Extraction, Evaluation and Selection of Motion Features for Human Activity Recognition Purposes

SEBASTIAN BRÄNNSTRÖM

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2006

Extraction, Evaluation and Selection of Motion Features for Human Activity Recognition Purposes

S E B A S T I A N B R Ä N N S T R Ö M

Master's Thesis in Computer Science (20 credits)
at the School of Engineering Physics
Royal Institute of Technology year 2006
Supervisor at CSC was Henrik Christensen
Examiner was Henrik Christensen

TRITA-CSC-E 2006:028
ISRN-KTH/CSC/E--06/028--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se

Abstract

Human activity recognition is a necessary part of a socially interacting robot. This can be done using a wide range of methods. One common such is the use of an inference engine which is trained by providing examples of desired activities. The success of this approach is highly dependent on the type of data used for this training.

Modern three dimensional tracking systems allow us to use articulated models to describe the human body. While it is possible to train an inference engine using these models, a large number of training examples is commonly required. One way to diminish this requirement is to use our background knowledge to extract more abstract features from the model.

In this thesis a system is developed which extracts more than 100 features from the motion of a 3-D human body model provided by an external system. The features are then evaluated using Mutual Information analysis. An optimal subset of features is selected and used to train a Neural Network.

The results show that abstract feature extraction is of significant use in an activity recognition system; that feature selection is sensible and that it can be successfully done using statistical methods; and that recognition rates well over 90% can be achieved over a wide range of activitites with only a small number of examples given carefully extracted and selected features.

Utvinning, utvärdering och urval av kännetecken hos rörelser för igenkänning av mänskliga aktiviteter

Examensarbete

Sammanfattning

Igenkänning av mänskliga aktiviteter är en nödvändig funktion hos en social robot. Detta kan åstadkommas med ett antal olika metoder. En vanlig sådan är användandet av en inferensmotor som tränas med exempel på de önskade aktiviteterna. Resultatet av denna metod beror till stora delar på vilken sorts data som används vid träningen.

Moderna tredimensionella spårningssystem möjliggör användandet av artikulerade modeller för att beskriva den mänskliga kroppen. Medan det är möjligt att träna en interferensmotor direkt med en sådan modell krävs vanligen ett stort antal exempel. En metod att minska detta krav är att använda vår kunskap om mänskliga rörelser för att utvinna mer abstrakta kännetecken (features) för olika aktiviteter.

I detta examensarbete beskrivs ett system som utvinner mer än 100 kännetecken från en tredimensionell kroppsmodell. Dessa kännetecken utvärderas sedan med hjälp av transinformationsanalys (mutual information). En optimal delmängd kännetecken väljs ut och används för att träna ett artificiellt neuronnät.

Resultaten visar att utvinning av abstrakta kännetecken är av betydande användning för igenkänningssystem, att urval av kännetecken är meningsfullt och att det kan åstadkommas med statistiska metoder, samt att en igenkänningsgrad på långt över 90% kan åstadkommas för en stor mängd aktiviteter med endast ett litet antal träningsexempel givet noggrant utvunna och utvalda kännetecken.

Foreword and Acknowledgements

This report is presented as a Master's Thesis at the School of Computer Science and Communication at the Royal Institute of Technology (KTH) in Stockholm, Sweden.

The work was commissioned by the Institut für Technische Informatik at Universität Karlsruhe (UKA) in Karlsruhe, Germany. I want to thank Professor Henrik I. Christensen at KTH for being my examiner and for providing me with the opportunity to do this project, as well as Professor Dr. Rüdiger Dillmann at UKA for generously accepting me at his institute. I also want to thank Steffen Knoop and Stefan Vacek, both at UKA, for supervising the work and giving invaluable advice and feedback throughout the project.

Sebastian Brännström, March 2006

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter contains the background for the subject matter, describes the problem to be solved, and gives the outline for the rest of the report.

1.1 Background

Human beings communicate to a large extent through physical movement of their limbs. Many activities can be readily recognized just by observing the motion of the limbs of the human body, or even the motion of the entire body.

How one person interacts with another is in many cases highly dependent on our observation of what the other person is doing. For example, we do not try to shake hands with a person running quickly past us.

When designing a socially interacting system such as a service robot, it is desirable to try to mimic this ability. Not only does activity recognition capabilities greatly improve the ability of the robot to interact, but it also tends to increase the level of comfort in the people with which it interacts.

Apart from transmitting important activity information, limb movements are an important accessory when communicating. The motions known as *gestures* generally make up a large part of the information flow when two people are talking to each other. Similarly for a robot, accurate activity recognition could provide a helpful context for its speech recognition and task planning systems.

It is therefore clearly desirable for a socially interacting robot to be able to interpret observed motions, and recognize the activities that underlie them.

1.2 Problem Definition

The problem underlying this thesis is to construct an activity recognition system and evaluate its effectiveness. Specifically, assuming existing tracking and segmentation of the human body into a model consisting of generalized cylinders, the problem is to create an adaptive machine learning system that performs reliable recognition of various human activities. The recognition should be based solely on the motion of the cylinders of the model over time and use an inference engine to allow training of new activities.

Since a tracking system is already available, and good inference engines exist as ready-to-use packages, the core of the work will be to extract sensible *features* from the model, evaluate their effectiveness, and automatically select a proper subset of features that reliably allows the inference engine to classify an activity as one of previously trained set of activities.

The activities to be classified are mainly larger scale motions, and only activities that can be recognized purely by observing the motion of the human body. Examples of such activities include walking, sitting and waving.

1.3 Project Overview

Since activity recognition is a thriving research area, a large amount of knowledge was to be found in the literature. The initial literature review gave a good overview of the field, and suggested several working approaches to the problem. Since a working state-of-the-art motion tracking and modeling system had recently been developed at the Universität Karlsruhe where this work was done, it seemed natural to make use of this system.

This placed the focus of the project on the actual feature recognition, in other words the choice of a machine learning system, and the training and classification problems associated with such a system. The choice here was narrowed down by the software availability, as well as the author's own knowledge.

Naive attempts to train a machine learning system using the data obtained directly from the motion tracking system turned out to produce convincing but not impressive results, which turned to focus of the project more towards methods to extract better and more abstract *features* from the tracking data. This turned out to be relatively simple, and results improved considerably using such methods.

It quickly became difficult to assess which features were actually helping us classify an activity, and focus then again moved, this time towards methods to evaluate the effectiveness of abstract features, and to methods to select good subsets of features using this evaluation.

The last part of the project consisted of the creation of a good set of experiments for the above methods, and of good testing data to be used in the experiments. Significant analysis of the data obtained was done, and conclusions drawn.

1.4 Report Outline

This report is arranged as follows:

Chapter 2 – Activity Recognition Theory describes the process of human activity recognition, and covers important concepts and techniques necessary for the thesis.

