos to one out of 10 classes containing home improvement video data (see figure 7.4), whereby the 200 videos covering database is reused five times. Moreover, the filter radius is set to 10. Depicted filter runtimes are averages of five separate test iterations. Averaging is necessary, since runtime differences are marginal and system operations can influence the runtime.

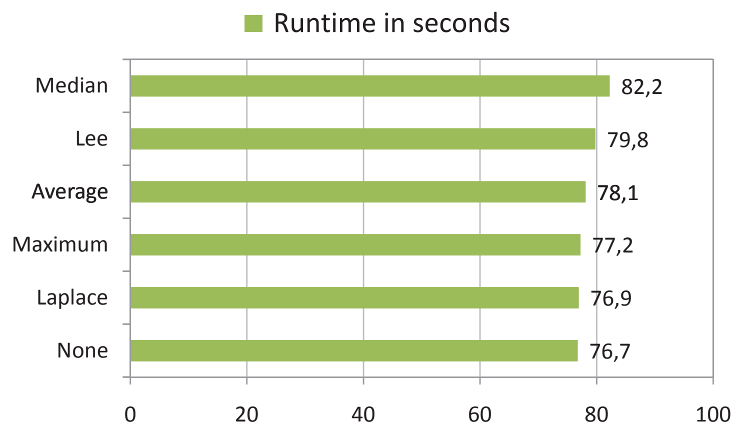

Figure 7.5: System's runtime with different filters

Figure 7.5 shows up small runtime increases when filters are applied. The standard classification without filter takes 76.7 seconds for 1000 videos. Applying Laplace, maximum or average filter the runtime increase stays below 2 seconds. These three filters have got similar algorithmic setups. Utilizing Lee filter the runtime is 79.8 seconds and therefore longer than the runtime for the previous three filters. Due to additional operations in order to find out the variance, Lee filter requires more runtime. Further, we measure a maximum runtime at 82.2 seconds for the median filter. The median filter has to arrange data values in order to find a medium data value. Sorting data values needs more operations than calculating the variance. Thus, the median filter takes more runtime than the Lee filter.
Experiments show that for the choice of the most performant filter the runtime aspect can be neglected, since runtime differences are very small. For instance, applying the median filter increases the runtime for 5.5 seconds. Only when extraordinary huge datasets are going to be classified this aspect plays a role. Apart from raising the number of videos it is also possible to classify sliding time windows inside videos. This again would increase the number of classifications clearly.

7.5 Conclusion

In this chapter the experimental part focused on filtering 1D-functions in order to receive more decisive frequency domains. Test results show that the Lee filter performs best, since this filter smoothens only noisy parts of a 1D-function. Hence, these results correspond to the experimental results for image processing in [88] and [60]. The maximum and Laplace filter reduce the system's accuracy in most cases, because either high frequencies are smoothed too strongly or noisy parts are emphasized, respectively. The maximum, median and average filter are all smoothing respectively low-pass filters and reduce amplitudes for high frequencies. So here again we find information reduction.
The main difference between the Lee filter and the other four filters is, that the parameter for the noise energy can be tuned precisely. Considering the Lee filter column in table 7.1 for some test cases we set a high noise energy parameter, but for the most part the parameter is set to very low values. So for these cases the

Lee filter has a minor impact on the 1D-function and the accuracy level does not change. As a result the Lee filter improves only 1D-functions containing clear noise, otherwise the 1D-function stays almost unmodified and there is less accuracy decrease.
Main results from table 7.1 hold for filters even with varying modules for background subtraction, image moments, transforms and classifiers. But there is one module, which has a strong impact on these results: The feature extraction module. Filter analysis in this chapter is performed by AAFIs. If we apply main frequencies, results will change. On the one hand feature vectors based on AAFIs are more sensitive to filters, because filters modify the whole frequency domain and AAFIs represent the whole frequency domain. On the other hand main frequencies are not affected by noise amplification as long as the noise amplitude is below the main frequencies' amplitude. Thus, it possible that the Laplace filter or smoothing filters work more accurate combined with feature vectors based on main frequencies.
A further aspect of this chapter is runtime analysis. Experiments show that the Lee filter needs more operations than the maximum, average or Laplace filter, but less operations than the median filter. But on the whole runtime differences are very small, so for a real world application this aspect can be ignored.
Applying filters to 1D-functions can improve the system's accuracy in some cases, but in general the accuracy is decreased. Even the Lee filter improves accuracies only in special cases. But there are still edge detection filters as the Prewitt filter or noise removing filters as the harmonic mean filter, which have to be analyzed and could reveal more accurate test results.

Chapter 8

Analyzing Transformation of 1D-Functions

In this chapter the focus is on transforms applied to 1D-functions. Different transforms yield differing frequency domains, which are the basis for AAFIs. So transforms have a direct effect on the overall accuracy of the classification system. Next experiments bring out the optimal transform for a motion frequency based approach by comparing five different transforms. Major parts of this chapter have been published by Ayyildiz and Conrad in [7].

8.1 Introduction

A transform is capable of converting a discrete input signal from its time domain to its frequency domain. So if there is periodic progression for an input signal, the frequency domain reveals this. For the present thesis 1D-functions represent the input signal and transforms make their dominating frequencies apparent. In the experiments of the previous chapters the 1D-functions were transformed to their frequency domain by the fast Fourier transform. Nevertheless, there are more transforms available, which need further investigation in the context of frequency features. So in this chapter transforms such as the fast Fourier transform (FFT), fast cosine transform (FCT), fast sine transform (FST), fast Haar wavelet transform (FWT) and fast Hadamard transform (FHT) are researched and compared to each other. Moreover, this chapter aims to bring out strong points and limits of the different transforms with respect to their accuracy and runtime.
Next section 8.2 will give an overview of the related work. Then section 8.3 defines formally the working methods of the analyzed transforms. In section 8.4 the presented transforms are evaluated experimentally. Last section 8.5 concludes the major insights.

8.2 Related Work

Some of the previously introduced related work apply transforms. For instance, Cheng et al. analyze sports videos by a modified FFT in [25]. Furthermore, Davis and Cutler obtain the main frequency of a motion by transforming measured self-similarity via Fourier transform [30]. In [62] Meng et al. consider the

frequency peaks of transformed MLD curves as features of cyclic motion. Here again transformation takes place via Fourier coefficients. Moreover, in paper [85] Tsai et al. determine optimal MLD curve patterns by FFT.
Direct comparison of transforms is provided by image processing research: Kekre et al. compare discrete cosine (DCT), Walsh-Hadamard (WHT), Haar-Wavelet (HWT) and Kekre's transform for an image retrieval system [52]. Here images are transformed before the feature extraction stage. DCT, WHT and HWT have a maximum accuracy at 0.41 and Kekre's transform at 0.42. Hunt and Mukundan compare HWT, WHT, DTT (discrete Tchebichef) and DCT [50]. The analysis shows that the HWT works best for image compression using orthogonal basis functions. WHT is positioned at the fourth place with a marginal difference to the other transforms.
Summarizing it can be stated, that the FFT and DCT are the most commonly used transforms in the image and video retrieval context. For the presented frequency domain based classification system related research, which directly compares all of the above transforms, could not be identified.

8.3 Transformation of 1D-Functions

Transforms explained in this section are used during the experimental stage. All introduced equations transform time discrete and periodic signals to their frequency domain.

8.3.1 Fast Fourier Transform

The most important transform in science is the Fourier transform. Since 1D-functions are finite and periodic, we utilize the discrete Fourier transform (DFT) [40]. Let $\hat{f} = (\hat{f}_0, ..., \hat{f}_{N-1}) \in \mathbb{C}^N$ the discrete Fourier transform of a complex vector $x = (x_0, ..., x_{N-1}) \in \mathbb{C}^N$, which is constituted by the interpolation of measured values. The transform's size is set to $N = 2n$. Then formula (8.1) for $m = 0, \ldots, 2n-1$ results.

$$\hat{f}_m = \sum_{k=0}^{2n-1} x_k \, e^{-\frac{2\pi i}{2n} mk} \tag{8.1}$$

Here $\hat{f}_m$ is also called Fourier coefficient or Fourier component. Realizing the algorithm of Cooley and Tukey we get the fast Fourier transform (FFT) [27]. FFT works with DFTs of half length and consequently needs the quarter of complex calculations. This transform implements a divide-and-conquer method. Depending on the length of the measured value sequence the method can be applied multiple times, whereby the runtime amounts to $\mathcal{O}(n \cdot log(n))$.

8.3.2 Fast Cosine Transform

The fast Cosine transform (FCT) is based up on the discrete Cosine transform (DCT) [2] and works similar to FFT. By contrast to DFT the DCT calculation takes place only with real coefficients. A finite data sequence is expressed as a finite sum of cosine functions. Since data values of discrete transforms are finite, an extension before and after this sequence is presumed. The DCT computation is realized by an even extension.

Definition 1 (Even Function)**.** *A function f with a domain D is called an even function, if $\forall x \in D\ f(x) = f(-x)$.*

Extending a sequence by even and odd functions allows four combinations (see section 8.3.3 for odd functions). Furthermore, it is possible to extend a sequence directly at a value or between two values. Hence, $4 \cdot 4 = 16$ combinations are possible. Moreover, 8 out of 16 possible extensions are even at the beginning. Sequences with this extension type are used for cosine transforms and the other 8 extensions are considered for sine transforms. We use DCT-I with an even extension around x_0 at the beginning and around x_{N-1} at the end. Let $x_n = (x_0, \dots, x_{N-1}) \Rightarrow X_n = (X_0, \dots, X_{N-1})$ a real value mapping and $k = 0, \dots, N-1$, then DCT-I is defined as:

$$X_k = \frac{1}{2}(x_0 + (-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n \cos\left[\frac{\pi}{N-1} nk\right] \tag{8.2}$$

By pre- and post-processing steps the DCT can be combined with FFT and solved with $\mathcal{O}(N \cdot log(N))$ operations.

8.3.3 Fast Sine Transform

Fast Sine transform (FST) is based on the discrete Sine transform (DST) [59], which is related to the DCT. Now DST transforms a sequence of values by sine functions. Further on, DST extends the beginning of a finite sequence by odd functions. Here again 8 different extensions are possible.

Definition 2 (Odd Function)**.** *A function f with domain D is called an odd function, if $\forall x \in D\ f(-x) = -f(x)$.*

For the experimental stage we use DST-I among all 8 extensions. Its boundary values around x_{-1} at the beginning and around x_N at the ending are odd. If a real value mapping $x_n = (x_0, \dots, x_{N-1}) \Rightarrow X_n = (X_0, \dots, X_{N-1})$ and $k = 0, \dots, N-1$ is given, then we receive

$$X_k = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{N+1}(n+1)(k+1)\right] \tag{8.3}$$

The FST is computed analogous to FCT by combining the incoming signal with equation (8.3) and FFT. Here again runtime amounts to $\mathcal{O}(N \cdot log(N))$.

8.3.4 Fast Haar-Wavelet Transform

Now the Haar-Wavelet is defined [18]:

$$\psi(x) = \begin{cases} 1 & \texttt{if } 0 \leq x < 1/2 \\ -1 & \texttt{if } 1/2 \leq x < 1 \\ 0 & \texttt{else} \end{cases} \tag{8.4}$$

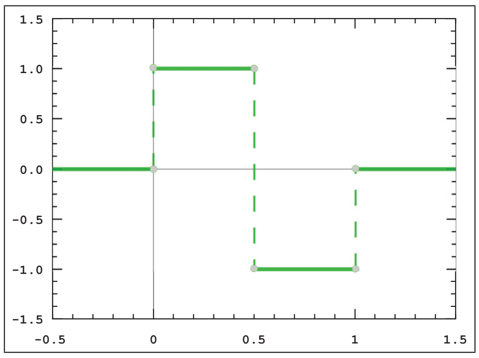

Figure 8.1: Haar-Wavelet

Figure 8.1 illustrates the Haar-Wavelet plot. For $j \in \mathbb{N}, 0 \leq k \leq 2^j - 1$ it is feasible to swage and to shift the Haar wavelet inside interval [0,1]:

$$\psi_{j,k}(x) = \psi(2^j \cdot x - k) \tag{8.5}$$

Functions $\psi(x)$ and $\psi_{j,k}(x)$ are orthogonal inside interval [0,1]. These assumptions lead to Haar matrices of different orders [18]. In equation (8.6), for instance, a second order Haar transform matrix H is presented.

$$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{8.6}$$

Considering fast Haar wavelet transform (FWT) a finite discrete signal is transformed into a sequence of two dimensional vectors. This process is the so-called *polyphase partition*. Hereby the data sequence is joined pairwise without changing the order of its elements.

$$f_p = \left(\ldots, \begin{pmatrix} f_{-2} \\ f_{-1} \end{pmatrix}, \begin{pmatrix} f_0 \\ f_1 \end{pmatrix}, \begin{pmatrix} f_2 \\ f_3 \end{pmatrix}, \ldots \right) \tag{8.7}$$

Each element of this new sequence is multiplied with transform matrix H.

$$\begin{pmatrix} s \\ d \end{pmatrix} := H \cdot f_p = \left(\ldots, \begin{pmatrix} s_{-1} \\ d_{-1} \end{pmatrix}, \begin{pmatrix} s_0 \\ d_0 \end{pmatrix}, \begin{pmatrix} s_1 \\ d_1 \end{pmatrix}, \ldots \right) \tag{8.8}$$

Then we receive $s_k = \frac{f_{2k}+f_{2k+1}}{\sqrt{2}}$ and $d_k = \frac{f_{2k}-f_{2k+1}}{\sqrt{2}}$. Moreover, the input signal can be partitioned into bigger blocks, if a corresponding orthogonal Haar matrix with entries $1/\sqrt{s}$ in its first line is generated. The FWT conforms to complexity class $\mathcal{O}(N \cdot log(N))$.

8.3.5 Fast Hadamard Transform

The Hadamard transform is an orthogonal, symmetric, involutional und linear function, which is realized by $N = 2^m$ input values [76]. Its calculation takes place with a $2^m \times 2^m$ Hadamard matrix H_m. This matrix transforms a 2^m sized real sequence x_n into a 2^m sized sequence X_k by multiplying H_m with x_n. For $H_0 = 1$ and $m > 0$ the Hadamard matrix is defined recursively as follows:

$$H_m = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{pmatrix} \tag{8.9}$$

Example for $m = 2$:

$$H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{8.10}$$

$$H_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{pmatrix} \tag{8.11}$$

$$H_2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{8.12}$$

Factor $\frac{1}{\sqrt{2}}$ is used for the purpose of normalization. Let $k, n \in \mathbb{N}_0$ indices for the matrix entries with k_j and n_j as their binary digits:

$$\begin{aligned} k &= k_{m-1}2^{m-1} + k_{m-2}2^{m-2} + \cdots + k_1 2 + k_0 \\ n &= n_{m-1}2^{m-1} + n_{m-2}2^{m-2} + \cdots + n_1 2 + n_0 \end{aligned} \tag{8.13}$$

Example for $m = 2$, $k = 1$ and $n = 3$:

$$\begin{aligned} 1 &= 01 = 0 + 1 \\ 3 &= 11 = 2 + 1 \end{aligned} \tag{8.14}$$

Hence, each entry (k, n) of a Hadamard matrix is defined by equation (8.15).

$$(H_m)_{k,n} = \frac{1}{2^{m/2}}(-1)^{\sum_j k_j n_j} \tag{8.15}$$

The Hadamard transform involves $\mathcal{O}(N^2)$ operations. Fast Hadamard transform (FHT) requires $\mathcal{O}(N \cdot log(N))$ calculation steps [37]. It also uses a divide and conquer algorithm in order to break down Hadamard transforms of size N into two smaller Hadamard transforms of size $N/2$ recursively.

8.4 Experiments

In this section we evaluate accuracy and runtime performance of the presented system concerning different transforms. For FFT, FCT, FST and FHT the *Apache Commons Mathematics Library* is applied [39]. The FWT is performed by the usage of the *JSci API* [20].

8.4.1 Motion Trajectory Transformation

Next in figure 8.2 an example 1D-function is illustrated. It regards to the y-axis coordinate of centroids and captures the main motion inside an external video. The motion in this video arises from a person using a paint roller. Furthermore, the 1D-function corresponds to the movement of the person, since the centroid moves from bottom to top and vice versa. Plots in figure 8.3 present frequency spectra of the 1D-function from figure 8.2. Each spectrum results from one of five transforms explained in section 8.3.

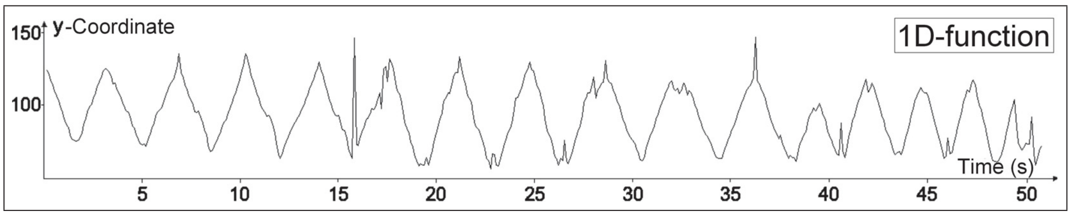

Figure 8.2: 1D-function of a person using a paint roller

At first glance it becomes apparent, that all transforms except FHT have a high amplitude around a frequency of 12. This frequency corresponds to the main movement of the paint roller in figure 8.2. FCT presents frequencies more detailed and partially with higher amplitudes, but roughly frequencies are identical with FFT. Now FST shifts FFT and FCT highs to the left, whereas FWT has high amplitudes at different frequencies. There is even a peak at a frequency of 165.