Chapter 3 – Feature Extraction in detail describes the extraction of features from body model data.

Chapter 4 – Feature Evaluation and Selection covers methods to evaluate and select a subset of features for training and classification.

Chapter 5 – Implementation describes details of the chosen implementations of the various parts.

Chapter 6 – Experiments and Results covers the experiments made and states the results achieved using the system implemented in chapter 5.

Chapter 7 – Summary and Conclusions summarizes this report, presents the conclusions drawn from the experiments and suggests possible future improvements.

Chapter 2

Background

In order to provide a framework for the thesis we shall start in this chapter by looking at the field of activity recognition from above. This will serve as an introduction to the field as well as provide solid footing for the main work.

2.1 Overview

Activity recognition is a highly active research field and the amount of published material is immense. Good starting points to the area are the comprehensive surveys composed by Cédras and Shah [8] and by Gavrila [11], and the slightly shorter review by Aggarwal and Cai [1]. Additionally, Wang, Hu and Tan [39] cover some of the work done after 2000.

Largely, activity recognition has been treated as a pattern recognition problem, where the aim is to identify an observed motion pattern with previously observed patterns. This has come primarily from the simplicity of the approach and the relatively advanced state of modern pattern recognition techniques, and this will be the approach used in this thesis.

The general trend has been to transfer successful techniques from the typically simpler problem of speech recognition to the more difficult problem of activity recognition. The impressive results within that field and its rapid convergence to a handful of techniques suggests similar approaches when designing activity recognition systems.

2.2 Activity Recognition Systems

In order to provide a framework for the theory covered in this chapter, we shall start by taking a look at how a typical activity recognition system works.

Activity recognition is commonly divided into several independent or weakly dependent sub-tasks. For most work both the overall solution as well as these sub-tasks are very similar, however as is common in rapidly advancing research fields, a lot of different terms are used by different authors to describe essentially the same things.

In this report we shall divide the process into three tasks. The first is *motion capture*, which covers the entire problem of observing a human subject and obtaining a digital representation of their motion. The second is *motion analysis*, where the motion capture data is processed to make it suitable for the third step, the actual *recognition* itself.

In the rest of this chapter we shall take a look at various methods used to handle each of these problems.

2.3 Motion Capture

The first step in an activity recognition system is the capturing of the human body pose as it moves over time. Somehow the activities of the target must be tracked, digitized and stored in a suitable format for processing. This task is commonly again divided into several sub-tasks, most notably *tracking* of the body over time and *model fitting* to the tracking data. Sometimes *detection* of a human subject is added as a first step, but for this thesis it is considered part of the tracking step.

Motion capture is a very large research field in itself. Previous work can generally be separated into 2-D and 3-D approaches depending on the type of body model used. While 3-D models have the obvious advantage of correctly representing all real-world dimension, typically they make reliable tracking much more difficult. Many successful recognition attempts have therefore made use of 2-D models.

2.3.1 Tracking

Tracking techniques have evolved together with the motion capture technologies. Using our knowledge of how the human eye works, the most obvious tracking technology is visual tracking by means of a video camera. The relatively large amount of signal bandwidth and data processing needed to accomplish this has historically made this approach impractical.

Early work therefore tended to seek alternative methods. One of the most cited is the Moving Light Display technique used by Johansson [18] which uses lights attached to body parts to track a person in 2-D.

With the availability of high quality digital video cameras and sufficiently fast processors, the bulk of the work within tracking systems within the last few years have used this approach. The first commonly cited work on visual human motion track-

ing is that of David Hogg[15], who used a model consisting of 14 elliptical cylinders to visually track a walking person.

A significant problem with visual tracking is that of occlusion, both by objects between the camera and the acting person, but equally significant so called self-occlusion, when body parts occlude one another. This can to some extend be alleviated by applying knowledge of the mechanics of the human body a so called body model.

Many commercial systems today exist to achieve highly accurate 3-D tracking, generally by making use of sensors attached to the body, such as *data gloves* and similar. Other capture devices include the commercial *CSEM Swiss Ranger*, which combines an ordinary CCD with a time-of-flight sensor system using IR diodes to obtain depth data.

Finally, attempts have been made to use more than one sensor to improve accuracy. Tracking using several cameras is for instance attempted by Gavrila and Davis [12].

2.3.2 Model Fitting

In order to fit a model to tracking data, the amount of information to be stored in the model must first be chosen, in other words the detail level. A wide range of body model detail might be considered, the simplest being a single point in space. The only information we can extract from this model is position and velocity, and even calling it a body model may be stretching that very definition. The importance of this course information should however not be underestimated, as we can indeed identify several activities from it.

Full body activity recognition without articulated models has however not proved to be reliable, partly because they make tracking intrinsically more difficult (it is harder to identify movement of something you do not know what it looks like). Focus has therefore generally moved towards more complete models. An exception to this is the field of sign-language recognition, where model-less approaches have shown convincing results, see for instance the work by Starner and Pentland [38].

A radically more detailed model is the stick-figure, where the length of the human limbs, and their relative angles are stored within the model. Strong motivation for this type of model is the curious fact that we humans tend to have no problems identifying activities carried out by a stick figure. Good examples of stick-figure work includes that of Guo *et al.* [13], who worked with monocular images, and that of Chen and Lee [9] who used line segments and joints to analyze the gait of walking people for identification purposes.

Even more refined models also try to describe the human flesh, using ribbons such as splines in 2-D, or spheres, cylinders, superquadrics or polygon meshes in 3-D, with increasing detail. This has the benefit of allowing more detailed tracking, however

it almost always comes at the cost of increased processing demands. Naturally with constantly increasing hardware capabilities, this type of model has become more common in the last few years.

In general, the recovery of a 3-D pose is a difficult task, depending on the tracking system used. For primitive visual tracking the Bayesian particle filtering techniques used by Sidenbladh, Black and Fleet [37] represent current state-of-the-art, while for 3-D point cloud type data, the Iterative Closest Point (ICP) methods used by Knoop *et al.* [20] show impressive results.

While 3-D models may appear superior, and in many cases are, the simple fact that humans appear to have no problem to identify activities from 2-D data lends credibility to recognition systems that do not model three dimensions.

2.4 Motion Analysis

Following the capture and modeling steps, we find ourselves with an unwieldy set of tracking data that is largely unsuitable for recognition purposes. Two subjects are extremely unlikely to perform the same action identically, and even for a single subject, motions tend to vary significantly between performances.

Nevertheless, humans are clearly able to identify an activity even when great variety is made within motions. Also, the speed at which the activity is done is largely unimportant within reasonable bounds. This suggests that it might be possible to find abstractions that given tracking data can somehow yield the essence of an activity.