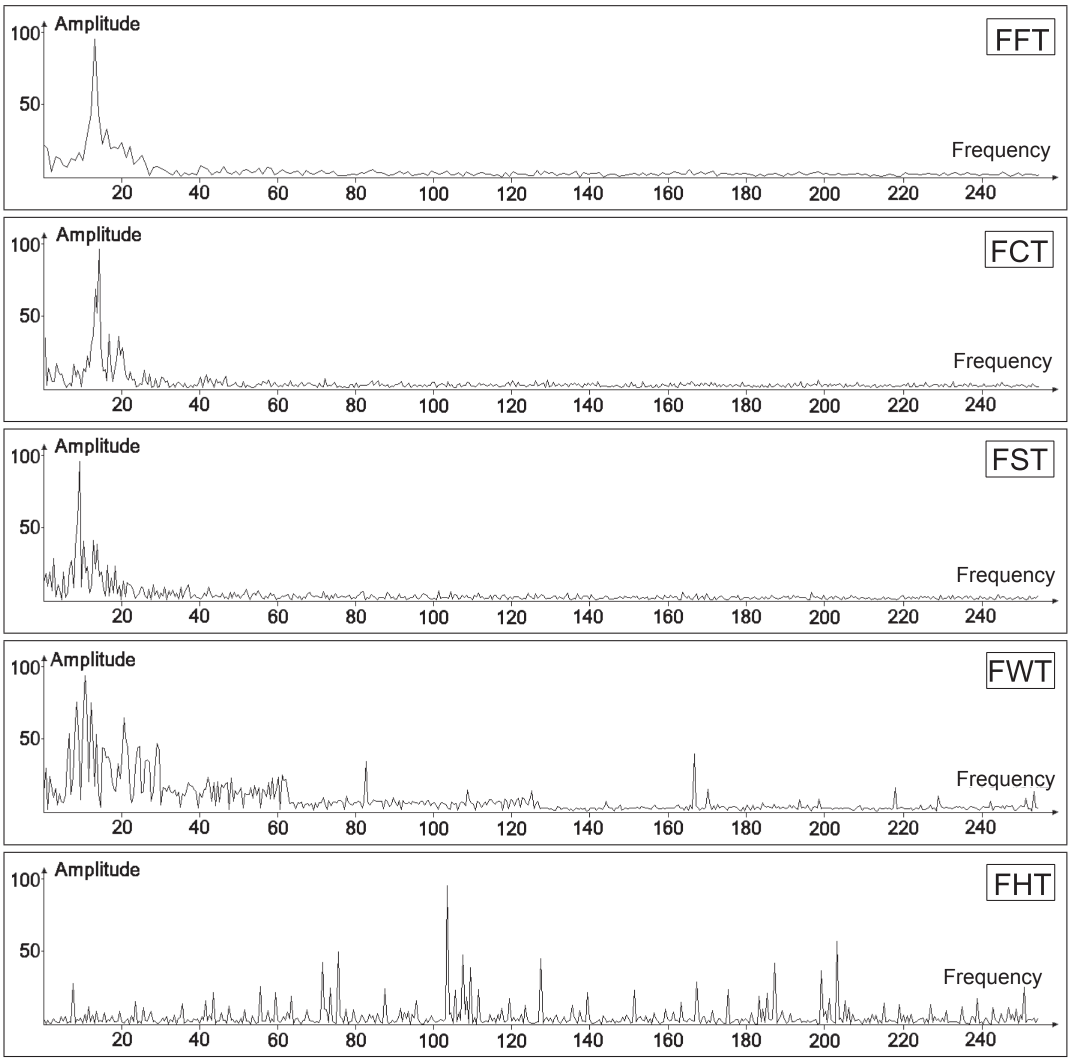

Figure 8.3: Transforms of paint roller 1D-function

8.4.2 Accuracy of Transforms

Next tables 8.1 and 8.2 show accuracies for own and external videos, respectively. Overall accuracies are not the average of class accuracies, but the ratio of cor-

Own Videos					
Class ACC	**FFT**	**FCT**	**FST**	**FWT**	**FHT**
File	0.85	0.85	0.70	0.35	0.25
Hammer	0.84	0.83	0.79	1.00	0.25
Paint Roller	1.00	1.00	0.80	0.55	0.45
Paste Brush	0.90	0.85	0.85	1.00	0.40
Plane	0.55	0.75	0.65	0.70	0.11
Putty Knife	0.95	0.95	1.00	0.80	0.40
Sandpaper	0.88	0.56	0.53	0.75	0.05
Saw	0.75	0.65	0.80	0.90	0.20
Screwdriver	0.95	1.00	1.00	0.70	0.75
Wrench	1.00	1.00	0.85	0.95	0.05
Overall ACC	**0.87**	**0.85**	**0.80**	**0.77**	**0.29**

Table 8.1: Transform accuracies for own videos

rect classifications to the total number of classifications. Now own videos are assigned to classes by direction information, whereas external videos are assigned using location information of image moments (see section 2.2.3). External video data contains more irregular motion, so for this case direction information is less reliable. Tables show overall and single class accuracies. In table 8.1 the transform with the maximum accuracy 0.87 is FFT. The transforms FCT, FST and FWT also achieve high accuracies. As shown in figure 8.3, here again FHT differs clearly from the other transforms. Its accuracy marks the minimum with 0.29. Considering the class accuracies for each transform we find high accuracies for home improvement activities with clear motion and an unique frequency feature.

External Videos					
Class ACC	**FFT**	**FCT**	**FST**	**FWT**	**FHT**
File	0.00	0.00	0.00	0.75	0.00
Hammer	0.00	0.00	0.22	0.00	0.11
Paint Roller	0.69	0.92	0.50	0.79	0.15
Paste Brush	0.00	0.25	0.00	0.25	0.25
Plane	0.00	0.00	0.00	0.00	0.00
Putty Knife	0.57	0.55	0.52	0.43	0.04
Sandpaper	0.36	0.18	0.18	0.45	0.18
Saw	0.33	0.22	0.22	0.00	0.10
Screwdriver	0.57	0.57	0.14	0.00	0.43
Wrench	0.50	0.63	0.13	0.38	0.38
Overall ACC	**0.40**	**0.41**	**0.27**	**0.32**	**0.14**

Table 8.2: Transform accuracies for external videos

For external videos FCT leads to a maximum accuracy of 0.41. Here again FHT has the lowest accuracy with 0.14.
If we compare results for own and external video scenes, as shown in figure 8.4,

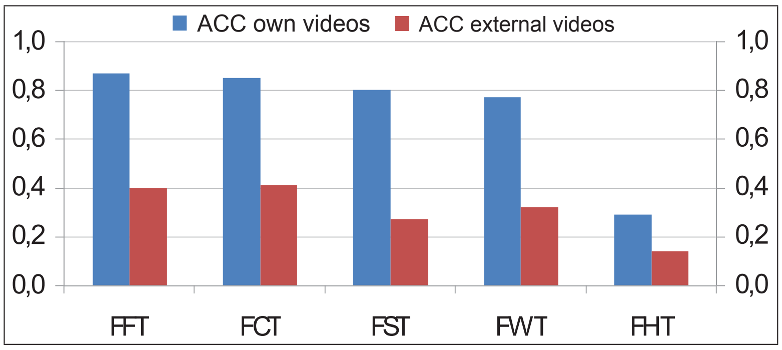

Figure 8.4: Overall accuracies for own and external clips

it becomes apparent that own videos are assigned twice as accurate as external videos. Moreover, it can be stated, that FFT and FCT work best for our purpose. FST and FWT result in moderate accuracies and FHT is the weakest approach. FFT and FCT plot similar frequency spectra. FST is also similar to FFT and FCT, but there are accuracy decreasing differences. Due to the fact that FST uses odd extensions at the beginning and at the end of a data sequence, we detect more high amplitudes for low frequencies. Furthermore, comparing FST to FFT amplitudes are shifted to the low frequency range a bit. On the whole the low frequency range is not as reliable as the low frequency range of FFT or FCT. As explained before wavelets are localized in both time and frequency. This means the FWT is capable to represent discontinuities and sharp peaks of a 1D-function. So the frequency domain reveals additional and decisive information, which could even increase the accuracy. Nevertheless, the presented repeating motion based approach is impaired by representing non-stationary movements inside the frequency spectrum. This result was open before the experimental evaluation. Hence, the FFT, which is localized only in frequency, works more accurate than the FWT, which is localized in time and frequency. FHT is extremely sensitive to time shift and not invariant under circular time shift of the input data. Even a minimal difference of the starting point of the dataset changes the frequency domain drastically. Thus, FHT is not useful for repeating motion based video classification.

8.4.3 Runtime Analysis

This subsection concerns the runtime of the presented approach with regard to different transforms. The bar chart in figure 8.5 shows runtimes for fast Fourier, fast Cosine, fast Sine, fast Hadamard and fast Wavelet transform. The results are closely together, since all transforms belong to the same complexity class.
When classifying 1000 videos by FFT the system runs 76.9 seconds. This means by FFT the system works faster than by any other transform. FCT and FST need 77.7 seconds and 79.0 seconds, respectively. The two transforms modify a sequence resulting from the FFT, so their runtime is the sum of FFT's runtime and additional operations. With 78.3 seconds FHT needs also more runtime than FFT. We detect the longest runtime for FWT with 79.3 seconds. Here the polyphase partition and the following transform-matrix multiplication with data elements are more expensive than the divide and conquer approach applied by

FFT or FHT.

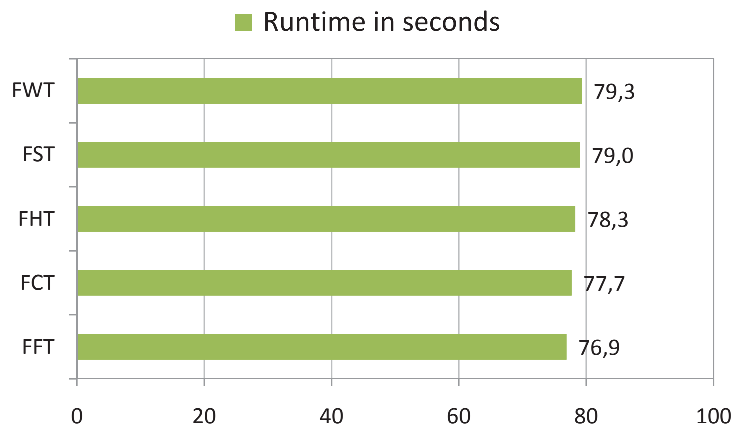

Figure 8.5: System's runtime with different transforms

Runtimes for the transforms are even closer to each other than the filter runtimes in figure 7.5. The runtime difference between FFT and FWT is only 2.4 milliseconds per video. So here again the runtime aspect can be ignored for the choice of the most performant transform. Only for a special classification setup, where a huge quantity of data has to be processed, this aspect can become important.

8.5 Conclusion

In this chapter the experimental stage discussed the transform process of 1D-functions extensively in order to determine the best transform for the presented approach. Results expose that the FFT and the FCT lead on to the best accuracies, where the FFT has a marginal runtime advantage. The FHT brings about the lowest accuracy values and the FWT needs the longest runtime.
So experimental results affirm the wide usage of Fourier and cosine transforms as stated in the related work section.

Chapter 9

Classifier Comparison for Frequency Features based on Repeating Motion

Most of the experiments in this thesis use the RBC as classifier. Since the classifier module is interchangeable and different classifiers handle AAFIs differently, this chapter compares the performance of six classifiers including the RBC. Core topics of this chapter were published in a similar way by Ayyildiz and Conrad in [9].

9.1 Introduction

Even frequency spectra of videos with same class labels can vary clearly. Thus, there is a need for a classifier, which is able to handle such differences. In order to determine the most accurate classifier for a frequency features based approach, the experimental section of this chapter performs tests with different types of classifiers. We can state four major categories for six analyzed classifiers: Let o an object, which is to be assigned to the nearest class. Then we can first categorize the RBC and KNN as classifiers, which utilize objects in the nearer environment of object o for distance calculations. Second, the NCC (next centroid) and ALC (average linkage) use positions of whole classes to calculate a minimal distance for object o. SVM bases on a vector space separating technique and represents the third category. The fourth category is represented by the ANN (artificial neural network), which simulates the decision process inside a biological neural network.
The following sections first present related work, then define the six assessed classifiers formally. Hereafter in the experimental section benefits and disadvantages of these classifiers are discussed in the context of classification accuracy and runtime.

9.2 Related Work

Cheng et al. use a neural network classifier for their main frequency based classification technique [25]. In literature the number of repeating motion and frequency features based research papers is limited. So a direct comparison of classifiers for

this specific research area is hard to find, but there are related research fields, which provide similar analyses.
Some research in computer vision considers direct comparison of classifiers based on motion features. Sobral et al. provide a system for highway traffic video classification [81]. Capturing crowd density and crowd speed of vehicles the system assigns traffic videos to one of three congestion classes. Test series with classifiers KNN, Naive Bayes, SVM and ANN yield accuracies of 0.93, 0.93, 0.93 and 0.95, respectively. The number of classified highway traffic videos is 254. Research paper [75] is about an outdoor personal navigation system, which works computer vision aided. Here Saeedi et al. select features by analyzing video images and position information provided by GPS. After comparing SVM, ANN and KNN classifiers the authors state that SVM works best for their purposes. Accuracies range from 0.50 to 1.00 for different activities like driving, walking, running, using stairs or an elevator. Another work by Glette et al. concerns hand signal classification by muscle contraction EMGs [41]. Here again SVM performs best, KNN shows high accuracies and DT (decision tree) shows the lowest performance. Authors Rocamora and Herrera use features from transformed audio signals to detect the singing voice part in music audio files [74]. In this case classifiers SVM, ANN, KNN and DT achieve accuracies of 0.85, 0.83, 0.77 and 0.73, respectively.

9.3 Defining Classifiers

As the main focus of this chapter is on the evaluation of classifiers, now several classifiers are introduced, which are applied during the classification process.
First of all, we define $C = \{C_1, \dots, C_m\}$ as our set of classes. Each class $C_i \in C$ contains a set of objects, so we define $C_i = \{o_{i_1}, \dots, o_{i_{n_i}}\}$, $C_i \neq \{\}$ and $C_i \cap C_j = \{\}$ for $i \neq j$. The total of all objects in classes is $A = C_1 \cup \dots \cup C_m$. Hence, the total number of objects results in $e = |A| = |C_1 \cup \dots \cup C_m| = \sum_{i=1}^{m} n_i$. Now a set of objects $B = \{o_1, \dots, o_l\} \neq \{\}$ without class labels is considered for classification.

9.3.1 K-Nearest Neighbors

The k-nearest neighbors classifier (KNN) [28] searches k neighbors with a minimal distance to a given object $o_b \in B$ among all objects $o_i \in A$. Object o_b is then assigned to the class with the maximal number of neighbors. The metric and the parameter k used play a central role for the rate of correct classifications. A large k can reduce noise, but it can also involve a wrong neighbor class.

$$class_{knn}(o_b, C) = \arg\max_{C_i \in \mathbf{C}} |C_i \cap n_k(o_b)| \tag{9.1}$$

Equation (9.1) shows the selection method of the KNN classifier. Let $n_k(o_b)$ the set of k-nearest neighbors for o_b. Then $class_{knn}(o_b, C)$ leads to the class with the most elements in $n_k(o_b)$. If several classes have the same maximal number of elements in $n_k(o_b)$, one class is chosen by chance.
Since e objects out of all classes need Euclidian distance calculations and each distance calculation amounts to d iterations (one iteration for each dimension),

the complexity order for distance calculations is $\mathcal{O}(e \cdot d)$. In order to find the k nearest neighbors we use the quicksort algorithm with time complexity order $\mathcal{O}(e \cdot log(e))$. So the overall time complexity order of the applied KNN classifier is $\mathcal{O}(e \cdot (d + log(e)))$.

9.3.2 Average Linkage Classifier

Hierarchical clustering algorithms often use the average linkage distance for distance calculations between clusters [90]. Now we apply this distance for a new classifier by assigning an object to the class with the minimal average linkage distance. Therefore, we call this classifier average linkage classifier (ALC). More precise the ALC computes all Euclidian distances between an object $o_b \in B$ and objects $o_i \in C_i$. Dividing the sum of these distances by the class size $|C_i|$ yields the average linkage distance δ_i between object o_b and class C_i.

$$\delta_i(o_b) = \frac{1}{|C_i|} \sum_{o_i \in C_i} \|o_b - o_i\| \tag{9.2}$$

$$class_{alc}(o_b, C) = \arg\min_{C_i \in \mathbf{C}}(\delta_i(o_b)) \tag{9.3}$$

The ALC assigns object o_b to the class with the minimal distance δ_i among all classes $C_i \in C$. If several classes have the same minimal distance to o_b, one class is chosen at random. ALC's main advantage is that it works without any parameter input. Object classes are given and average linkage distances are normalized by the class size. Its time complexity is of the order $\mathcal{O}(m \cdot n \cdot d) = \mathcal{O}(e \cdot d)$. The implementation is a nested loop, where Euclidian distance calculations need d iterations. These calculations take place for n objects of each class and m classes have to be compared.