There are two main approaches to this problem, segmentation of the activities into what is commonly called *motions primitives*, and the more abstract *feature extraction* method.

2.4.1 Motion Primitive Decomposition

The Motion Primitives approach attempts to model activities by dividing them into brief motions which can then be uniquely identified and transcribed. A fundamental problem here has been the lack of a notation, i. e. a vocabulary and grammar which can describe motion. The general idea here has been to try to segment actions into atomic motions, commonly called *primitives* or *movemes*. There is biological evidence that this approach is to some extent used by animals.

Pioneers within this field were Badler and Smoliar [4], who defined eight desirable features of a such a notation. Noting that similar systems are used within the art of dance they looked at two popular notations, the so-called Labanotation (after Rudolf Laban), and Eshkol-Wachmann. They then went on and devised a complete system for recreating human motion based on Labanotation and a network of "pro-

cessors" in each joint of the human body. Using information from the Labanotation, convincing human motion was then successfully recreated.

The reverse process, encoding observed motion into a compact notation is a more difficult problem. The first problem is segmentation into atomic motions, which itself appears to be far from trivial and has been generally done manually by letting human observers agree on cutting points. Automated segmentation has been attempted for instance by Fod, Matarič and Jenkins [10] by looking at zero-crossings of various functions of angles. Ali and Aggarwal [2] trained a machine learning system using manually selected samples of good breakpoints, and obtained convincing results.

For the motion primitive analysis, the main idea so far has been to use combinations of dimensionality reduction, clustering and interpolation. Inamura *et al.* [16] use Probabilistic Neural Networks to create generalized patterns from manually segmented motions while Jenkins and Matarič [17] develop a technique they call Isomap to solve a similar situation.

A slightly different approach to the subject has come from the field of *motion synthesis*. Noteworthy are the works by Li, Wang and Shum [27] on "Motion Texture" and the "Verbs and Adverbs" approach by Rose, Bodenheimer and Cohen [35].

While the idea behind motion primitives is sound and even elegant, no convincing method exists today by which primitives can be reliably and reproducibly extracted from motion capture data. Hence we must look at other methods to analyze activities.

2.4.2 Feature Extraction and Selection

The other main approach is to avoid the decomposition problem and instead look at various abstract *features* that we may observe. The idea here is that while activities may be carried out slightly different by different individuals, there is still a large amount of overlap which can be used to infer an abstract description of an activity.

Compared to the motion primitives approach which attempts to identify the structure of an activity, feature extraction tries to model what an activity looks like to the observer. Recent work that make us of this method is that of Mori *et al.* [29], who uses abstract features selected by human observers as characteristic for an activity to achieve very reliable recognition.

The main benefit of this approach is that it is relatively simple to extract a large amount of abstract time signals from activities. From plots of these signals an intelligent observer may often readily identify an activity based on the properties of the plots. However, this may not be as trivial for a machine learning system, given that most signals are generally nothing but noise. This calls for a powerful *feature selection* system as an extra processing step.

In this thesis we shall look closely at this approach, and methods to extract strong features are discussed in detail in Chapter 3 , while the selection problem is covered in Chapter 4.

2.5 Recognition

The final step of the activity recognition chain is the actual recognition itself. This is commonly done by means of a *classifier*.

A classifier is typically trained by feeding it tuples of data consisting of an instance of data to be trained, and the appropriate class to which it belongs. The querying of a classifier works similarly, except here no class is provided as input data. An instance is fed to the classifier, which then returns which of the trained classes the new instance belongs to (if any). Additionally, a probability is commonly provided as well.

A very important part of a good classifier is the ability to generalize from input data. Instead of just learning the very samples given, it tries to generalize the the samples into a concept.

There is an abundance of classifiers in the literature, with various properties that make them suitable for various tasks. Many classifiers share a few common traits: The training step is generally slow, while the classification is comparatively fast.

Classifier designs can be divided into three broad techniques: *template matching*, *inference engines* and *state-space models*.

2.5.1 Template Matching

A straight-forward approach to recognition is that of template matching, also called nearest-neighbor classifiers, where an acquired motion pattern is compared to previously stored reference patterns. It has generally only been applied to 2-D models, specifically by means of optical flow by Polana and Nelson [31] and by means of a devised energy measure by Bobick and Davis [6]. The work on spatio-temporal Isomap by Jenkins and Matarić [17] allows for this technique to be further investigated.

The major objection to the template matching approach is its inherent sensitivity towards noise and larger variation in space and time. Dynamic Time Warping is one technique to eliminate the explicit time dependency of a data sequence while keeping time ordering, covered thoroughly by Myers, Rabiner and Rosenberg [30].

2.5.2 Inference Engines

Inference engines is a broad class of classifiers that work within a *configuration space*, which is simply the abstract n-dimensional space spanned by all possible configurations of input data that may be classified. Within this space, regions are then inferred (hence the name) from the examples given to the inference engine during the training phase by means of a so-called *training function*.

The primordial inference engine is the (Artificial) Neural Network, which is extensively used within almost all areas of machine learning problems. One good example of its use for activity recognition is the work of Guo *et al.* [13], who used Neural Networks to classify stick-figure motion.

A related method is the use of Radial Basis Functions, which were used by Yacoob and Davis [36] to recognize human facial expressions. Another closely related model is the Support Vector Machine, used by Lee and Xu [26] for recognizing table tennis motions from 2-D data.

The main downside with the Neural Network approach is the need for a substantial body of training data in order to find representative boundaries within the configuration space, as well as a relatively slow learning rate. A large amount of improvements to the original Neural Network idea have been proposed over the years, particularly pertaining the training functions used.

2.5.3 State-space Models

Another type of classifiers are so-called *state-space models*, where we move dynamically between individual *states* in the system. One example of such a model is the Hidden Markov Model (HMM). HMMs alleviate explicit time dependency altogether, by introducing a state-space with probabilistic transitions. The model is then trained by using training data to infer appropriate states and to adjust the probabilities of transition between them.

Hidden Markov Models have been applied to speech recognition with impressive results, to the extent that nearly all modern speech recognition packages use HMMs as part of the system. Rabiner covers the theory behind HMMs, and practical usage for speech recognition in his excellent tutorial [32].

Much work within this area has also been done for activity recognition purposes in the last few years. Starner and Pentland [38] use HMMs to classify sign language, while Yamato, Ohya and Ishii [40] attempt to classify tennis strokes. Rigoll, Kosmala and Eickeler [34] use Hidden Markov Models to classify gestures based on 2D features. Campbell and Bobick [7] work with velocity and acceleration within a *phase-space* to achieve good results.

2.5.4 Other Methods

A completely different approach to recognition are the attempts to generate natural language descriptions directly from observed motion. Kojima and Tamura [22] use a hierarchical approach to build such a semantic expression of observed activities.