9.3.3 Nearest Centroid Classifier

In machine learning a nearest centroid classifier (NCC) assigns an object $o_b \in B$ to the class with the closest centroid μ_i [58]. Just as ALC NCC's main advantage is its independence of any input parameter.

$$\mu_i = \frac{1}{|C_i|} \sum_{o_i \in C_i} o_i \tag{9.4}$$

$$class_{ncc}(o_b, C) = \arg\min_{C_i \in \mathbf{C}} \|\mu_i - o_b\| \tag{9.5}$$

Equation (9.5) predicts class C_i with the nearest centroid to o_b. Centroids for m classes are calculated. Moreover, each centroid calculation depends on n objects in each class and on the dimensionality d of the objects. These centroids can be stored and reused for further classifications. So the time complexity order for the centroid calculations is $\mathcal{O}(m \cdot n \cdot d) = \mathcal{O}(e \cdot d)$. Moreover, each object-centroid distance computation depends again on the dimensionality. Thus, the operations for this step are of the order $\mathcal{O}(m \cdot d)$.

9.3.4 Radius Based Classifier

In section 2.3 we already explained the working method of the radius based classifier (RBC). Now we define this classifier formally. Let object $o_b \in B$ and class $C_i \in C$, then radius ε determines the ε-neighborhood $N_\varepsilon(o_b, C_i)$. This ε-neighborhood encloses all objects of class C_i inside the predefined radius around o_b. RBC is sensitive to the compactness of a class. So the choice of parameter ε is decisive. The distance between objects is measured by Euclidian distance.

$$N_\varepsilon(o_b, C_i) = \{o_s | o_s \in C_i \wedge \|o_b - o_s\| < \varepsilon\} \tag{9.6}$$

Based upon $N_\varepsilon(o_b, C_i)$ we define the distance between an object o_b and a class C_i.

$$\nu_i(o_b) = 1 - \frac{|N_\varepsilon(o_b, C_i)|}{|C_i|} \tag{9.7}$$

$$class_{rbc}(o_b, C) = \arg\min_{C_i \in \mathbf{C}}(\nu_i(o_b)) \tag{9.8}$$

Equation (9.8) gives the class with the minimal distance to o_b among all classes. If there are multiple classes with the same distance, then one class is chosen at random. Now here again the time complexity order is $\mathcal{O}(m \cdot n \cdot d) = \mathcal{O}(e \cdot d)$, because for each class the ε-neighborhood calculation considers all objects of one class.

9.3.5 Support Vector Machine

A support vector machine (SVM) is a classifier, that separates the vector space for a given set of labeled training objects into class spaces [45]. Afterwards, a new object can easily be assigned to a class. So class borders have to be determined. These borders are named *hyperplanes*. A valid separation by hyperplanes is only possible, if the object set is linearly separable. If not, kernel functions are applied in order to describe the hyperplanes in higher dimensional vector spaces. We receive SVM by minimizing error function

$$\frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i \tag{9.9}$$

with restrictions:

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \texttt{ and } \xi_i \geq 0, i = 1, \ldots, m \tag{9.10}$$

Variable w is the normal vector to the hyperplane and index i labels the m training classes. Parameter ξ_i handles inseparable input data. It measures the degree of misclassification of the training data x_i. The objective function (9.9) is then raised by a function, which penalizes non-zero ξ_i. So the optimization becomes a tradeoff between a large margin and a small error penalty. Additionally, capacity constant C regulates that tradeoff. Consider that $y_i \in \pm 1$ are class labels for training objects x_i. Bias b is a further constant that represents the distance of the hyperplane from the origin. Kernel function ϕ transforms input data to the feature space. Now for multi-class classification we use the *one-against-one*

approach: If m is the number of classes, we generate $m(m-1)/2$ models. Each model involves only two classes of training data and assigns an instance to one of these two classes. The class with the maximal number of assignments determines the final classification. SVM's training phase and finding the best kernel or best parameters take much time.
Solving equation 9.9 is generally known as a quadratic programming problem (QP) [86], since a quadratic function with several variables has to be minimized with linear constraints on these variables. QP needs in worst case $\mathcal{O}(t^3)$ operations. Olivier Chapelle explains in [23] that in the case of the SVM classifier the training's time complexity is $\mathcal{O}(max(e,d) \cdot min(e,d)^2)$, where e is the number of training objects and d the dimensonality. The author starts with equation 9.9 and deduces the time complexity in a detailed way.
As we use the RBF (radial basis function) as kernel the complexity order for predicting with SVMs is $\mathcal{O}(m^2 \cdot N_S \cdot d)$. We consider $m(m-1)/2$ models, so m^2 is the first factor. For each model N_S support vectors with the dimensionality d are involved. In order to predict the next class for a test object via RBF kernel, a linear combination of inner products is calculated. This means for each support vector there is a product calculation and this includes also a distance calculation between test object and support vector. A more detailed formalization of the SVM prediction step and its time complexity can be found in [26].

9.3.6 Artificial Neural Network

In this subsection we explain the general structure of an artificial neural network (ANN). The chosen ANN type and its setup for the experiments can be found in subsections 9.4.2 and 9.4.3.

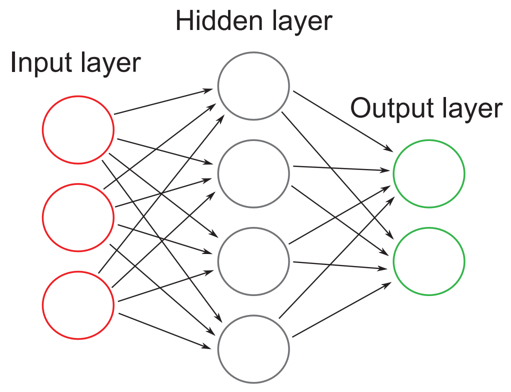

Figure 9.1: Example feedforward artificial neural network with interconnections

An ANN is composed of artificial neurons with connections or edges among each other [38]. Commonly its topology consists of three layers: input, hidden and output layers. Hidden layers are not visible from the outside, but they improve the abstraction of the ANN model. ANN is an adaptive system and can change its structure during training phase. This means building and deleting connections, adjusting edge weights or adjusting a neuron's activation function. Each neuron has a network function $f(x)$, that is a composition of functions $g_i(x)$, which again can be a composition of functions. By adding arrows between these functions or neurons it is possible to visualize a network structure. An established composition type is the *nonlinear weighted sum*:

$$f(x) = K\left(\sum_i w_i g_i(x)\right) \tag{9.11}$$

Here K represents the activation function and w_i is the weight for each function. Finding optimal parameters and training can be an effort. During the training phase three parameters are crucial for the overall runtime: The number of training iterations, the number of hidden layers and the number of neurons in layers. The numbers for hidden layers and neurons in hidden layers are also important with regard to the execution runtime of the neural network.
For instance, if we consider a simple neural network with three layers as illustrated in figure 9.1, the first layer represents the input layer with three neurons. This means our input feature vector is three dimensional. The third layer is the output layer with two neurons, since we assume that there are only two possible classes for our input feature vector. Now the central hidden layer consists of four neurons. The neural network's time complexity is of the order $\mathcal{O}(S \cdot (d \cdot N \cdot q \cdot m))$, with S as the cost for multiplying the weight with the input and adding it to the sum. Variable d is the number of input neurons and at the same time the dimensionality of the input vector. N denotes the number of neurons inside hidden layers and q is the number of hidden layers. Finally, m is again the number of classes.
According to our example in figure 9.1 the parameters have following values: $d = 3, N = 4, q = 1, m = 2$. Each hidden layer neuron computes the network function again for each input value of the input neurons. Output neurons compute their network function for each hidden layer neuron. Furthermore, the depicted complexity order above is an upper bound for the runtime, because not every neuron will be activated during prediction phase. This depends on the input values, activation function K and the network structure.

9.4 Experiments

This section focuses on accuracy and runtime performance of the analyzed system with respect to the introduced classifiers.

9.4.1 Parameter Settings

In section 2.4.1 we explained, that for own videos 20-fold cross validation is applied. This means, for instance, to find the optimal parameter k for the KNN classifier the 20-fold cross validation is repeated automatically several times with different values for parameter k. Then we select the value for k with the minimum classification error. The same procedure is passed through for RBC's radius parameter ϵ.
Finding the optimal parameter setting for the SVM classifier is different. Here the parameters for training the SVM model are essential. Furthermore, the SVM model is trained for each of the 20 cross validations separately based on the 190 training videos in each case. Hence, there is no connection between test and training videos. This procedure is repeated several times with different kernel and

parameter combinations. Afterwards, we pick the combination with the minimum classification error. By using the *LIBSVM Framework* [22] we substitute ϕ by linear, polynomial, RBF and sigmoid kernels. Also different kernel parameters are tested. The LIBSVM Framework allows combining parameter C with weights for each class. Moreover, different SVM types provided by LIBSVM as for instance C-SCV, nu-SCV or ϵ-SVR with related parameters are analyzed. The standard SVM type C-SCV combined with the RBF kernel worked accurate for our purpose. C is set to its default value 1 and there is no weighting applied. By this parameter we can influence the number of selected support vectors and by association the simplicity of the decision surface. The RBF kernel parameter γ provided by the LIBSVM framework was more important than parameter C during classification phase. With this parameter we can adjust the influence region of support vectors.
The artificial neural network is trained and optimized in the same way as the SVM model. Before each cross validation the neural network is trained once with 190 training videos. We manually adapt a set of parameters before each 20-fold cross validation. By the usage of the *Neuroph Framework* [64] we set the number of hidden layers, number of neurons in layers, number of learning iterations, learning rate, maximum network error, learning rule (e.g. momentum backpropagation) and parameters related to the learning rule.
When classifying external videos via KNN classifier or RBC, we also minimize the classification error by testing different parameters. But in these cases there is no cross-validation, external videos are assigned directly to the own video classes. For external videos the SVM model and the neural network are trained by own videos. Again here is no cross-validation applied, but the training with own videos is repeated until the best parameters are found.
Another interesting aspect is the classification of external videos with training parameters of self-taken videos. We have tested this case and as a result the classification of external videos worked accurate. In figure 4.3, for instance, one can see the accuracy progression for different 1D-functions and interval sizes. For both self-taken and external videos the centroid location based 1D-function has a maximum accuracy at an interval size of 4. For self-taken videos the accuracy is 0.80 and for external videos 0.40. After training the RBC with the self-taken video dataset, we applied the same ϵ-radius in order to classify the external video dataset. The accuracy declined from 0.40 to 0.36. The difference is only 0.04, so it is possible to train a classifier with one dataset and to use this classifier for the classification of a completely different dataset. This holds also for other classifiers as the KNN, SVM or ANN classifier.

9.4.2 Classifier Accuracies

Again for own videos direction information and for external videos location information of image moments is applied.
The ANN is a multilayer perceptron neural network, which utilizes supervised learning technique *backpropagation*. Neuron weights w_i are randomized and by adjusting the number of neurons in layers, the number of iterations and the learning rate the neural network is optimized. SVM's supervised learning algorithm is based on the C-SVC (C-Support Vector Classification) and the RBF as kernel. Here classes are not weighted.

Own Videos						
Class ACC	**KNN**	**ALC**	**NCC**	**RBC**	**SVM**	**ANN**
File	0.85	0.40	0.80	0.85	0.80	0.85
Hammer	0.85	0.80	0.75	0.95	0.85	0.75
Paint Roller	1.00	0.80	1.00	1.00	1.00	0.45
Paste Brush	0.95	0.90	0.95	1.00	0.90	0.85
Plane	0.80	0.65	0.55	0.50	0.80	0.85
Putty Knife	1.00	1.00	1.00	1.00	1.00	0.85
Sandpaper	0.90	0.40	0.90	0.94	0.65	0.80
Saw	0.90	0.95	0.90	0.75	0.90	0.60
Screwdriver	1.00	0.95	0.95	0.90	0.90	0.75
Wrench	0.95	1.00	0.95	1.00	1.00	1.00
Overall ACC	**0.92**	**0.79**	**0.88**	**0.89**	**0.88**	**0.78**

Table 9.1: Classifier accuracies for own videos

Table 9.1 shows resulting accuracies. Overall accuracies are not the average of class accuracies, but the ratio of correct classifications to the total number of classifications. Moreover, table 9.1 depicts that the classifier with the maximum accuracy 0.92 is the KNN classifier. Classifiers NCC, RBC and SVM also achieve high accuracies, whereas ANN classifier yields a minimum accuracy at 0.78. Considering single class classifications home improvement activity "Putty Knife" achieves extraordinary high accuracies. Almost all classifiers can assign all test videos correctly, because movements and by association features of this activity class are clearly different from other classes. Furthermore, the entire system works so accurate for own videos, that there is no single class with low accuracies for each classifier.

External Videos						
Class ACC	**KNN**	**ALC**	**NCC**	**RBC**	**SVM**	**ANN**
File	0.00	0.00	0.00	0.00	0.00	0.00
Hammer	0.00	0.00	0.00	0.00	0.00	0.00
Paint Roller	0.79	0.64	0.86	0.69	0.64	0.57
Paste Brush	0.25	0.00	0.00	0.25	0.25	0.25
Plane	0.09	0.00	0.09	0.00	0.09	0.00
Putty Knife	0.65	0.57	0.57	0.57	0.39	0.83
Sandpaper	0.36	0.09	0.55	0.18	0.64	0.64
Saw	0.18	0.45	0.27	0.33	0.27	0.45
Screwdriver	0.86	0.71	0.86	0.57	0.57	0.57
Wrench	0.25	0.63	0.25	0.50	0.25	0.25
Overall ACC	**0.41**	**0.37**	**0.42**	**0.40**	**0.35**	**0.45**

Table 9.2: Classifier accuracies for external videos

Now for external video classification ANN is the strongest approach. Its accuracy marks the maximum at 0.45. SVM falls down to 0.35, thus it is not as reliable for external videos as the other classifiers. Accuracy values for KNN, ALC, NCC and RBC are located around 0.40. Videos of the classes "File" and "Hammer" are completely misclassified, because of feature similarities to videos of the classes "Paint Roller" and "Paste Brush".

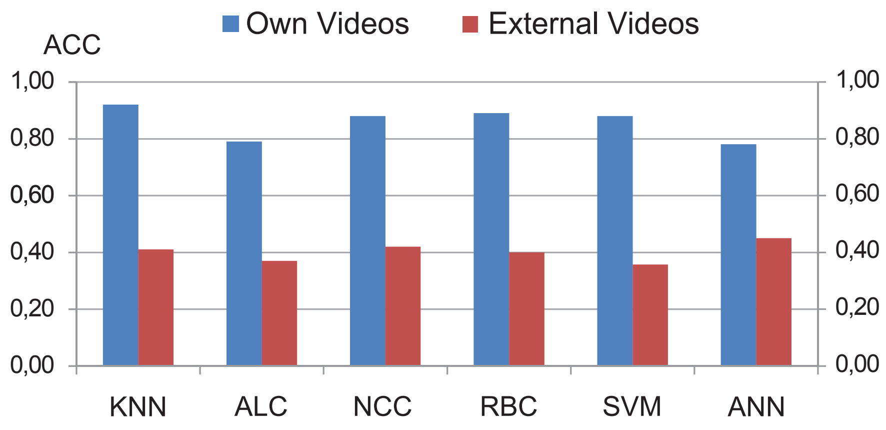

Figure 9.2: Overall accuracies for own and external videos

Figure 9.2 shows that own videos are assigned twice as accurate as external videos. This is due to irregular motions and recordings in external clips. For clear and smooth video motions the KNN classifier works most accurate. Otherwise, ANN classifier works best. A neural network can handle irregularities better than space model based classifiers. On the whole the KNN classifier seems to be the most effective classifier.
The SVM classifier is only reliable for external videos, if the amplitudes of the frequency domains are normalized before the classification. Otherwise, the accuracy is not 0.35, but 0.08 for 1D-functions based on the centroid's location. Since in external videos the motion range and also the camera distance to the motion are varying, the amplitude heights are not always at the same level. For instance, video C is a close-up, hence its dominating frequency has a high amplitude. Video D is distant to the camera and the same frequency has a low amplitude. This means, the feature vectors for videos C and D are not exactly at the same position inside the vector space, although the two videos show up the same main frequncy. This issue occurs for centroid location or centroid speed based 1D-functions. The direction of centroid's can only have three values and therefore resulting amplitude heights for different videos are usually at the same level.
Figure 9.3 illustrates the reason, why the SVM classifier has a low performance for external videos without normalization compared to a distance based classifier as the NCC. We consider two non-linearly separable classes A and B. After training the SVM, class A dominates the vector space and as a consequence the green test objects fall into the vector space of class A. From the point of view of a NCC the centroid of class B is nearer to the objects than the centroid of class A. So in this case class B would be the next class.
The green dots represent external videos, which are slightly shifted out of the vector space of class B, because the detected motion range differs. In fact, external video featues have 128 dimensions during evaluation phase. After normalizing each frequency domain by the maximum amplitude, test objects fall into the correct vector space. Normalizing takes place for both training data (own videos) and test data (external videos). Thus, compared to the other classifiers the SVM classifier needs an additional normalization step in order to work accurate. Adjusting parameters C and γ could not solve this issue.