Finally, hierarchical approaches which rely on recognition at various levels may very well turn out to be superior the single-tier models commonly used today. Some work has been attempted at this, with the aforementioned work done by Mori *et al.* [29] as one example.

2.6 Chapter Summary

In this chapter we discussed how a typical activity recognition system works. With this in mind, we shall now move on to the core of the thesis: Feature extraction, evaluation and selection.

Chapter 3

Feature Extraction

As discussed in Chapter 2, a critical step in order to achieve successful activity recognition is *motion analysis*. In this chapter we look at various methods to do this effectively by means of *feature extraction*, assuming tracking and model fitting has already been done.

3.1 Overview

Since for this thesis we assume the existence of a motion capture and model fitting system, we may assume that we are given a descriptive model of the human body as it moves over time. From this model we can then reconstruct a complete representation of the activity which can easily be recognized by a human observer.

In the previous chapter we discussed two methods to do motion analysis on the tracking data: *motion primitive decomposition* and abstract *feature extraction* and found the latter the only presently feasible method. Therefore we shall devote this chapter to methods by which we can extract good features.

Previous work that makes use of the same principles include that of Mori *et al.* [29], who uses abstract features selected by human observers as characteristic for an activity to achieve very reliable recognition, and that of Ribeiro and Santos-Victor [33] who extract features from a visual tracking system in a similar manner.

3.2 Feature Extraction Principles

According to the our definition from Section 2.4.2, the feature extraction method tries to capture what an activity looks like to an observer. In other words we want to transform visible patterns and features into good signals with which we can train our inference engine.

For a human it is generally quite simple to describe in words what is a telltale feature of an activity. For instance, we may describe the activity *walk* as periodic motion of the legs together with a change in position of the entire body. Hand *clapping* can be described as a symmetric movement of the arms in front of the body. Likewise, we may describe *sit* as being stationary with bent knees and hips.

The problem now is to translate these descriptions into mathematical functions which we can use to generate a signal. It turns out that this is not necessarily very difficult. If we look at the descriptions above, we can quite easily identify properties which we can describe as mathematical properties. Concepts like periodicity, motion and symmetry all have mathematical representations, not to mention the even simpler properties such as position, which we obtain directly from the tracking system.

3.3 Desired Feature Properties

Since there is an almost infinite number of ways to describe an activity, and hence to extract features, in order to limit ourselves to extracting *good* features we must first define which properties a good feature might have.

As the entire point of feature extraction is to simplify *classification*, there are some obvious qualities that a good feature must possess. Namely, a good feature would have to be:

- Characteristic of at least one activity.
- Separable between different activities.
- Reproducible for an activity independent of individual and other variations .

With a general idea how features might be constructed, as well as a few simple criteria by which we may quickly judge a feature we shall look at detailed methods to extract features.

3.4 Feature Types

Using only the definitions above a large set of features could be created using various mathematical techniques. Many of these features are simple variations on other, and in this part of the thesis it is sensible only to describe their general properties, and to try to group them into distinct feature types.

The most important distinction is the type of time dependency a feature carries. Whereas the tracking data and its derivatives are *explicitly time dependent* by being

motion capture data collected on a frame-by-frame basis, the derived features are only *implicitly time dependent*.

3.5 Explicitly Time Dependent Features

All data coming directly from the motion capture system is explicitly time dependent as they come from a frame-by-frame sampling. Additionally, time derivatives of explicitly time dependent functions also belong with this group.

3.5.1 Tracking Data

Many activities are readily recognizable just by observing the angles between specific limbs, and positions of hands, feet and head. The simplest features are therefore the tracking data themselves.

The relations between the limbs should in theory be sufficient to recognize any activity. After all, since the activities are recognizable to a human directly from observing a reconstruction of the tracking data, it should be possible to create a classifier which operates directly on these data.

In general, however, model data is highly irrelevant, and even with the best classifiers of today, results are highly dependent on the data used during training and querying. It seems highly optimistic that a classifier will be able to recognize all the activities with a limited training data set. Experimental data supports this, as seen in the results section of this report.

3.5.2 Time Derivatives

Using simple Euler derivation we may obtain time derivatives from all tracking values. This amounts to finding the velocity of the body, its limbs and the angular velocities.

Many activities have ranges of velocities that may identify them. For instance, while walking and running are to some extent similar, velocity is a telling difference between them.

3.6 Implicitly Time Dependent Features

The implicitly time dependent features are not directly influenced by time in the same sense as the explicitly time dependent features, however still capture the time dimension in one way or another.

3.6.1 Statistics

Statistics is one of the first areas we may look at when we want to find abstractions of our tracking data. The simplest methods we may apply are averaging methods such as the *mean value*, which could be very useful for some activities where mean positions of limbs are significant.

The related *variance* measurement tells us how much a signal varies from its mean. Variance is defined as

$$var(X) = E((X - \mu)^2) \tag{3.1}$$

where $\mu = E(X)$, i. e. the expectancy or mean value of X.

Slightly more advanced methods include covariance, which is a simple statistical measurement of how two variables vary linearly together, in other words their linear dependence. For two random variables X and Y covariance is defined as

$$cov(X, Y) = E((X - \mu) \cdot (Y - \nu)) \tag{3.2}$$

where $\mu = E(X)$ just like above, and $\nu = E(Y)$.

The sign of the covariance function tells us how the variables X and Y relate. A positive covariance indicates that thy tend to have the same direction, i. e. when one increases so does the other. A negative covariance implies the inverse, when one increases the other decreases.

In other words we can use covariance as a measure of *symmetry* between two body parts. Activities where we might find this useful is hand clapping and walking, for which we have symmetric hand positions, and antisymmetric leg motions, respectively.

3.6.2 Principal Component Analysis

Principal Component Analysis (PCA), also known as the Karhunen-Loève Transform is a linear transformation applied to a data set in order to obtain a new orthogonal coordinate system whose first axis lies along the direction of greatest variance in the data set, and whose second axis lies along the direction of the second greatest variance, and so on. These new axes are known as the *principal components*.

PCA is commonly used to reduce dimensionality, by first obtaining the principal components and then discarding the dimensions contributing the least to the variance of the data set. This method is a common solution to the Feature Selection problem covered in the next chapter, however was decided against for this thesis because the linear combinations of features it yields are not desirable.

There is however another area in which we would find PCA useful, namely the determination of planarity or even linearity. If a signal has an exceedingly large portion of its variance in two or one dimensions we can confidently say it is planar or linear.

For activities such as waving, walking and kicking we have a strong planar property in the hand motion, the legs and the feet, respectively, while for hand clapping the hand tend to move linearly and periodically.

3.6.3 The Fourier Transform

A characteristic feature of some activities is a periodic motion of some kind. Being able to identify and classify the most significant frequency of the motion could therefore be a good feature. Naturally, there usually exists a range within which an activity is commonly carried out, but as long as this range is limited a frequency based feature is still sensible.

A mathematical means to determine frequency from a time series is through the Fourier Transform, defined for the time dependent function $f(t)$ as

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt. \tag{3.3}$$

By applying the Fourier Transform we obtain a breakdown of the frequency content $F(\omega)$ of the signal. For periodic activities we will expect a significant peak in the frequency spectrum, and we may use the existence of such a peak as evidence of a periodic activity.

Examples of activities where periodicity might be important include arm waving, walking and hand clapping.

3.6.4 Other

External signals from sensors is another very useful feature source. Examples here include object sensors which would help us tell if the observed human is holding an object in their hands, or contextual information obtained from a speech recognition system. These signals could help resolve ambiguity between activities whose motions are very similar.

As the problem statement in Chapter 1 refers exclusively to motion recognition such signals will not be used in this thesis.

3.7 Chapter Summary

In this chapter we covered methods to extract abstract features from tracking data, the first part of motion analysis. In the next chapter we shall look at methods to evaluate these features and select good subsets of them.

Chapter 4

Feature Evaluation and Selection

With a large set of features produced using the methods outlined in the previous chapter, we now turn to the question of how to evaluate their relevance. Definitions of relevance are provided, as are various methods to estimate it. We then discuss methods to select a subset of relevant features that can be used for classification purposes, and specifically take a look at two algorithms.

4.1 Overview

Given a large feature set designed to capture a wide variety of activities we should expect many features to be irrelevant for any given activity. In practice, these features will contribute nothing but noise to the classifier. By blindly feeding it all our extracted features, we depend on the classifier to be able to filter out the irrelevant features.

Modern classifiers seem to handle this reasonably well. A study done by John *et al.* [19] found that while it feature selection does not necessarily improve predictability for most classifiers, it does improve training speed, and also seems to aid generalization.

There are however several other reasons to select a subset of features. By focusing on a few properties of the activity, it will be easier to provide stable recognition of activities even under heavy occlusion, as long as the features we need can be successfully extracted.

Likewise, having some insight into what is actually being evaluated removes a large part of the "black box" mystery from the activity recognition. This in turn may greatly simplify algorithm selection and parameter tuning, as well as provide a feedback loop between the person whose activities are being recognized.

Based on this it seems sensible to try to select an optimal set of features for each

activity, and use only this set to train and query the classifier.

4.2 Relevance Definitions

In order to evaluate the relevance of a feature, we must first define what we mean by relevant. Since the sole purpose of extracting features is to predict the target class, a vague definition of relevance is *a measurement of the predictability that a feature adds to a sample.*

Mathematically, the most general statement we can make about a relevant feature F_i and a target class C is that for a given class value c and at least one feature value f_i, it satisfies the relation

$$p(C = c|F_i = f_i) \neq p(C = c) \tag{4.1}$$

which simply means that the posterior probability about the class of a sample after observing the feature is different compared to the prior.

Although the above definition seems intuitive, it is possible to find sets of clearly relevant features where it does not hold [19]. It seems necessary to define two relevance classes: *strong relevance* and *weak relevance.*

We say that feature F_i is strongly relevant iff there exists a set S_i of all features except for F_i, i. e. $S_i = \{F_1, \ldots, F_{i-1}, F_{i+1}, \ldots, F_n\}$ and some value assignments f_i, c and s_i for which $p(F_i = f_i, S_i = s_i) > 0$ such that

$$p(C = c|F_i = f_i, S_i = s_i) \neq p(C = c|S_i = s_i). \tag{4.2}$$

We say that F_i is weakly relevant iff it is not strongly relevant, and there exists a subset S'_i of S_i and some f_i, c and s'_i with $p(F_i = f_i, S'_i = s'_i) > 0$ such that

$$p(C = c|F_i = f_i, S'_i = s'_i) \neq p(C = c|S'_i = s'_i). \tag{4.3}$$

Strong relevance implies that a feature can not be removed without loss of prediction accuracy. Weak relevance implies that a feature can not always be removed without loss of prediction accuracy, depending on which subset of features it is used together with. In other words, weak relevance generally implies that a feature overlaps at least one other feature in the subset S_i.

4.3 Evaluation Methods

While the definitions of equations 4.2 and 4.3 help us formalize the concept of feature relevance, they do not tell us how to find the probabilities they depend on. This is by no means a trivial problem, and much work has been done within this area. A well written introduction to the area is that of Guyon and Elisseeff [14].

In general, the best we can do is estimate the probabilities from sample data. Such methods are generally called *filters* because of their similarity to a sieve that filters out the relevant features from a set. Several methods to do this have been proposed based on concepts from statistics and information theory.

Another approach is to forgo the probabilistic analysis altogether, and instead look for empirically good features. This is the approach taken by the so-called *wrapper* methods which make use of embedded classifiers to evaluate the relevance of a set of features.

The two evaluation methods each have their strengths and weaknesses.

4.3.1 Wrappers

By nature of its use of results obtained from an actual classifier, a wrapper method produces empirical evaluations that are well adapted for the classifier used. This generally means that for a different classifier, the evaluation will be different.

A comprehensive treatment of various wrapper methods is that of Kohavi and John [21]. In this article they strongly argue for wrappers as superior to filters. They implement two different wrapper methods, and conclude that as long as the care is taken to prevent problems such as overfitting, wrappers consistently perform better than existing filter methods.

4.3.2 Filters

The benefit of filter methods is that they are not affected by the properties of any classifier. This means that the evaluation could theoretically work equally good, or bad, for all classifiers. Filters are also generally orders of magnitude faster than wrappers. The main downside is of course that a filter can only give an estimation of the relevance.

The success of the filter method is highly dependent on the sample set used. A representative set can give very reliable probabilities which will allow a filter method to accurately estimate relevance.

Filter methods typically make use of various statistical means to evaluate the relevance of features. We shall take a look at two such methods: *Correlation Analysis* and *Mutual Information Analysis*.

4.3.3 Correlation Analysis

One simple method by which to estimate relevance R of feature F_i when predicting class C is the *Pearson correlation coefficient*:

$$R(F_i; C) = \frac{cov(F_i, C)}{\sqrt{var(F_i) \cdot var(C)}} \tag{4.4}$$

where *cov* stands for the covariance and *var* the variance.

When applying linear regression we find that the square of $R(F_i; C)$, the so-called determination coefficient, is a measurement of the variance around the mean value of C explained by the linear relation between C and F_i. Conversely, $R(F_i; C)$ can be used as a rough relevance estimation. Naturally, only taking linear relations into account is a very restricted method.