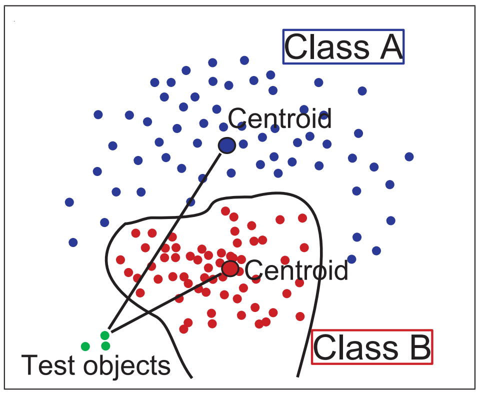

Figure 9.3: Simple illustration of slightly shifted external video features: The SVM classifier considers shifted, green dots as class A dots, whereas the NCC would assign them to class B.

9.4.3 Runtime Analysis

In this section we evaluate the runtime for the introduced classifiers. Two of the classifiers need a training phase before the classification process: The SVM classifier builds up a SVM model with hyperplanes and support vectors by the usage of training data during a training phase. This SVM model is used to assign objects to classes. Also the ANN classifier needs to train a neural network based on training data before the classification step. SVM models or neural networks can be saved and reused for further assignments.

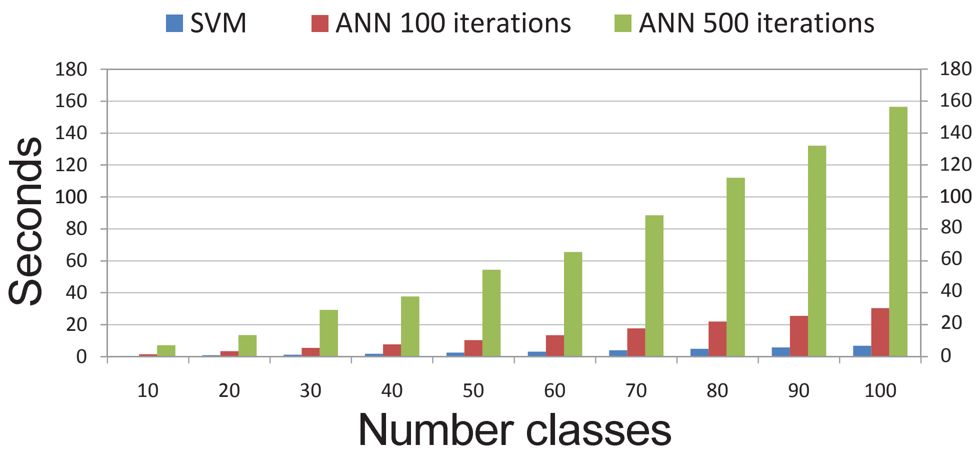

Figure 9.4: Training time for increasing number of referenced classes

Figure 9.4 shows that training the SVM model is much more efficient than training the neural network. For our approach a neural network with 16 neurons in the second layer combined with 500 training iterations worked most accurate. In this case the training time is 156 seconds for 100 training classes with 20 videos, whereas the SVM model training takes only 7 seconds. Concerning the neural network both parameters the number of neurons and the number of iterations are crucial for the runtime.

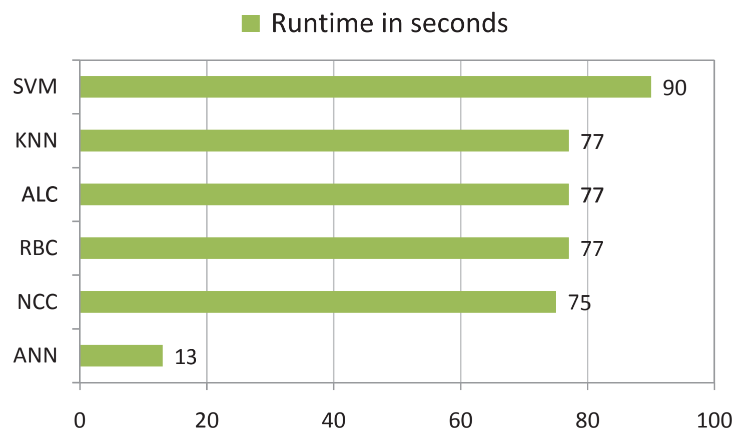

Figure 9.5: System's runtime with different classifiers

Figure 9.5 depicts the runtime for each classifier, when the number of test videos is set to 1000. The classification process relies on a database with 20 videos for each of the ten classes. Test series with SVM and ANN classifier are based on previously trained test model and neural network, respectively. Otherwise, runtime would exceed plotted time range.
The bar chart shows that for both RBC and ALC the classification process takes about 77 seconds. Since the two classifiers have similar algorithmic concepts, their runtime is identical. Each of the two classifiers computes distances between test object and all objects of each class. By the usage of the KNN classifier the runtime is also 77 seconds, because there are distance calculations between the test object and the entire object set. The additional operations for sorting the distances play a minor role as long as the class size is small. For the NCC only distance calculations to class centroids are necessary, so here the classifier is more efficient and the runtime amounts to 75 seconds.
Nevertheless, the ANN classifier outperforms all other classifiers by assigning 1000 videos in 13 seconds. Moreover, the mean runtime for each video assignment is 0.013 seconds. As we use the *Neuroph Framework* the weighted sum input function is optimized and the neural network is a matrix based implementation [64]. Hence, decisions by the ANN classifier are highly performant. On the other side the SVM classifier shows up low performance. In this case the *LIBSVM Framework* is utilized [22]. The one-against-one approach explained before needs much more operations to find the next class.
In figure 9.6 the runtime for 1000 classified videos with 20 to 300 referenced videos per class in database is presented. Here again the ANN classifier performs best. We detect a rather constant runtime around 13 seconds. The absolute runtime increase for the whole range is 0.3 seconds. For the SVM classifier the runtime stays around 90 seconds and the absolute runtime increase from 20 to 300 videos per class is 1.8 seconds. Here a growing referenced class size has a minor effect on the runtime, because the calculation of new divisions in space via hyperplanes relies only on support vectors. Furthermore, the number of support vectors does not necessarily increase, when new objects are added to the space. Besides the ANN and SVM classifiers, the NCC is also robust against a strong class size rise. For this case the line chart starts at 75.3 seconds and ends up at 80.0 seconds. Thus, the runtime difference is 4.7 seconds.
Furthermore, the RBC and the ALC show up a parallel runtime rise. As men-

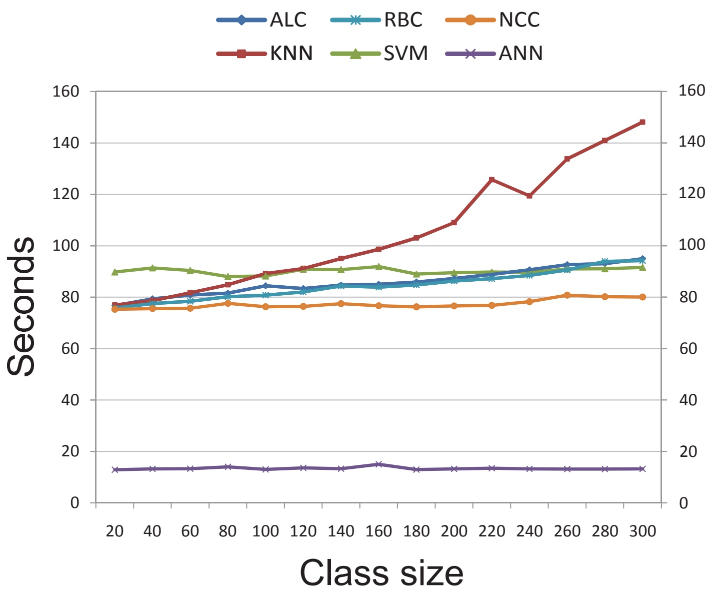

Figure 9.6: System's runtime with increasing referenced class size

tioned before the implementation for the two classifiers is pretty similar. For both of the classifiers the overall increase is 18 seconds. In contrast to the other classifiers the line chart of the KNN classifier has a clear upward movement. When the class size rises, the overall amount of objects rises and therefore more distances need to be sorted in order to find the k-nearest neighbors. This additional sorting step leads to the strong runtime increase.

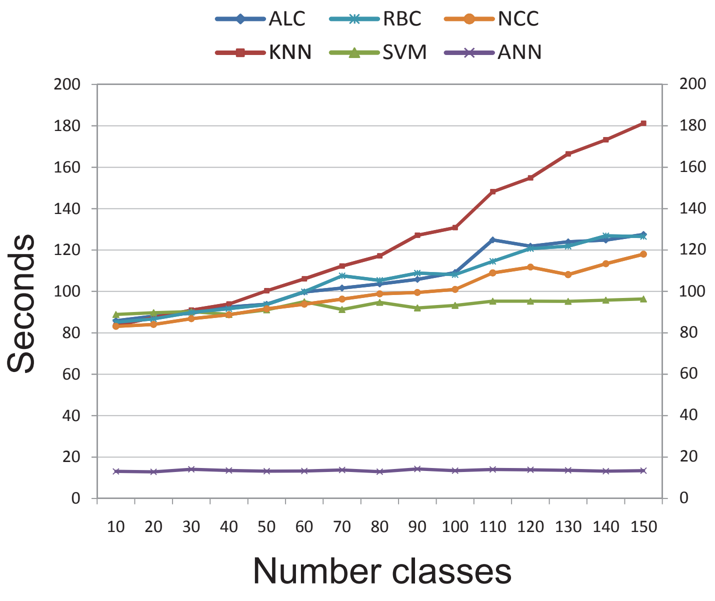

Figure 9.7: System's runtime with increasing number of classes

Figure 9.7 illustrates the runtime increase for a rising number of classes. The line chart shows runtimes for each classifier and 10 to 150 classes. Here again 1000 videos are classified and the class size of referenced classes in the database is set to 20.

At first glance it becomes apparent, that the increase of classes has a minor effect on the ANN classifier, but a major effect on the other classifiers. As mentioned

before the neural network is trained before the actual object classification process. This training involves all classes and yields a trained neural network, which only needs an input object in order to complete an assignment. So on the one hand the ANN classifier works fast for small classes and on the other hand core operations concerning the amount of classes are done during the training phase. These two aspects differ the ANN clearly from the other classifiers.
For a high number of classes the SVM classifier needs less time than the NCC, RBC, ALC and KNN classifier. Surprisingly the runtime stays rather constant even for a high number of classes. This means for a trained test model and additionally for small classes the one-against-one approach has a marginal influence on runtime. Contrary to figure 9.6 here the NCC reveals a clear upward movement, because this time the number of centroids increases. Again the KNN classifier surpasses all other classifiers, because the overall number of objects rises, when the number of classes rises. This means a higher cost for the k-nearest neighbors search.

9.5 Conclusion

The experimental stage of this chapter focused on different classifiers in order to find the most efficient one. In literature SVM and ANN classifiers show up the best results. The presented approach performs best with KNN and ANN classifiers. Although the KNN classifier is very simple, for videos with clear recording conditions it works best. With the KNN classifier 184 videos out of 200 videos could properly be assigned just relying on repeating motion data. The ANN classifier achieves higher accuracies than other classifiers, when recording conditions are not homogenous. A neural network is capable of handling irregularities concerning the frequency domain features better than spatial distance or spatial partitioning based approaches. Adding weights and activation functions to neurons leads to correct classifications even if AAFIs of some frequency areas vary. Hence, a video classification system relying on frequency features will work more accurately with ANN for most real world databases. A second clear advantage of the ANN classifier is its runtime performance. Runtime analyses show that the ANN classifier outperforms all other classifiers with a huge runtime difference.

Chapter 10

Optimized Background Subtraction for Moving Camera Recordings

As a very first step of the classification process the background is subtracted for each frame in the video. In the previous chapters Frame Differencing was applied, since this method brings out only highly active areas. This works accurate for our approach with a static camera. As soon as there is camera motion in the scene, this first step of the classification process will fail, because the foreground and background motion mix together. As a consequence the whole process is affected. Since camera motion can be found in many real world databases and applications, this aspect needs further investigation in order to ensure precise video assignments even for this type of videos.

This chapter exposes how state of the art background subtraction models can be optimized for moving camera recordings. In the course of the presented research work we found out that none of the commonly used background subtraction models is able to subtract the background accurately, when the camera is moving. Camera motion leads to motion areas in the background, whereas only motion of the foreground object is aimed to be detected. For the most part camera motion produces edges in the background. Therefore, a novel, iterative approach is presented, which is able to detect and remove edges from the background. The experiments show the accuracy increase for four chosen state of the art background subtraction models. Moreover, the introduced approach is compared to camera motion resistant background subtraction models and optical flow applications. The study for this chapter was published by Ayyildiz and Conrad in [11].

10.1 Introduction

Background subtraction is often the basis for content based image and video processing. Hence, background subtraction and foreground detection are highly investigated research areas and many background models are very accurate for static camera recordings. Some research focuses on handling moving objects in the background, but only little research is done on moving camera recordings. Since camera motion can be found in many video databases and real world

surveillance systems, this chapter analyzes the accuracy of a series of state of the art background subtraction models by applying videos with clear camera motion. We found out that none of the models is capable of handling camera motion or camera zoom accurately. In both cases the background becomes part of the detected foreground motion respectively foreground object. Often camera motion or zooming leads on to edges in the background. So now the research focuses on removing these edges in order to receive more accurate results.
First, we choose background subtraction models with a high potential of being accurate after edge removal. Then an iterative edge removal algorithm is applied and the resulting background subtraction image is smoothened. Segmentation is avoided in order to keep the evaluation focus on the improvement by edge subtraction. Moreover, the evaluation considers comparisons to camera motion resistant background subtraction methods and motion flow approaches. We also integrate this novel approach into the presented video classification system and evaluate the classification results.
Next in section 10.2 work related to the current contribution is introduced and explained. In section 10.3 four background subtraction methods are explained, which play a central role during the evaluation phase. Then in section 10.4 we present the novel edge removal approach. Finally, in section 10.6 the presented approach is analyzed.

10.2 Related Work

The most popular background subtraction models are the Mixture of Gaussians model (MOG) [83; 96] and the Kernel Density Estimation model (KDE) [33; 55]. In [83] the authors use a Mixture of Gaussians model in order to assign pixel intensity values to the background. Since each pixel has a history, this history can be modeled by a mixture of K Gaussian distributions. So the probability of belonging to the background is determined heuristically. If a pixel's intensity value does not belong to the background, the pixel is considered as part of the foreground. Foreground pixels are segmented into regions by a two-pass, connected components algorithm. Elgammal et al. [33] estimate the probability density function for N recent pixel intensity values. As kernel density function a normal Gaussian function is used, where each color channel is considered as independent. The resulting probability density function again is used to assign pixels to the background respectively to the foreground.
Commonly background subtraction algorithms are not able to distinguish between foreground and background, when the camera is moving. Therefore, some background subtraction research deals especially with camera motion. In [17] a Bayesian approach for detecting the foreground and estimating its motion at the same time is presented. Here the background is modeled by MOG and additionally the motion cue in a maximum a posteriori probability Markov Random Field (MRF) framework is used. Thus, the authors exploit advantages of detected motion at pixel and segmentation level, which leads to accurate results for mobile camera recordings. In [84] the authors present an approach that builds up a panoramic background image based on a moving camera video sequence. This panoramic image is created by first extracting and then tracking feature points via Kanade-Lucas-Tomasi (KLT) algorithm. Afterwards the same video sequence

is compared framewise to this background image in order to detect foreground objects. The authors of [95] describe local dependencies between pixels by so called Local Dependency Histograms (LDH). Based on this LDH idea each pixel is modeled as a group of weighted LDHs. After that a comparison of the current pixel LDH against its model LDHs yields, whether the pixel belongs to the foreground or background.
In section 10.6 state of the art background subtraction models are analyzed. For instance, models proposed by [13] and [46] are subject of the analysis. Both models are robust against camera jitter. The approach in [13] is based on a fuzzy mixture of Gaussians and [46] relies on a pixel based adaptive segmenter. Further on, we consider optical flow based ideas [63], that are robust to camera motion. In [63] the authors segment pixels by their motion flow orientation. So the system detects foreground motion even if there is a camera translation.
Background subtraction algorithms are the basis for video classification and surveillance systems. In [73] the authors use background subtraction for video genre classification. Their system classifies videos in three steps: First, the background motion caused by camera movement is calculated via optical flow analysis. Second, a background subtraction algorithm that compensates the measured background motion is applied in order to detect foreground motion. Third, object motion signals are calculated and serve as features during the classification phase.

10.3 Background Subtraction

Background subtraction algorithms have all a basic idea in common. Since we intend to detect the foreground in frames, there has to be at least one background model, that can be subtracted from a particular frame.