4.3.4 Mutual Information Analysis

A more general evaluation of the relation between two variables is *mutual information*, also sometimes referred to as *information gain*. Mutual information is a statistical measure used in numerous contexts which loosely speaking attempts to measure the "amount of information" that is common between two data sets. The theory behind it is simple, and is based on the statistical concept of entropy.

The entropy H of a stochastic variable F_i is defined as

$$H(F_i) = -\sum_{f_i} p(F_i = f_i) \log_2 p(F_i = f_i) \tag{4.5}$$

where $p(F_i = f_i)$ is the probability that the variable F_i is in state f_i.

The entropy is commonly referred to as a measurement of disorder, or in other terms, a measurement of the information needed to describe this disorder.

Mutual Information uses the concept of entropy to describe a measurement of the information that is common between two stochastic variables. The Mutual Information I between the stochastic variables X and Y is defined as:

$$I(X, Y) = H(X) - H(X|Y) \tag{4.6}$$

where $H(X|Y)$ is the conditional entropy of X given Y, that is, the remaining entropy if X given a fixed value of Y. If we substitute the above definition of H we obtain

$$I(X,Y) = \sum_{x,y} P(X=x, Y=y) \log_2 \frac{P(X=x, Y=y)}{P(X=x) \cdot P(Y=y)} \tag{4.7}$$

Finally, a more recently proposed method is that of *Markov Blankets* [23], somewhat similar to mutual information analysis.

4.3.5 Feature Complementarity

An interesting question is that of feature complementarity. Since we are generally looking for a subset of features, is it sensible to judge features by themselves? Taking into account the two relevance classes, we would expect that strongly relevant features be possible to evaluate independently, while weakly relevant features would have to be evaluated together with other. Guyon and Elisseeff [14] analyze this and find that results agree with this idea, noting that:

- Noise reduction may be obtained by adding highly redundant variables.
- A feature that is useless by itself can be useful together with others.
- Two features that are useless by themselves can be useful together.

This suggests that features should ideally always be evaluated together. In the next section we take a look at various ways to do this.

4.4 Selection Strategies

With several methods to evaluate the relevance of features at hand, we now turn to the problem of selecting an optimal subset of features. In order to do this we need to define desirable properties of such a set.

The point of feature selection is to find a minimum number of features to achieve the highest possible accuracy. These two goals are commonly mutually exclusive, especially at the extremes (no features versus perfect accuracy). We may however try to find a sufficiently small set of features that provide satisfactory prediction.

In the words of Battiti [5], we may define the goal of feature selection as

> Given an initial set of n features, find the subset with $k < n$ features that is "maximally informative" about the class

where "maximally informative" should be interpreted as providing enough relevant information about the class, while discarding irrelevant information.

Returning to our two relevance classes 4.2 and 4.3, the goal would be to find a minimal set of all strongly relevant features, as well as one subset of weakly relevant which can still predict the class.

The problem of finding a minimum set of features is analogous to that of inference of minimal structures, which sadly is known to be NP-hard in the general case. This leaves exhaustive search as the only method certain to find the optimum solution.

4.4.1 Exhaustive Search

The simplest method is the exhaustive search, where we let an algorithm test all combinations of features in order to empirically determine the quality of recognition achieved using a given combination. Some quality value is recorded for a specific combination, and based on these values the relevance of every single feature and of every combination of features can be calculated.

As stated above, the exhaustive search is the only method known that is certain to find the true relevance of a feature. The downside, naturally, is the sheer amount of processing needed when working with large feature sets.

In order to test all combinations of k features out of a set of n we need $\binom{n}{k}$ evaluations, a number which is commonly too large for all but the smallest feature sets, especially for wrapper methods where each evaluation may take seconds or even minutes.

FOCUS-1 [3] is a well known exhaustive filter algorithm which works on binary domains. It exhaustively tests all subsets of features and selects the minimal subset that can still predict the class, with the property that the sets tested grow in size.

One common way to alleviate the time problem is to make use of lower order simplifications of the evaluation problem, where in this context, the order is the number of simultaneously evaluated features.

FOCUS-2 [3] is one example of how lower order simplifications may significantly speed up an algorithm, in this case FOCUS-1.

4.4.2 Greedy Selection

One common form of lower order simplification is the greedy algorithm. A greedy algorithm has no fixed limitation on the maximum order considered, but the simplification lies in that we never let it re-evaluate former decisions, which leads to a significant reduction of the search space.

In terms of feature selection, this limit implies that a feature that was selected at one point, will remain selected forever. This is invariably correct for strongly relevant features, but might be sub-optimal for weakly relevant features.

There are two ways to select features greedily:

- *Forward selection*, where we start with an empty set of features, and successively fill the set with the most relevant features until we obtain a minimal set which can successfully predict the class.
- *Backward elimination*, where we start with a complete set of features and successively remove the least relevant until we obtain a minimal set which can still predict the class.

Forward selection is generally a more efficient strategy to select a minimal set, however it has been shown, for instance by Guyon and Elisseeff [14], that less relevant subsets are typically selected since features are not necessarily evaluated together with the right complementary subset of features.

Kohavi and John [21] propose two simple wrapper methods to do greedy forward selection, while Battiti [5] applied this approach to device a filter method. We shall return to these algorithms in Sections 4.5 and 4.6.

4.4.3 Monte Carlo Simulation

Monte Carlo simulation is a name for a class of partly non-deterministic algorithms, i. e. algorithms that make use of a non-deterministic step to approximate a function. Typically, such an algorithm will randomly choose input data from a set, and use the results obtained with these data to progressively refine some hypothesis.

The RELIEF-F algorithm proposed by Kononenko [24] is an example of a Monte Carlo filter algorithm for feature selection, which assigns a relevance value to each feature. Samples are randomly selected from the training set and the relevance is updated based on a distance measure between the sample and the nearest instance of the same class as well as the nearest instance of a different class.

The explicit goal of RELIEF-F is to find every relevant feature, which in practice means that all weakly relevant features will eventually be selected, leading to significant redundancy. This could naturally be improved upon.

An example of a Monte Carlo wrapper algorithm is the Las Vegas Wrapper (LVW) [28], which is structurally similar to the greedy Hill Climbing Feature Selector algorithm discussed in Section 4.5.

4.4.4 Other

Many other optimization techniques could theoretically be applied to the problem of feature subset selection. Examples of such include simulated annealing and genetic

algorithms. However, as of early 2006 not much seems to been done using such techniques.

One final method that should not be overlooked is simple manual selection. For each activity we select the features we find sensible. This has the advantage that we can do this iteratively together with the feature extraction step. For an activity that is poorly recognized, we simply add a new feature which we think would aide the classifier, and select it.

The main downside of the method is that for a large set of features, it becomes quite tedious to select an appropriate feature set. Also, since features tend to overlap to some extent, it is not always trivial to select the best features. In fact, as subjective observers, it may not be at all obvious to a human which features are actually the best. Finally, with automatic feature selection, the entire idea of using an example based classifier comes into question.