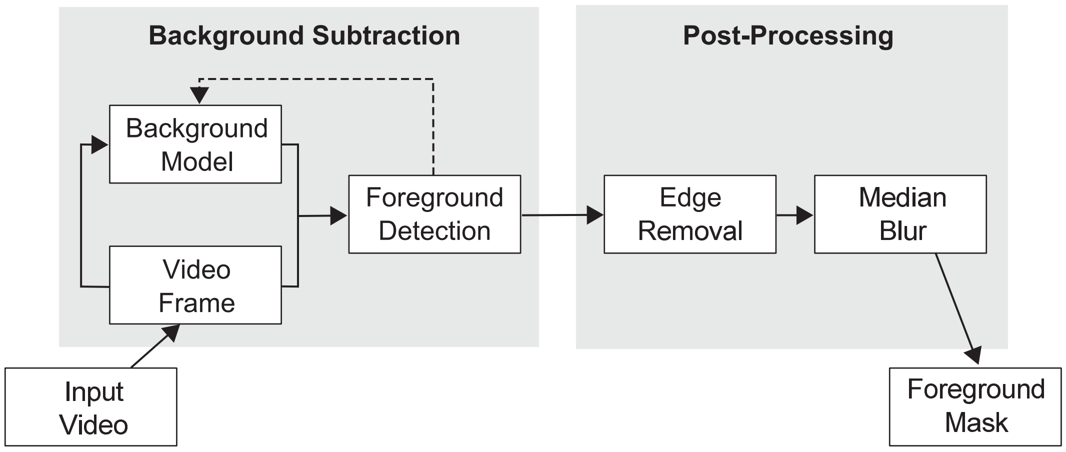

Figure 10.1: Background subtraction flow diagram

Figures 10.1 and 10.2 illustrate this basic technique. Furthermore, the flow diagram in figure 10.1 shows the post-processing phase. Both background subtraction models and post-processing steps are analyzed in section 10.6. The most important background subtraction models we evaluate are Frame Differencing (FD), Weighted Moving Variance (WMV), Vu Meter (VM) and GMG.

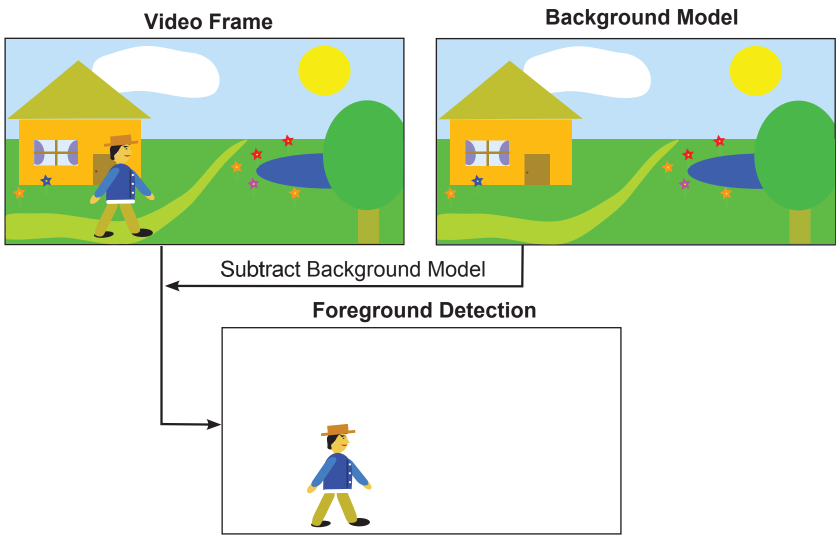

Figure 10.2: Background subtraction illustration

10.3.1 Frame Differencing

Frame Differencing is the simplest background modeling technique [77]. Here the video frame at time $(t-1)$ is considered as the background model for the frame at time t. This approach assumes that all foreground pixels are moving and all background pixels are static. It is very efficient, since for m pixels only $\mathcal{O}(m)$ operations are necessary for frame comparison. Interior pixels of homogeneously colored moving areas may not be recognized by this approach. This is generally referenced as the *aperture problem*.

$$FD_t(x,y) = |I_t(x,y) - I_{t-1}(x,y)| \tag{10.1}$$

Equation (10.1) formulates the absolute pixel intensity difference for two consecutive frames. Here x and y represent the pixel coordinates. A pixel is considered as part of a motion, only if equation (10.2) is fulfilled. Parameter T represents a predefined threshold.

$$FD_t(x,y) > T \tag{10.2}$$

10.3.2 Weighted Moving Variance

Weighted Moving Variance is a further background subtraction model, which basically models the background by computing variance values for each pixel [80]. The variance calculation takes place for the last n frames, where each frame has a different weighting. Moreover, the WMV algorithm is based on a weighted moving mean. Mean and variance calculations are formulated by equations (10.3) and (10.4). Parameter w represents the weight and I is the pixel intensity at coordinates (x,y). We define $\sum_{i=0}^{n} w_i = 1$. Now the dominating runtime factors are the frame pixel number m and the size n of the time window, so the time complexity is $\mathcal{O}(m \cdot n)$.

$$\mu(x,y) = \sum_{i=0}^{n} w_i I_i \tag{10.3}$$

$$\sigma^2(x,y) = \sum_{i=0}^{n} w_i (I_i - \mu)^2 \tag{10.4}$$

Equation (10.5) defines the difference between the variance of the current frame and the variance of the last n frames.

$$WMV_t(x,y) = |(I_t(x,y) - \mu)^2 - \sigma_t^2(x,y)| \tag{10.5}$$

Only a significant difference beyond a predefined threshold denotes a pixel as foreground motion.

$$WMV_t(x,y) > T \tag{10.6}$$

10.3.3 Vu Meter

A probabilistic and non parametric approach to define the image background model is proposed by [43]. The authors name their model Vu Meter. Here each pixel can take two possible states: The first state ω_1 denotes that the pixel is part of the foreground and the second state ω_2 marks the pixel as part of the background. Let $v_t(p) = \begin{pmatrix} \scriptstyle red\ green\ blue \end{pmatrix}^T$ the color vector for pixel p at time t. Then we estimate the probability density function for the background by decomposing $v_t(p)$:

$$P(\omega_1|v_t(p)) = \prod_{i=1}^{3} P(\omega_1|v_{t_i}(p)) \tag{10.7}$$

with

$$P(\omega_1|v_{t_i}(p)) \approx K_i \sum_{j=1}^{N} \pi_{t_{ij}} \delta(b_{t_i}(p) - j) \tag{10.8}$$

K_i is a normalization constant that ensures $\sum_{j=1}^{N} \pi_{t_{ij}} = 1$, where $\pi_{t_{ij}}$ represents a discrete mass function. Moreover, δ is the Kronecker delta function. Bin index vector $b_t(p)$ is associated with $v_t(p)$ and j is the bin index. Furthermore, there is a temporal model update for $\pi_{t_{ij}}$. For each new frame at time t we define:

$$\pi_{t_{ij}} = \pi_{t-1_{ij}} + \alpha \cdot \delta(b_{t_i}(p) - j) \tag{10.9}$$

Finally, foreground pixels are determined by equations (10.10) and (10.11).

$$VM_t(p) = P(\omega_1|v_t(p)) \tag{10.10}$$

$$VM_t(p) > T \cdot \alpha \tag{10.11}$$

Thus, bin size N, threshold T and variable α are essential parameters for an accurate foreground detection.

10.3.4 GMG

This section presents a background modeling approach proposed by Godbehere, Matsukawa and Goldberg (GMG) in [42]. This algorithm combines statistical background image estimation and per-pixel Bayesian segmentation. An image $\mathcal{M}(t)$ is compared to the statistical background image model $\mathcal{H}(t)$ in order to create a posterior probability image. Filtering, segmenting and tracking algorithms are applied afterwards to detect the foreground and update the image background model.
The initial background model or estimated pixel probability mass function is based on the first n frames of a video. Histogram $\mathcal{H}_{ij}(t)$ is interpreted as a vector in the RGB color-space. $\mathcal{H}_{ij}(n)$ is then considered as an average of the last n features (pixel colors) f_{ij}.

$$\mathcal{H}_{ij}(n) = \frac{1}{n}\sum_{k=1}^{n} f_{ij}(k) \tag{10.12}$$

Let F the foreground and B the background. Since prior probability $P(F)$ and $P(f_{ij}|B) = f_{ij}(t)^T\mathcal{H}_{ij}(t)$ are known, we can calculate posterior pixel $\mathcal{P}_{ij}(t)$:

$$\mathcal{P}_{ij}(t) = P(F|f_{ij}(t)) = 1 - P(B|f_{ij}(t)) \tag{10.13}$$

Posterior image $\mathcal{P}(t)$ is then filtered by a morphological open and close. Afterwards, Godbehere et al. detect 8-connected regions of pixels and label these as foreground. Also a set of bounding boxes $\mathcal{B}(t)$ around the connected components are determined. The Kalman filter bank maintains a set of tracked objects and has predicted bounding boxes $\mathcal{Z}(t)$. A Gale-Shapley matching algorithm pairs elements of $\mathcal{B}(t)$ and $\mathcal{Z}(t)$. By these pairs the Kalman Filter bank is updated. The result is $\widehat{\mathcal{Z}(t)}$, the collection of pixels identified as foreground.
The background image model is updated by the usage of $\widehat{\mathcal{Z}(t)}$ and image $\mathcal{M}(t)$, where only pixels identified as background are updated. The oldest feature is subtracted from $\mathcal{H}_{ij}(t)$. Then by equation (10.14) the last observed feature is added to the background histogram. The histogram's adaption rate is controlled by parameter α.

$$\mathcal{H}_{ij}(t+1) = (1-\alpha)\mathcal{H}_{ij}(t) + \alpha f_{ij}(t) \tag{10.14}$$

10.4 Edge Subtraction

State of the art background subtraction methods work very accurate when videos are captured by a static camera. If the camera is moving while it captures an activity, then background subtraction methods become inaccurate, because motion in the foreground and the moving background cannot be distinguished clearly anymore. Often camera movements lead on to edges in the detected foreground image. Therefore, we developed an edge subtraction algorithm, which aims to reduce false foreground detections. As shown in figure 10.1 it is part of the optimization and post-processing step.

10.4.1 Edge Removal Idea

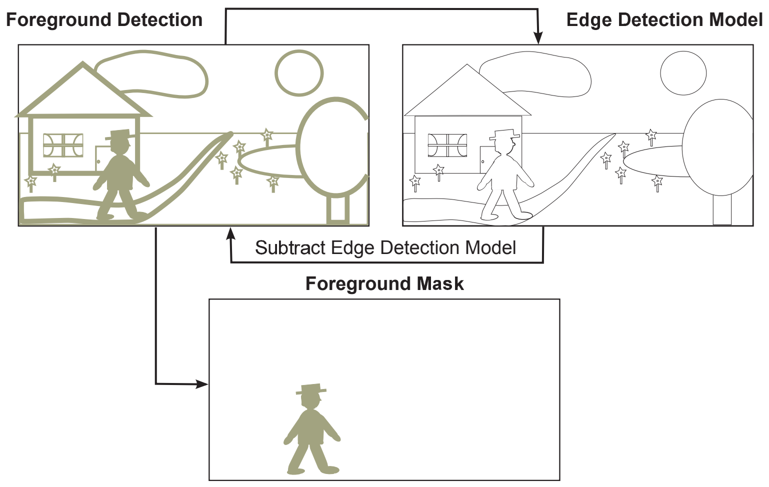

Figure 10.3: Edge subtraction illustration

Several standard background subtraction algorithms are able to detect foreground motion accurately even if the camera is moving. Unfortunately these algorithms also interpret edges of background objects as part of the foreground motion. Figure 10.3 illustrates that there are thin and bold edges. The most efficient way for eliminating different edge types is removing thin layers iteratively. These thin layers are computed by an edge detection algorithm.
The following algorithm depicts this idea. For an input image *FG_Detection*, which represents the foreground detection image, iteratively the edges are detected and removed. For experimental purposes we apply the Canny and the Laplace edge detectors. Subsequently, a median blur on the edge subtracted foreground image is performed. The optimal number of iterations depends on the edge strength and the video frame resolution. The presented edge subtraction algorithm has one disadvantage; it removes not only background edges but also thin borders of the foreground object.

Algorithm Egde Subtraction

```
Input: Image FG_Detection
Output: Image FG_Mask

for i = 1 to n  do
  Image Edge_Model = detectEdges(FG_Detection)
  FG_Detection = removeEdges(FG_Detection,Edge_Model)
end for

FG_Detection = blur(FG_Detection)
FG_Mask = FG_Detection
```

10.4.2 Laplace Edge Detector

A Laplace edge detector uses the *Laplace-Operator* [89]. This operator is the sum of second derivatives of $f(x)$ and $f(y)$ in the context of two dimensional image data as shown in equation (10.15). Only if $\Delta f = 0$ is fulfilled, there is actually a local minimum or maximum, which again gives a hint for an edge inside an image.

$$\Delta f = \frac{\partial^2 f(x)}{\partial x^2} + \frac{\partial^2 f(y)}{\partial y^2} \tag{10.15}$$

By discretization of the second partial derivatives the Laplace operator can be described as a convolution matrix. This convolution matrix is applied to discrete input signals.

$$D^2_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{10.16}$$

10.4.3 Canny Edge Detector

A further efficient edge detector is the Canny edge detector [21]. This edge detector is based on a multistage algorithm with four main stages. As a first step noise reduction is conducted by a Gaussian filter. Second, the gradient strength G and direction θ are determined by applying convolution matrices G_x and G_y. Since each pixel has eight neighbors, there are four possible angles 0°, 45°, 90° and 135°. θ is rounded to one of these angles.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{10.17}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{10.18}$$

$$\theta = arctan(\frac{G_x}{G_y}) \tag{10.19}$$

The third step is called *non-maximum suppression.* This term means that a pixel is not allowed to have a neighbor pixel with a gradient strength that is higher than its own. The only exception is a neighbor pixel along the edge direction. An edge is maximal one pixel wide. During the fourth and last step hysteresis thresholding eliminates further false edge pixels. Let T_1 an upper threshold and T_2 a lower threshold, then we define:

1. If $G > T_1 \rightarrow$ accept pixel as part of an edge
2. If $G < T_2 \rightarrow$ reject pixel as part of an edge
3. If $G < T_1 \wedge G > T_2 \wedge \exists$ neighbor with $G > T_1 \rightarrow$ accept pixel as part of an edge

10.5 F-Score

In order to measure the accuracy of a foreground mask, we utilize pixel based precision and recall values. There is also a further measurement called *F-score*, that combines precision and recall [87]. Sometimes this measurement is referenced as F-measure or F_1 score. The result is a harmonic mean, where recall and precision are evenly weighted. A F-score of 1.0 proves maximal accuracy, whereas 0.0 represents the lowest possible performance.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{10.20}$$

Let $\beta \in \mathbb{R}_{\geq 0}$, then we can define a more general F_β score, which allows to weight precision and recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{10.21}$$

Hence, a F_2 score for instance weights recall higher than precision. By contrast a $F_{0.5}$ score puts more weight on precision than recall.

10.6 Experiments

In this section we discuss the accuracy of background subtraction and edge subtraction algorithms in the context of moving camera recordings. We compare the edge subtraction approach to background subtraction and optical flow approaches, which are basically robust against camera movements. Furthermore, we present experimental results for edge subtraction applied to our video classification system. Runtime evaluation for background subtraction and edge subtraction algorithms takes place at the end of this section.

10.6.1 Background Subtraction Comparison

For the study at hand more than 30 background subtraction algorithms provided by the BGS Library [80] were analyzed. The results show that state of the art background subtraction algorithms cannot handle camera motion accurately. Figure 10.4 illustrates background subtraction results for a selection of background models.

Foreground detection images obtained by WMV, FD, GMG and VM contain edges and background noise. Nevertheless, these background subtraction algorithms are the most accurate ones. WMV can handle the aperture problem gradually better than FD, because the background model depends not only on the previous frame. Background subtraction by GMG detects almost the entire foreground object, since a morphological close and a per-pixel Bayesian segmentation are applied. But at the same time these steps lead to large regions labeled as foreground in the background. The VM algorithm shows contours of the foreground object correctly, but its probability density function is sensitive to background motion.

BGS models MOG [83], Adaptive SOM [57], Adaptive BG Learning [94], Prati

Medoid [29] and Eigenbackground [67] have all one property in common: Adapting the background model takes place slowly. Therefore, the athlete in figure 10.4 seems to be mirrored. If he returns to his original position, his body is partially invisible. Even more crucial is the fact, that the more the camera moves from its original position the more the background becomes part of the detected foreground. For MOG it is possible to increase the learning rate, which reduces these side effects. But this again leads to the aperture problem. Multi Cue modeling utilizes a segmenting algorithm [66]. This segmenting step results in large, filled background regions detected as foreground.

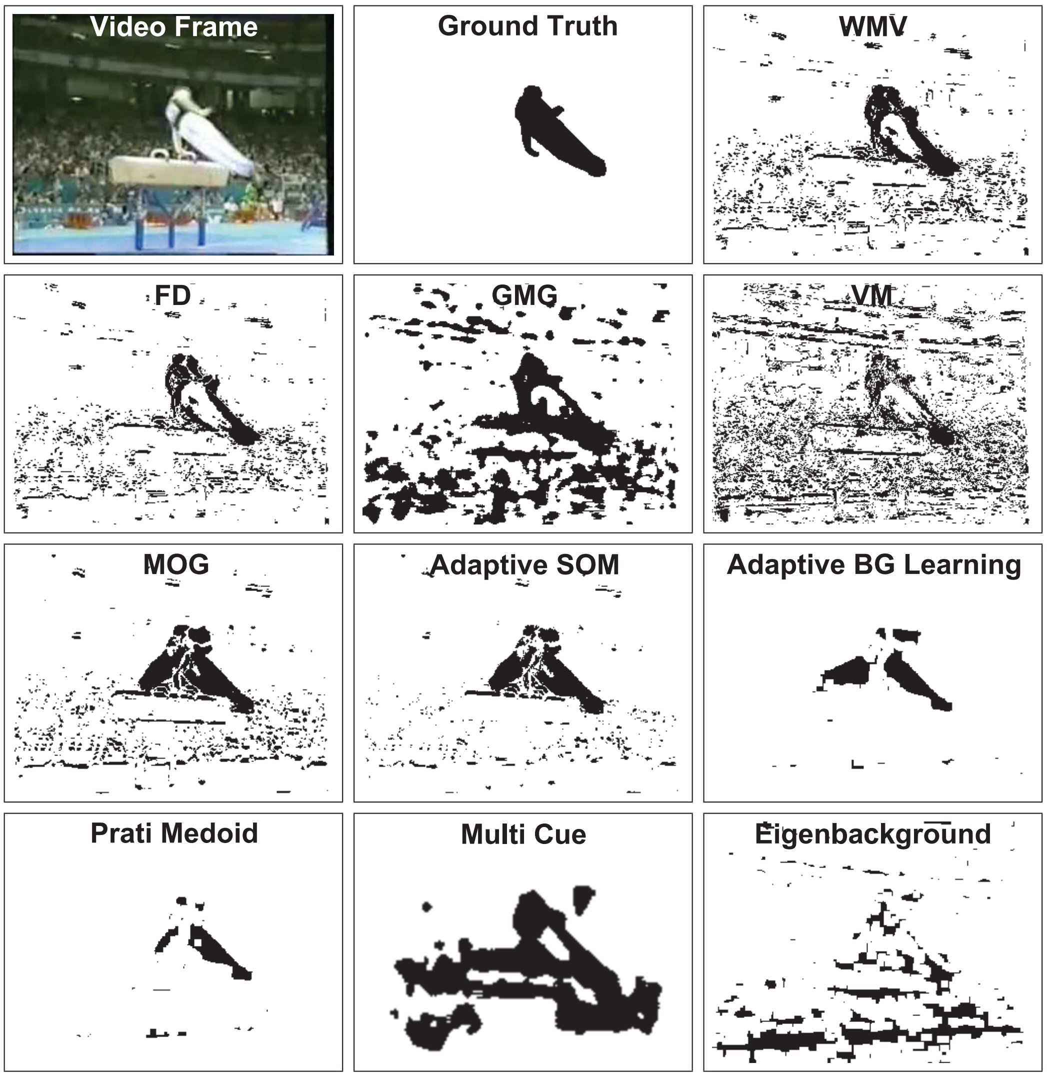

Figure 10.4: State of the art BGS algorithms applied to a video scene recorded by a moving camera

10.6.2 Edge Subtraction Illustration

On the left figure 10.5 shows a colored video frame with a person exercising body weight squats. The original video is recorded by a handheld camera. Besides, Laplace and Canny edge detection for a Vu Meter foreground detection are illustrated. The two approaches show up similar results.
Now for the purpose of illustration we take edges detected by Laplace edge detec-

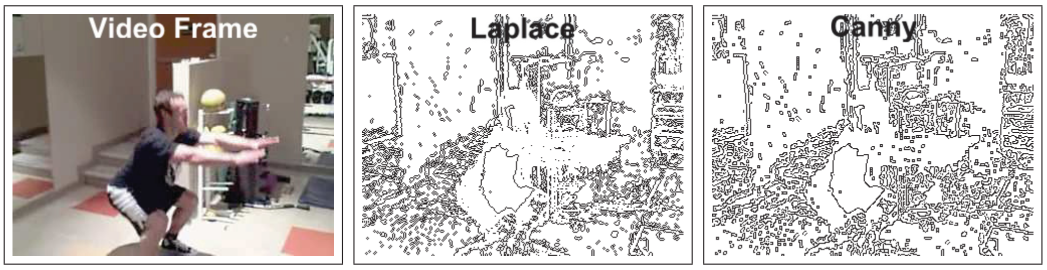

Figure 10.5: Laplace and Canny edge detections for a Vu Meter foreground detection image

tor and subtract these edges iteratively. As a last step a median blur is performed. Figure 10.6 depicts the resulting foreground masks.

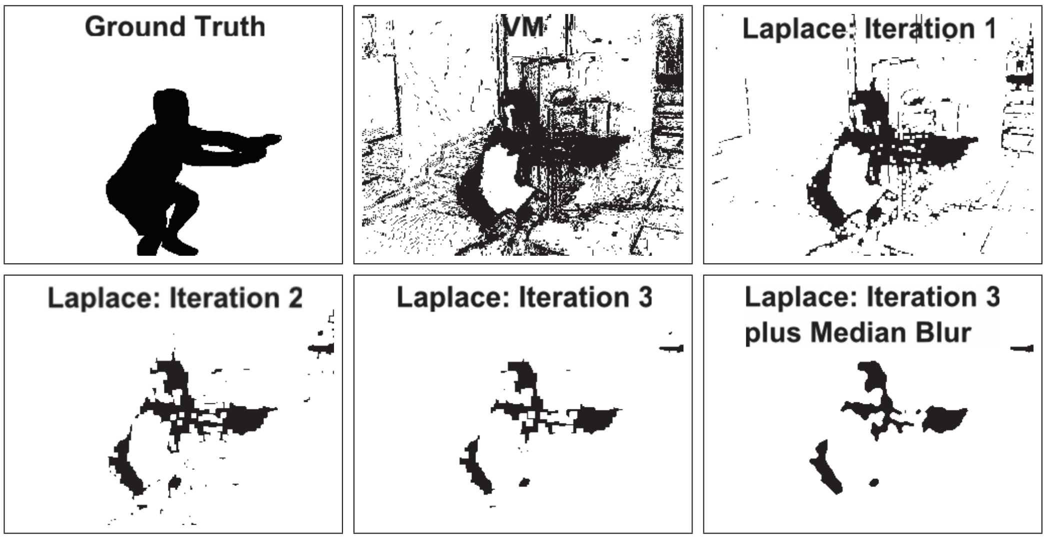

Figure 10.6: Iterative edge subtraction via Laplace edge detector for a Vu Meter foreground detection image

The last frame in figure 10.6 shows that the background is almost completely removed. Thus, the precision becomes very high, but at the same time the recall is decreased.

10.6.3 Edge Subtraction Accuracy

In order to determine the accuracy of the edge subtraction algorithm, we compute the recall and precision values for 50 different video frames. Ground truths are created manually. Each video frame originates from a different video and each video belongs to one out of ten sports video classes taken out of the UCF101 database [82]. Hence, we have 5 analyzed frames per sports video class. We consider only frames with a clear and strong camera pan, tilt, zoom or jitter.

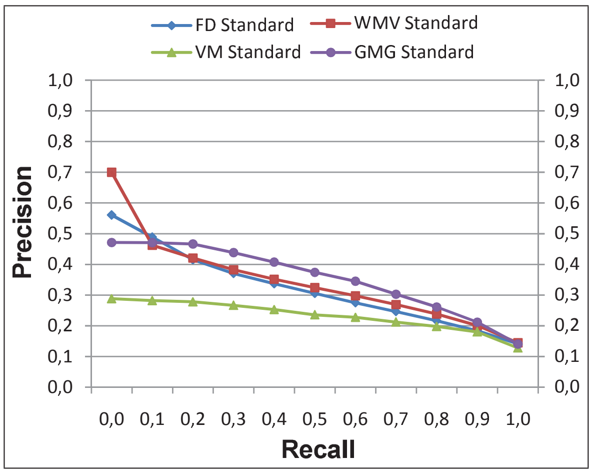

Figure 10.7: Precision and recall progression for BGS without optimization

Figure 10.7 depicts the precision-recall progression for the four main BGS algorithms without optimization. GMG performs best, WMV and FD are in the middle field and VM performs weakest. Next in figure 10.8 we see that the presented edge subtraction algorithm improves the performance for each of the four BGS algorithms. The WMV algorithm benefits most and outperforms even the GMG algorithm. On the whole Laplace edge subtraction works slightly better than Canny edge subtraction. Furthermore, a following median blur improves results slightly in each case.

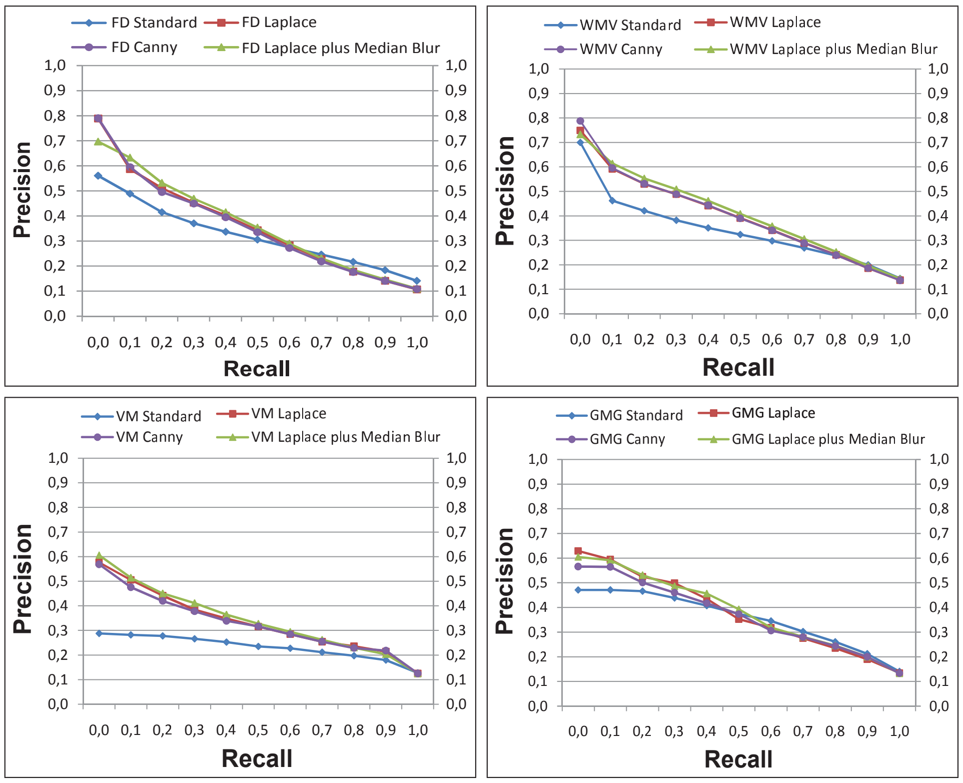

Figure 10.8: Precision and recall progression for BGS with optimization

Figure 10.9 shows precision values for a fixed recall at 0.40. Since athletes in sports videos often move only few body parts and not the whole body, high

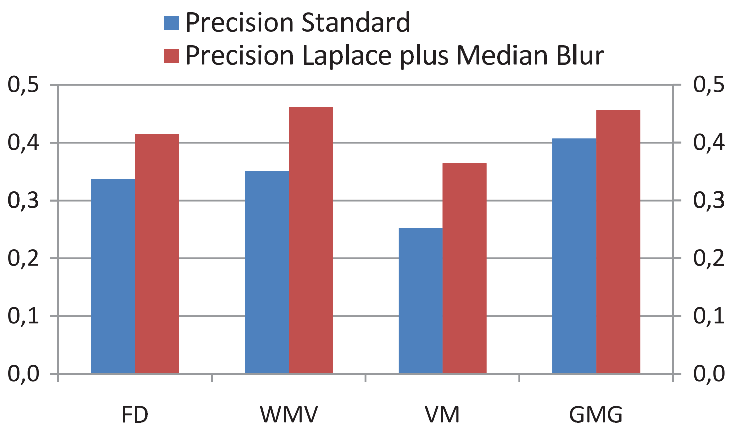

Figure 10.9: Precision values with and without edge subtraction for a fixed recall value

precision values for a recall at 1.0 are not attainable (only with segmentation). For each BGS approach we can state a clear precision increase after edge subtraction. WMV and GMG achieve maximal precision values of 0.46. In addition, both WMV and VM have maximal precision increases from 0.35 to 0.46 and from 0.25 to 0.36, respectively. The increase in both cases is 0.11. Figure 10.4 illustrated, that WMV and VM have both very edgy and noisy backgrounds. Hence, edge subtraction optimizes these approaches more than FD or GMG. Moreover, GMG without edge subtraction has already a high precision level at 0.41. Here the morphological filter and the segmenting increase the number of correctly detected foreground pixels and eliminate edges and noise at the same time.

F-Score				
Video Class	**FD**	**WMV**	**VM**	**GMG**
Bench Press	0.40	0.46	0.44	0.46
Bodyweight Squats	0.55	0.55	0.46	0.54
Boxing Speedbag	0.39	0.40	0.42	0.43
Hula Hoop	0.42	0.43	0.31	0.42
Jumping Jack	0.35	0.42	0.40	0.39
Jump Rope	0.29	**0.29**	**0.20**	**0.24**
Pommel Horse	0.43	0.58	0.38	0.48
Pullups	0.36	0.44	0.39	0.46
Pushups	**0.60**	**0.60**	**0.52**	**0.58**
Wall Pushups	**0.26**	0.36	0.40	0.36
Overall	0.40	**0.45**	**0.39**	0.44

Table 10.1: F-score for BGS models combined with Laplace edge subtraction and median blur

Table 10.1 lists maximal F-score values for each sports video class and BGS approach. Laplace edge subtraction with a following median blur is applied to each approach. Here WMV performs best with an overall F-score at 0.45. It achieves the highest high and the highest low with F-score values 0.60 and 0.29, respectively. With an overall F-score at 0.39 VM is the weakest approach. There is only little correlation between sports class and F-score. The F-score depends more on camera and environment settings. For instance, pushups achieve high F-score values, because the camera is often near to the moving athlete in this class. A large foreground area reduces the number of false foreground detections.

F-Score				
Video Class	**Standard**	**Laplace**	**Canny**	**Laplace & MB**
Bench Press	0.44	0.45	0.45	**0.46**
Bodyweight Squats	0.48	0.54	0.54	**0.55**
Boxing Speedbag	0.39	0.39	0.39	**0.40**
Hula Hoop	0.38	0.41	0.41	**0.43**
Jumping Jack	0.39	0.40	0.40	**0.42**
Jump Rope	**0.30**	0.28	0.28	0.29
Pommel Horse	0.40	0.55	0.55	**0.58**
Pullups	0.34	0.42	0.42	**0.44**
Pushups	0.52	0.58	0.59	**0.60**
Wall Pushups	0.35	0.35	0.35	**0.36**
Overall	**0.40**	0.44	0.44	**0.45**

Table 10.2: F-score for WMV combined with edge subtraction

Table 10.2 compares F-score values for WMV standard to WMV combined with edge subtraction. Results confirm the precision-recall progression for WMV in figure 10.8. The overall F-score for WMV standard rises from 0.40 to 0.45 with Laplace edge subtraction and additional median blur. Moreover, the strongest F-score increase is detected for the pommel horse class. Here the F-score rises from 0.40 to 0.58. Particularly videos of this class show up clear background edges. Both Canny and Laplace edge subtraction without a subsequent median blur show up an overall F-score at 0.44. For Laplace edge subtraction the F-score for every single class rises, when a median blur is applied.
Finally, typical edge subtraction results are showcased by figure 10.10. The analyzed video frame belongs to the sports video class ”Bodyweight Squats”.