4.5 The Hill-Climbing Feature Selector

A very simple feature selection algorithm is the Hill-Climbing Feature Selector (HCFS) [23], also commonly called steepest ascent.

HCFS is a naive wrapper algorithm, which greedily forward-selects features starting from an empty set of selected features S. At every iteration we test all features v_i from the set of not selected features $V = \{v_1, ...v_m\}$ one at a time together with the set of selected features using the test algorithm $TEST$, which typically returns the classification rate.

The feature v that is found to add the most relevance to the set S as measured by $TEST$ is then added to this set, and we go on to select the next feature. The algorithm stops when we decide we have enough features, or when adding more features does not improve the relevance.

As with all greedy algorithms we tend to miss higher order relations between features, which means that while strongly relevant features will typically be selected while weakly relevant may be left out.

The Hill-Climbing Feature Selector was considered an appropriate reference algorithm, and was therefore chosen to be implemented in this project. The algorithm is printed in its entirety in Figure 4.1.

4.6 Mutual Information Feature Selector

Battiti [5] reinterpreted the feature selection problem in terms of information theory:

> Given an initial set F with n features, find the subset $S \subset F$ with k

```
1: function S ← HCFS(F, C, k)
2:     S ← ∅
3:     R_S ← 0
4:     V ← F
5:     change ← true
6:     while |S| < k and |V| > 0 and change = true do
7:         change ← false
8:         for all v ∈ V do
9:             T ← S ∪ v
10:            R_T ← TEST(T, C)
11:            if R_T > R_S then
12:                S ← T
13:                R_S ← R_T
14:                change ← true
15:            end if
16:        end for
17:        V ← F \ S
18:    end while
19: end function
```

Figure 4.1. The Hill-Climbing Feature Selector Algorithm

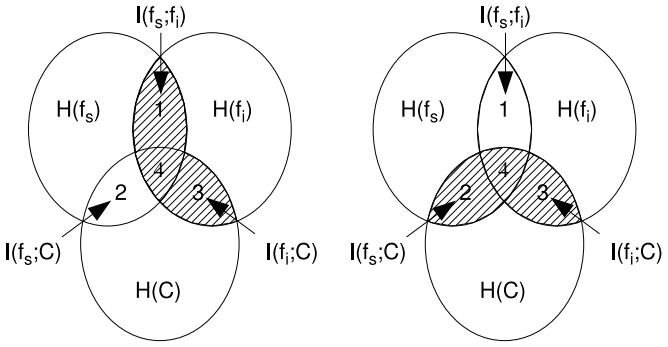
features that minimizes $H(C|S)$, i.e. that maximizes the mutual information $I(C;S)$.

Using this interpretation, he proposed a greedy algorithm called *Mutual Information Feature Selector (MIFS)* to select features based on their mutual information with the target class $I(C;F_i)$.

The function $I(C;F_i)$ can be estimated through histogram generation over the sample data, i. e. we divide the sample data of C and F_i into small bins and calculate entropy and mutual information from the sizes of these bins using equations 4.5 and 4.7.

Figure 4.2(b) shows in a information content Venn diagram an ideal greedy mutual information based algorithm. Since the desired result is maximum predictability, the ideal algorithm would find features that maximize the overlapping between the feature set and the target class, i.e. the area $\{2,3,4\}$, or in mutual information terms: $I(C;F_i,F_s)$. Sadly, this turns out to be incredibly difficult to calculate using the histogram method.

The original MIFS makes use of a simplification, by calculating mutual information between every feature and all the features that have already been selected $I(F_i;F_s)$, and subtracting a fraction times this value from the final evaluation $R(F_i)$ of a feature.



(a) MIFS (b) Ideal

Figure 4.2. MIFS (a) vs. Ideal Greedy Algorithm (b). The left region is the feature being evaluated, the right is the set of features already selected (if any), and the lower region is the target class.

$$R(F_i) = I(C; F_i) - \beta \cdot I(F_i; F_s) \tag{4.8}$$

Figure 4.2(a) shows the result of this equation, namely that instead of maximizing area $\{2, 3, 4\}$, we maximize $\{1, 3, 4\}$. This makes MIFS very sensitive to the amount of overlapping chosen, i. e. the value of β.

Kwak and Choi [25] suggested a simple way to approximate the desired area $\{2, 3, 4\}$. The shaded area for the ideal algorithm can be specified as $I(C; f_i, f_s)$, which can be rewritten as

$$I(C; f_i, f_s) = I(C; f_s) + I(C; f_i|f_s) \tag{4.9}$$

where $I(C; f_s)$ can be identified as areas 2 and 4, and $I(C; f_i|f_s)$ as area 3. Since $I(C; f_s)$ is constant for every feature evaluated, its value does not affect the relative effectiveness of a feature and we do not need to calculate this, which leaves $I(C; f_i|f_s)$ as the area to be maximized. This however turns out to be extremely hard in the general case. Kwak and Choi therefore suggest we expand this expression as

$$I(C; f_i|f_s) = I(C; f_i) - I(f_s; f_i) - I(f_s; f_i|C). \tag{4.10}$$

Assuming the information is distributed evenly throughout $H(f_s)$ in Figure 4.2(b), we may write

$$\frac{H(f_s|C)}{H(f_s)} = \frac{I(f_s; f_i|C)}{I(f_s; f_i)} \tag{4.11}$$

which combined with equations 4.10 gives

$$\begin{aligned} I(f_i; C|f_s) &= I(f_i; C) - \left(1 - \frac{H(f_s|C)}{H(f_s)}\right) I(f_s; f_i) \\ &= I(f_i; C) - \frac{I(f_s; C)}{H(f_s)} I(f_s; f_i) \end{aligned} \tag{4.12}$$

and finally we give an improved estimation $R_2(f_i)$ of the area $\{2, 3, 4\}$ as

$$R_2(f_i) = I(C; f_i) - \beta \left(\frac{I(f_s; C)}{H(f_s)} I(f_s; f_i)\right). \tag{4.13}$$

Once again, this only holds under the assumption that $H(F_s)$ is approximately evenly distributed. Kwan and Choi observe that this estimation is good to within 10% for typical data sets.

The improved Mutual Information Feature Selector algorithm appeared to be a highly suitable algorithm for our purposes, and was therefore chosen to be implemented. The algorithm is printed in its entirety in Figure 4.3.