10.6.4 Edge Subtraction Comparison

In this section the presented approach is compared to state of the art BGS applications claiming to be robust to camera movement or camera jitter. Figure 10.11 illustrates BGS results for pixel based adaptive segmenters *PBAS* and *ViBe* [46; 14], a fuzzy based technique [13] and a kernel density estimation based technique [33]. In fact, none of the four approaches is robust to camera motion; camera motion leads to edges in the background and slowly adapting background models intensify this behavior. By contrast WMV combined with edge subtraction is very close to the ground truth.
Despite of high accuracy for the novel, edge subtraction based method two further methods were found, which seem to be even more accurate. The experimental results in figures 10.12 and 10.13 rely on data provided by [63] and [34], respectively. The two approaches use segmentation algorithms, which is avoided here in order to stay focused on the edge subtraction aspect. Narayana et al. [63]

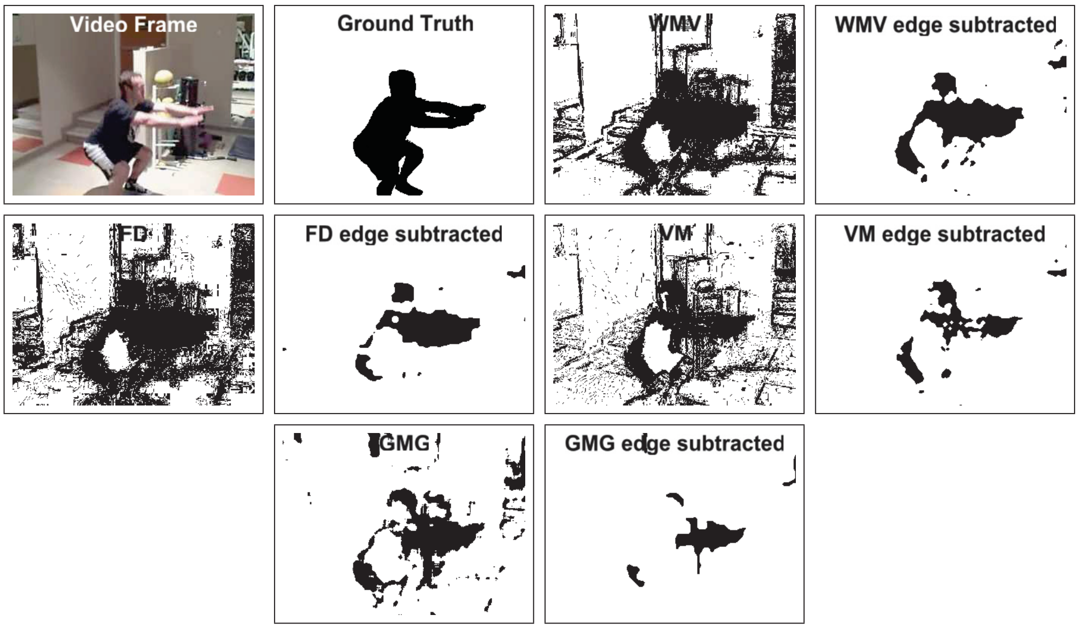

Figure 10.10: Comparing BGS results without and with edge subtraction via Laplace and median blur

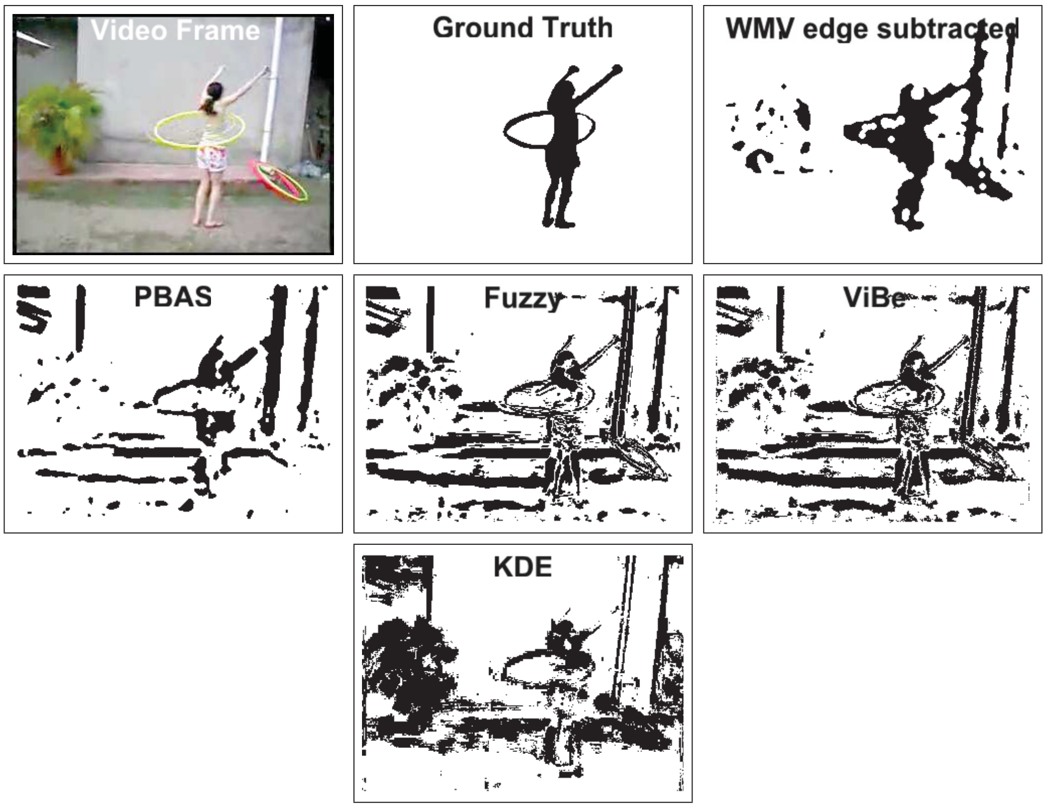

Figure 10.11: Comparison between edge subtraction and BGS approaches stating to be robust to camera motion

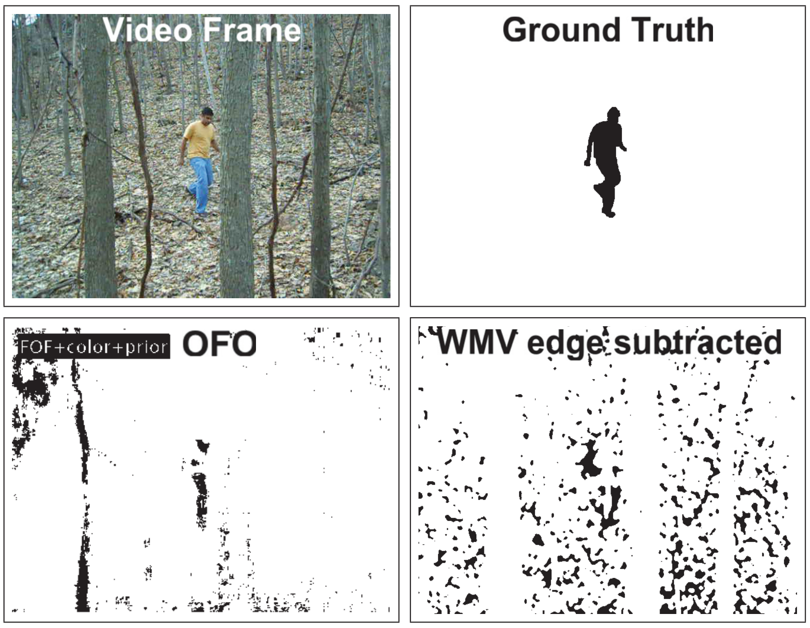

Figure 10.12: Comparison between edge subtraction and BGS approach using optical flow orientations

use optical flow orientations instead of optical flow vectors or magnitudes, because orientations are independent from the object's depth. On the one hand this means camera translation does not emphasize objects that are nearer to the camera and on the other hand the whole background has the same orientation and differs from the foreground motion. Figure 10.12 approves these properties for a challenging forest hiking video.

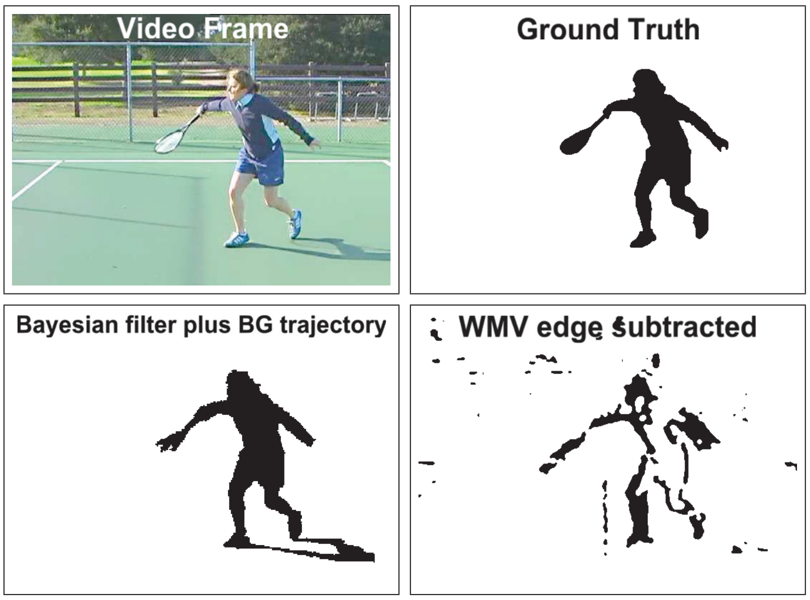

Figure 10.13: Comparison between edge subtraction and BGS approach using Bayesian filtering and BG trajectories

Elqursh and Elgammal [34] utilize long term background trajectories and combine this information with pixel based Bayesian filtering. Detected foreground masks are very near to the ground truth as shown in figure 10.13. Only the shadow of the tennis player on the floor reduces the precision. For this frame additional segmentation could increase the recall of our presented approach strongly.

10.6.5 Edge Subtraction applied to Video Classification

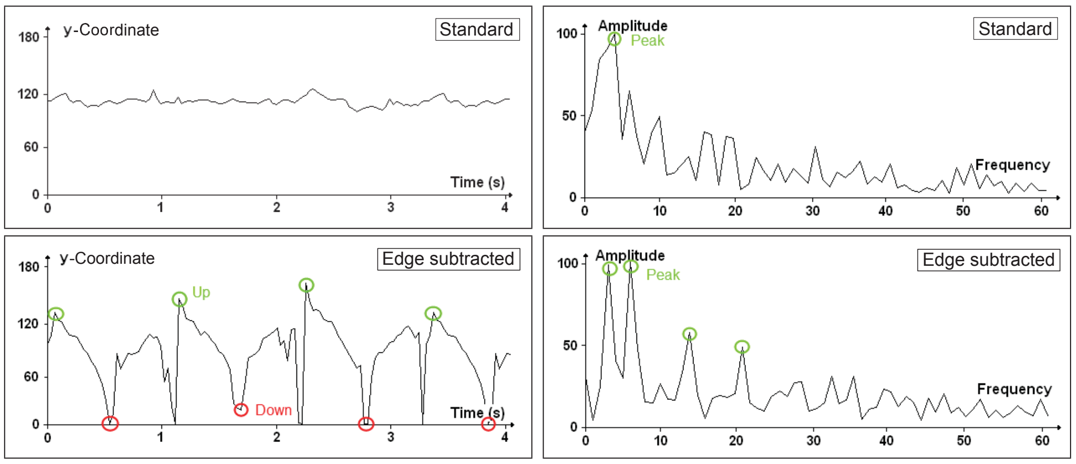

Figure 10.14: Trajectory of the centroid's y-coordinate and related frequency domain with and without edge subtraction

Section 2.1 introduced a video classification system that is based on frequency features of repeating movements. Now we analyze the system's accuracy with and without Laplace edge subtraction for the same 50 videos we already utilized in section 10.6.3. We do not use our self-recorded home improvement videos, since these videos do not contain camera motion. For this type of videos FD as background modeling approach is sufficient.

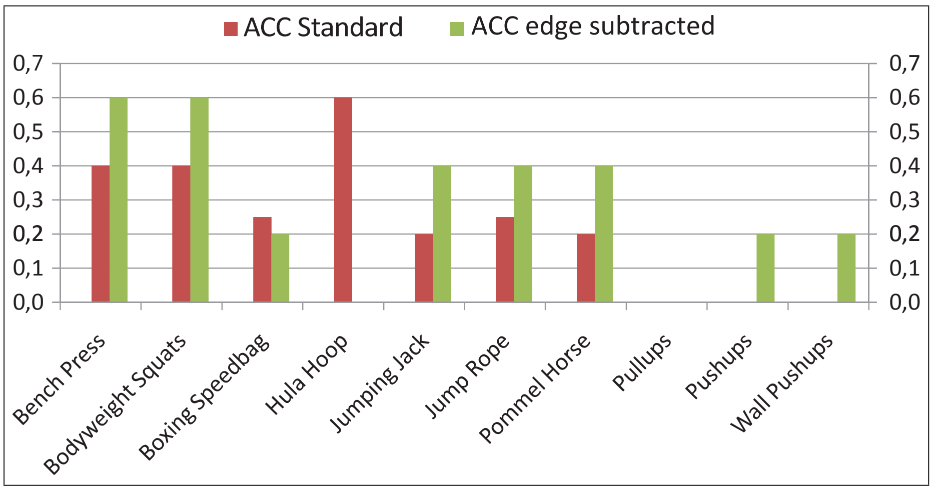

Figure 10.15: Accuracy of classification with and without edge subtraction

The foreground mask calculation takes place via WMV. First of all, the trajectory for the y-coordinate of the foreground centroid for a pullup video without edge subtraction is illustrated; see upper left plot in figure 10.14. The upper right plot represents the frequency domain and shows only one clear peak. The plots at the bottom illustrate the y-coordinate trajectory and the frequency amplitudes with

edge subtraction. The main motion and the resulting frequencies become more decisive. We receive up to four clear amplitude peaks. The additional y-lows between each up and down appear, when the athlete swings his feet.
For the video classification process the 5-fold cross validation is applied. The bar diagram in figure 10.15 visualizes that edge subtraction increases the accuracy for the most classes. Further, the overall accuracy increases from 0.23 to 0.30. For the hula hoop and pullups classes no video could be assigned properly. This has two main reasons: First, edge subtraction works not accurate for all videos. Second, since only five videos per class are available, the classifier works with features of four videos per class. As mentioned before a classifier relying on a small databases cannot handle varying frequency patterns for one action type accurately.

10.6.6 Runtime Analysis

After determining the accuracy of different background and edge subtraction algorithms, now we focus on the runtime of those algorithms. Therefore, we save background and edge subtracted frames for a tennis video clip. For edge subtraction the number of iterations is set to 5. The video is 31 seconds long and has a frame rate of 15 frames per second. Each frame has a 530 × 380 resolution. The runtime analysis is conducted by a 2.2 GHz CPU. Runtime results are presented by figure 10.16. GMG needs the most operations, whereas FD is the fastest approach. As explained in section 10.3.4 GMG combines a set of algorithms for instance for filtering, segmenting or tracking. These additional operations again lead to a long runtime. On the contrary, FD has the simplest algorithmic concept and is therefore so fast. The bar diagram on the left shows a slight runtime increase, when Laplace edge subtraction is applied. Moreover, the bar diagram on the right depicts a strong runtime increase, when Canny edge subtraction is used. This strong runtime increase arises from the multistage algorithmic concept explained in section 10.4.3. A median blur has a minor influence on the runtime.

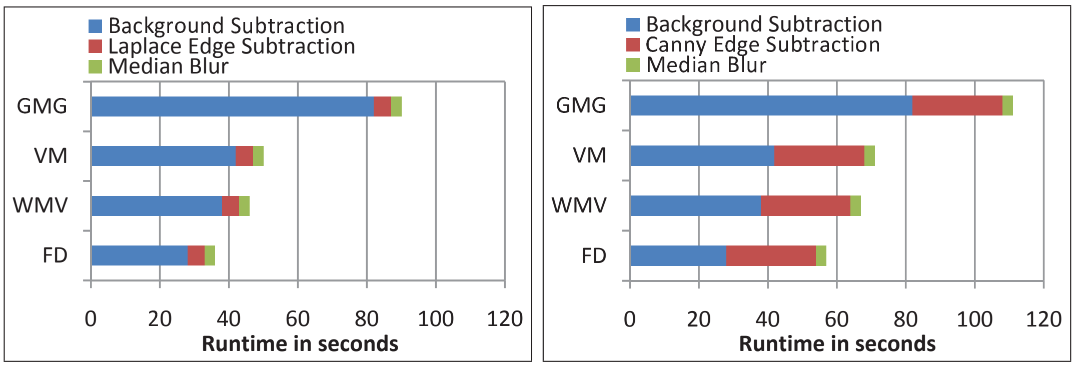

Figure 10.16: Runtime for background subtraction plus edge subtraction

10.7 Conclusion

This chapter presented a novel approach for optimizing background subtraction models with respect to camera motion. It mainly removes edges from the back-

ground, which appear when the camera moves. As a separate module this technique can easily be added to any background subtraction method. Hence, it represents a post-processing step. We found out that background subtraction methods FD, WMV, VM and GMG have a high potential to benefit from this approach. For edge subtraction basically Laplace and Canny edge detectors are used. With regard to accuracy and runtime WMV and Laplace edge detector are the most efficient algorithms. Moreover, the experimental results show that the presented technique outperforms most of the analyzed state of the art techniques stating to be robust against camera motion.
Furthermore, foreground objects detected by edge subtraction have a potential for accuracy improvement by additional segmenting. This aspect remains a task for future research.

Chapter 11

Summary and Insights

This research work at hand explained and explored a novel video classification system. The premise for this system is the occurrence of recurring movements in videos, so that frequency features of these movements can be extracted and applied for classification. For the setup of this video classification approach we considered advantages and drawbacks of state of the art techniques.

11.1 Major Insights

Since the video classification system has a modular structure, each module is tested by differing algorithmic concepts in order to find the most accurate one. For instance, the classifier module can be realized via RBC, SVM or ANN. Moreover, the radius based classifier (RBC) was first introduced by Ayyildiz and Conrad in [5] and turned out as an efficient classifier during the experimental analysis. A further technique, that can be widely applied in computer vision, is the edge subtraction algorithm presented in chapter 10 and published in [11]. Edge subtraction can optimize foreground masks detected by background subtraction clearly, when videos are recorded by a moving camera.
Depending on the recording quality and database the system can work highly accurate. For example, in chapter 9 six classifiers are compared to each other and the KNN classifier achieves a maximal accuracy of 0.92 for own videos. This means 184 videos out of 200 videos are assigned correctly. For external videos the ANN classifier shows up a maximal accuracy of 0.45. Concerning edge subtraction another central research result is presented in chapter 10: Here edge subtraction leads to a significant precision increase for a fixed recall value, when WMV as background subtraction algorithm is applied. The accuracy rises from 0.35 to 0.46, which means an accuracy difference of 0.11.
A further insight of this research study is, that average amplitudes of frequency intervals (AAFI) work more accurate than main frequencies (MF) as feature vectors. Additionally, transforming two 1D-functions is more effective than transforming one 2D-function. Frequency features are highly robust against a series of camera presets and recording conditions. As experiments in chapter 6 show, frequency features are robust against frame reflection, camera rotation, zooming, object translation, object occlusion, input data distortion and time shift. In chapter 7 filters are applied to 1D-functions in order to improve the transformation process. But experimental results show that the system's accuracy is increased

only in some cases, but in general the accuracy is decreased. So applying filters to motion trajectories is not recommended. Furthermore, in chapter 8 several transforms are compared and as a result the fast Fourier transform (FFT) stands out as the most accurate transformation algorithm.
So concluding it can be stated that the presented approach works accurate and is robust. But there is also a major weak point of the approach. When there is an activity class with strongly varying motion frequencies, the frequency features will probably be assigned to false classes. This challenge can be overcome by increasing the number of frequency features in the referenced database.

11.2 Module Combination

Another aspect is the combination of modules. The accuracy of each module depends also on other modules applied for the classification. So the most accurate module for one classification step can change, when preceding or following modules are exchanged. For instance, with own videos we receive an accuracy of 0.92, if we combine following modules:

Module type	**Module name**
Background subtraction	Frame Differencing
Image moment	Centroid
1D-function	Direction
Filter	Average
Transform	FFT
Feature	AAFI
Classifier	RBC

Table 11.1: Optimal module combination

This combination of modules leads to the highest overall accuracy for the system, if we apply Frame Differencing as background subtraction model. If we replace the background subtraction model by a module, which fills out the interior area of the foreground object more completely, the variance values will become even more decisive in some cases. On the other side, the centroid trajectory will be less reliable, because the centroid will not only represent the most active areas in the scene but also areas of the foreground object, which are less active. This can have an effect on the accuracy of different 1D-functions based on location, direction and speed information. It is possible that location information of centroids becomes more reliable even for own videos.
In chapter 7 the filter analysis is conducted with AAFIs. Feature vectors based on AAFIs are more sensitive to filters than feature vectors based on MFs, because AAFIs represent the whole frequency range. Applying MFs could reveal different accuracy levels for each filter, because in this case the feature vector is only affected, when single, clear peaks are reduced or emphasized. For instance, the Laplace filter emphasizes important peaks but also noise. In the case of MFs this noise could easily be ignored by selecting only peaks above a certain amplitude. Another option for the module combination would be the usage of the FWT

instead of the FFT. A transform via FWT reveals more peaks at a wider range. Here again filtering will reveal differing accuracies.
The transform module works for the most part independently from the other modules. Exchanging other modules of the classification process will not have an effect on the basic insights of chapter 8. Only the motion in the video influences the accuracies of the different transforms. In chapter 9 accuracies for the different classifiers are very close. So changing the preceding modules could change the most performant classifier. Especially for MFs as features the results will differ, since feature vectors based upon MFs have a smaller dimensionality. Maybe with MFs the SVM classifier could reveal a better performance for external videos.
As a consequence we can conclude that the most performant module in each chapter has to be considered in the context of the surrounding modules. This is the reason why we perform a combined analysis for image moments, 1D-functions and feature vectors in section 4.4.2 and also combine filters and 1D-functions in section 7.4.3.

11.3 Future Work

For future work it remains an open issue to recognize different activities inside a single video. This would require a further pre-processing step, which divides the original video frames into subframes. Afterwards, these subframes can be analyzed separately. In order to ensure that detected motion areas do not belong to the same motion, the algorithm can apply distances or frequency differences between active areas. Also adapting and analyzing the presented approach for real time action recognition stays an open task. A surveillance system, for instance, based on our technique can be triggered by activity in the scene. As soon as there is activity the system is able to repeatedly analyze short time windows. Here the length of these time windows is important. On the one side short time windows ensure that there are less frames without activity and that only one action type is considered. On the other side long time windows with many repetitions of one movement can reveal more decisive frequency domains.

Bibliography

[1] C. Abdelkader, R. Cutler, and L. Davis. Motion-based recognition of people in eigengait space. *International Conference on Automatic Face and Gesture Recognition*, pages 267–277, 2002.

[2] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *Transactions on Computers*, pages 90–93, 1974.

[3] M.C. Allmen and C.R. Dyer. Cyclic motion detection using spatiotemporal surfaces and curves. *International Conference on Pattern Recognition*, pages 365–370, 1990.

[4] Y. Alsultanny and N. Shilbayeh. Examining filtration performance on remotely sensing satellite images. *WSEAS International Conference on Speech, Signal and Image Processing (SSIP)*, pages 75–80, 2001.

[5] K. Ayyildiz and S. Conrad. Video classification by main frequencies of repeating movements. *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, 2011.

[6] K. Ayyildiz and S. Conrad. Analyzing invariance of frequency domain based features from videos with repeating motion. *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 659–666, 2012.

[7] K. Ayyildiz and S. Conrad. Analyzing transformation of 1d-functions for frequency domain based video classification. *World Academy of Science, Engineering and Technology (WASET)*, pages 921 – 927, 2012.

[8] K. Ayyildiz and S. Conrad. Video classification by partitioned frequency spectra of repeating movements. *World Academy of Science, Engineering and Technology (WASET)*, pages 154–159, 2012.

[9] K. Ayyildiz and S. Conrad. Classifier comparison for repeating motion based video classification. *International Symposium on Visual Computing (ISVC)*, pages 725–736, 2013.

[10] K. Ayyildiz and S. Conrad. Applying filters to repeating motion based trajectories for video classification. *International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 25–32, 2015.

[11] K. Ayyildiz and S. Conrad. Optimized background subtraction for moving camera recordings. *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2015.

[12] R. Babu and K. Ramakrishnan. Compressed domain human motion recognition using motion history information. *International Conference on Image Processing*, pages 321–324, 2003.

[13] F. Baf, T. Bouwmans, and B. Vachon. Type-2 fuzzy mixture of gaussians model: Application to background modeling. *International Symposium on Visual Computing (ISVC)*, pages 772–781, 2008.

[14] O. Barnich and M. Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *Transactions on Image Processing (TIP)*, pages 1709–1724, 2011.

[15] M. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan. Fully automatic facial action recognition in spontaneous behavior. *International Conference on Automatic Face and Gesture Recognition*, pages 223–230, 2006.

[16] F. Bashir, A. Khokhar, and D. Schonfeld. View-invariant motion trajectory-based activity classification and recognition. *Multimedia Systems*, pages 45–54, 2006.

[17] S. Berrabah, G. Cubber, V. Enescu, and H. Sahli. Mrf-based foreground detection in image sequences from a moving camera. *International Conference on Image Processing (ICIP)*, pages 1125–1128, 2006.

[18] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. *Communications on Pure and Applied Mathematics*, pages 141–183, 1991.

[19] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *Transactions on Pattern Analysis and Machine Intelligence*, pages 257–267, 2001.

[20] D. Brogioli, I. Frohne, J Ballinger, J. Tan, M. Hale, R. Goncalves, S. Martin-Michiellot, and W. Springer. Jsci - a science api for java. `jsci.sourceforge.net`, 2012.

[21] J. Canny. A computational approach to edge detection. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 679–698, 1986.

[22] C. Chang and C. Lin. Libsvm: A library for support vector machines. `csie.ntu.edu.tw/~cjlin/libsvm`, 2013.

[23] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, pages 1155–1178, 2007.

[24] X. Chen, D. Schonfeld, and A. Khokhar. Robust null space representation and sampling for view-invariant motion trajectory analysis. *Computer Vision and Pattern Recognition*, pages 1–6, 2008.

[25] F. Cheng, W. Christmas, and J. Kittler. Periodic human motion description for sports video databases. *International Conference on Pattern Recognition (ICPR)*, pages 870–873, 2004.

[26] M. Claesen, F. Smet, J. Suykens, and B. Moor. Fast prediction with svm models containing rbf kernels. *Computing Research Repository*, 2014.

[27] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, pages 297–301, 1965.

[28] T. Cover and P. Hart. Nearest neighbor pattern classification. *Transactions on Information Theory*, pages 21–27, 1967.

[29] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video streams. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 1337–1342, 2003.

[30] R. Cutler and L. Davis. Robust real-time periodic motion detection, analysis, and applications. *Transactions on Pattern Analysis and Machine Intelligence*, pages 781–796, 2000.

[31] R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. *International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 416–421, 1998.

[32] J. Davis and S. Taylor. Analysis and recognition of walking movements. *International Conference on Pattern Recognition*, pages 315–318, 2002.

[33] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, pages 1151–1163, 2002.

[34] A. Elqursh and A. Elgammal. Online moving camera background subtraction. *European Conference on Computer Vision (ECCV)*, pages 228–241, 2012.

[35] L. Estrozi, L. Rios-Filho, A. Bianchi, R. Cesar, and L. da Fontoura Costa. 1d and 2d fourier-based approaches to numeric curvature estimation and their comparative performance assessment. *Digital Signal Processing*, 2003.

[36] C. Fanti, L. Zelnik-Manor, and P. Perona. Hybrid models for human motion recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1166–1173, 2005.

[37] B. Fino and V. Algazi. Unified matrix treatment of the fast walsh hadamard transform. *Transactions on Computers*, pages 1142–1146, 1976.

[38] G. Foody. Relating the land-cover composition of mixed pixels to artificial neural network classification output. *Photogrammetric Engineering and Remote Sensing*, pages 491–499, 1996.

[39] The Apache Software Foundation. Commons math: The apache commons mathematics library. `commons.apache.org/math`, 2012.

[40] M. Frigo. A fast fourier transform compiler. *Special Interest Group on Programming Languages (ACM SIGPLAN Notices)*, pages 642–655, 2004.

[41] K. Glette, T. Gruberand, P. Kaufmann, J. Torresen, B. Sick, and M. Platzner. Comparing evolvable hardware to conventional classifiers for electromyographic prosthetic hand control. *NASA/ESA Conference on Adaptive Hardware and Systems*, pages 32–39, 2008.

[42] A. Godbehere, A. Matsukawa, and K. Goldberg. Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. *American Control Conference (ACC)*, pages 4305–4312, 2012.

[43] Y. Goyat, T. Chateau, L. Malaterre, and L. Trassoudaine. Vehicle trajectories evaluation by static video sensors. *Intelligent Transportation Systems Conference (ITSC)*, pages 864–869, 2006.

[44] Q. He and C. Debrunner. Individual recognition from periodic activity using hidden markov models. *Workshop on Human Motion*, pages 47–52, 2000.

[45] T. Hill and P. Lewicki. *Statistics: Methods and Applications*. StatSoft, 2006.

[46] M. Hofmann, P. Tiefenbacher, and G. Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 38–43, 2012.

[47] J. Honec, P. Honec, P. Petyovský, S. Valach, and J. Brambor. Parallel 2-d fft implementation with dsps. *International Conference on Process Control*, 2001.

[48] M. Hu. Visual pattern recognition by moment invariants. *Transactions on Information Theory*, 1962.

[49] Z. Huang and J. Leng. Analysis of hu's moment invariants on image scaling and rotation. *International Conference on Computer Engineering and Technology (ICCET)*, 2010.

[50] O. Hunt and R. Mukundan. A comparison of discrete orthogonal basis functions for image compression. *Image and Vision Computing New Zealand*, pages 234–245, 2004.

[51] C. Hwang. Inverse vening meinesz formula and deflection-geoid formula: applications to the predictions of gravity and geoid over the south china sea. *Journal of Geodesy*, 1998.

[52] H. Kekre, S. Thepade, and A. Maloo. Performance comparison of image retrieval using fractional coefficients of transformed image using dct, walsh, haar and kekres transform. *International Journal of Image Processing (IJIP)*, pages 142–155, 2010.

[53] H. Lee and J. Kim. An hmm-based threshold model approach for gesture recognition. *Transactions on Pattern Analysis and Machine Intelligence*, pages 961–973, 1999.

[54] J. Lee. Digital image enhancement and noise filtering by use of local statistics. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 165–168, 1980.

[55] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikäinen, and S. Li. Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1301–1306, 2010.

[56] R. Lienhart. Indexing and retrieval of digital video sequences based on automatic text recognition. *ACM International Conference on Multimedia*, pages 419–420, 1996.

[57] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *Transactions on Image Processing (TIP)*, pages 1168–1177, 2008.

[58] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[59] S. Martucci. Symmetric convolution and the discrete sine and cosine transforms. *Transactions on Signal Processing*, pages 1038–1051, 1994.

[60] G. Mastin. Adaptive filters for digital image noise smoothing: An evaluation. *Computer Vision, Graphics, and Image Processing (CVGIP)*, pages 103–121, 1985.

[61] LLC Media. Alfamovie. `www.alfamovie.com`, 2010.

[62] Q. Meng, B. Li, and H. Holstein. Recognition of human periodic movements from unstructured information using a motion-based frequency domain approach. *Image and Vision Computing (IVC)*, pages 795–809, 2006.

[63] M. Narayana, A. Hanson, and E. Learned-Miller. Coherent motion segmentation in moving camera videos using optical flow orientations. *International Conference on Computer Vision (ICCV)*, pages 1577–1584, 2013.

[64] Neuroph. Java neural network framework. `neuroph.sourceforge.net`, 2013.

[65] J. Niebles and L. Fei-fei. A hierarchical model of shape and appearance for human action classification. *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[66] S. Noh and M. Jeon. A new framework for background subtraction using multiple cues. *Asian Conference on Computer Vision (ACCV)*, pages 493–506, 2012.

[67] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 831–843, 2000.

[68] S.A. Orange. Dailymotion. `www.dailymotion.com`, 2010.

[69] N. Patel and I. Sethi. Audio characterization for video indexing. *SPIE on Storage and Retrieval for Still Image and Video Databases*, pages 373–384, 1996.

[70] S. Pei and F. Chen. Semantic scenes detection and classification in sports videos. *Conference on Computer Vision, Graphics and Image Processing*, pages 210–217, 2003.

[71] R. Polana and A. Nelson. Detection and recognition of periodic, nonrigid motion. *International Journal of Computer Vision*, pages 261–282, 1997.

[72] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood. View-invariant alignment and matching of video sequences. *International Conference on Computer Vision*, pages 939–945, 2003.

[73] M. Roach, J. Mason, and M. Pawlewski. Video genre classification using dynamics. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1557–1560, 2001.

[74] M. Rocamora and P. Herrera. Comparing audio descriptors for singing voice detection in music audio files. *Brazilian Symposium on Computer Music*, 2007.

[75] S. Saeedi, A. Moussa, and N. El Sheimy. Vision-aided context-aware framework for personal navigation services. *International Society for Photogrammetry and Remote Sensing*, pages 231–236, 2012.

[76] H. Sarukhanyan, S. Agaian, K. Egiazarian, and J. Astola. Reversible hadamard transforms. *International Journal on Electrical Energy*, pages 309–330, 2007.

[77] S. Shaikh, K. Saeed, and N. Chaki. Moving object detection using background subtraction. *International Conference on Signal Processing, Image Processing and Pattern Recognition (SIP)*, pages 15–23, 2014.

[78] M. Sideris and B. She. A new, high-resolution geoid for canada and part of the us by the 1d-fft method. *Bulletin Geodesique*, 1995.

[79] V. Singh, S. Gupta, and U. Dalal. Performance comparison of discrete hartley transform (dht) and fast fourier transform (fft) ofdm system in awgn channel. *International Journal of Computer Applications*, 2013.

[80] A. Sobral. BGSLibrary: An opencv c++ background subtraction library. *Workshop de Visão Computacional (WVC)*, 2013.

[81] A. Sobral, L. Oliveira, L. Schnitman, and F. De Souza. Highway traffic congestion classification using holistic properties. *IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, 2013.

[82] K. Soomro, A. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *Center for Research in Computer Vision (CRCV-TR-12-01)*, 2012.

[83] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2246–2252, 1999.

[84] Y. Sugaya and K. Kanatani. Extracting moving objects from a moving camera video sequence. *Memoirs of the Faculty of Engineering, Okayama University*, pages 279–284, 2004.

[85] P. Tsai, M. Shah, K. Keiter, and T. Kasparis. Cyclic motion detection. *Pattern Recognition*, pages 1591–1603, 1994.

[86] I. Tsang, J. Kwok, P. Cheung, and N. Cristianini. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, pages 363–392, 2005.

[87] C. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[88] R. Vanithamani, G. Umamaheswari, and M. Ezhilarasi. Modified hybrid median filter for effective speckle reduction in ultrasound images. *International Conference on Networking, VLSI and Signal Processing (ICNVS)*, pages 166–171, 2010.

[89] L. Vliet, I. Young, and G. Beckers. A nonlinear laplace operator as edge detector in noisy images. *Computer Vision, Graphics, and Image Processing (CVGIP)*, pages 167–195, 1989.

[90] E. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing and Management*, pages 465–476, 1986.

[91] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer vision and image understanding*, pages 249–257, 2006.

[92] W. Wong, W. Siu, and K. Lam. Generation of moment invariants and their uses for character recognition. *Pattern Recognition Letters (PRL)*, pages 115–123, 1995.

[93] LLC YouTube. Youtube: Broadcast yourself. `www.youtube.com`, 2010.

[94] C. Zhang, S. Chen, M. Shyu, and S. Peeta. Adaptive background learning for vehicle detection and spatio-temporal tracking. *Pacific-Rim Conference on Multimedia (PCM)*, pages 15–18, 2003.

[95] S. Zhang, H. Yao, and S. Liu. Dynamic background subtraction based on local dependency histogram. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, pages 1397–1419, 2009.

[96] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *International Conference on Pattern Recognition*, pages 28–31, 2004.

List of Figures

List of Tables