4.7 Chapter Summary

In this chapter we defined the relevance of a feature and looked at methods to evaluate this. We then discussed the problem of selecting a minimum subset of features that can still predict the class. Two algorithms were discussed in detail, the Hill-Climbing Feature Selector, and the Mutual Information Feature Selector. Following this, we shall go on and implement them in order to evaluate their performance at selecting features.

```
1: function S ← MIFS(F, C, β, k)
2:     S ← ∅
3:     i_best ← 0
4:     f_best ← ∅
5:     for all f ∈ F do
6:         i ← I(C; f)
7:         if i > i_best then
8:             i_best ← i
9:             f_best ← f
10:        end if
11:    end for
12:    F ← F \ {f_best}
13:    S ← f_best
14:    while |S| < k do
15:        for all f ∈ F, s ∈ S do
16:            i(f, s) ← I(f; s)
17:        end for
18:        i_best ← 0
19:        f_best ← ∅
20:        for all f ∈ F do
21:            i ← I(C; f) − β Σ_{s∈S} i(C; s) · i(f, s) ÷ H(s)
22:            if i > i_best then
23:                i_best ← i
24:                f_best ← f
25:            end if
26:        end for
27:        F ← F \ {f_best}
28:        S ← S ∪ {f_best}
29:    end while
30: end function
```

Figure 4.3. The Mutual Information Feature Selector Algorithm

Chapter 5

Implementation

In the preceding chapters we discussed various methods to extract, evaluate and select features. In order to solve the problem presented in Chapter 1, a software implementation of the methods are needed. This chapter discusses implementation details and software design considerations.

5.1 Overview

A typical design of an activity recognition system was outlined in Chapter 2. This design uses three major sub-systems: *motion capture*, *motion analysis* and *recognition*.

The thesis itself concerns mainly the motion analysis part, and the previous two chapters covered this part in great detail. However in order to properly methods to extract, evaluate and select features a complete activity recognition system had to be designed and implemented.

The system was built using MATLAB R13. The decision to use MATLAB was based on its rich toolboxes, its ability to manipulate vectors and matrices, and its familiarity with the author.

Originally the system was run offline for speed reasons, but at the end the MATLAB compiler was used to create a C++ interface which was then integrated the tracking system.

5.2 Motion Capture

According to the problem definition the motion capture part should not be a central part of the thesis, and hence an readily available system was used. Also, the

standard body model used in this system was used.

5.2.1 Body Model

The body model describes the 10 main human limbs: one each for both upper and lower arms and legs, one for the torso and one for the head. The hands and the feet are not tracked, but their positions can be estimated from the model.

The reason for aiming at this level of detail is simple. It is detailed enough to make activity recognition trivial for a human observer, yet primitive enough to facility efficient digital storage on a frame-by-frame basis.

The limbs are stored as elliptical cylinders, together with transformations that describe their angular and positional relations relative to one another. This in practice means we have a fleshed out stick figure type model. Figure 5.1 shows how a visualization of the model looks.

The relation between each limb in the model is described with a homogeneous transformation matrix, which combines a rotation and a translation into a convenient unit. This allows us to construct the model as in a tree fashion, where all the limbs are described as relative only to their parent limb, while still being able to quickly recover the global position and angle of each limb.

5.2.2 Tracking

A complete tracking and body modeling system called Voodoo was used. Voodoo [20] is developed by Steffen Knoop and Stefan Vacek here at UKA, and makes use of Iterative Closest Point (ICP) adaptation of any predefined body model to a point cloud given by an external 3-D tracking system. The adapted body model is stored in an XML file, which is simple to read into MATLAB.

The hardware tracking system used was the CSEM SwissRanger, an infrared time-of-flight camera capable at operating at over 25 frames per second. The optical resolution of is however quite low, which limits the range at which reliable ICP adaptation can be done.

The Voodoo system also allows merging with optical hand and face tracking data to improve the model fitting, however this was not found to be necessary for this project.

5.3 Motion Analysis

Since the motion analysis part is the core of the thesis, this part had to be developed from scratch using the theory developed in Chapters 3 and 4.

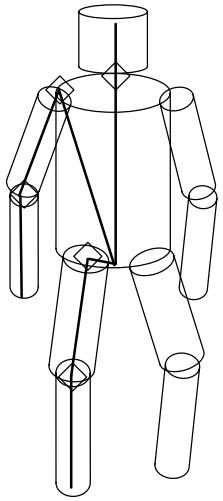


Figure 5.1. Cylindrical Body Model

5.3.1 Feature Extraction

The theory behind feature extraction were covered in Chapter 3. These methods were implemented in MATLAB as a flexible framework to extract features from selected model data.

For this part the vast libraries available in MATLAB were of great use, as functions like Fast Fourier Transform, Principal Component Analysis and Covariance did not have to be written anew, which significanly decreased the amount of code that had to be written.

Additionally, the native MATLAB implementations are extremely fast and generally vectorized. This is important as features have to be extracted in real-time when the system is running in an on-line fashion.

5.4 Feature Post-processing

An attempt to improve the quality of the signals given by the feature extraction system was done by means of filtering. Most signals carry a certain amount of noise with them, and a good filter can smoothen out the signal. A so-called Finite Impulse Response (FIR) filter was attempted, which produced aesthetically more pleasing plots of the features. However it later turned out that unfiltered data produced much better training and recognition rates in the inference engine, and all filtering was therefore switched off.

Normalization is another useful technique. Since we will not model time dependen-

cies in the recognition step, the actual shape of each signal over time is not very important. Instead we are looking for notable differences in signal value. Therefore the maximum value for each feature measured over all activities was stored, and used as a normalization value. This in effect places all features within the range [-1, 1].

5.4.1 Feature Evaluation and Selection

For the feature selection the two algorithms MIFS and HCFS printed in detail in Chapter 4 were implemented. For this part not much was available in the libraries, and these functions had to be written almost from scratch.

This meant that performance was not extremely good, however since the evaluation and selection parts are only run during training, this was thought not to be a major problem. Later it turned out that the enormous run time needed for HCFS limited the results we could obtain therefrom.

5.5 Recognition

As with the tracking part, the recognition step itself is not a central part of the thesis, so a readily available inference engine was decided upon to be used as a classifier. A number of such exist in MATLAB, so a suitable choice had to be made.

Since the number of inputs to the inference engine is likely to vary because of the feature selection step, each activity has to have its own classifier. This in turn enables us to use a single output signal which we can then treat as the likelihood of the activity having been observed.

5.5.1 Inference Engine Choice

The choice of inference engine fell on the Neural Network (NN), more specifically a Feed Forward Neural Network (FFNN). The reasons behind this choice were three-fold. Firstly, the ready availability of NNs in MATLAB. Secondly, the relatively good tolerance towards superfluous dimensions in the training data is essential in order to be able to extract new features. Thirdly, the author's familiarity with NNs played a big part in the choice.

Other engines that were evaluated include Radial Basis Functions (RBFs), Support Vector Machines (SVMs) and Hidden Markov Models. The former two are structurally similar to the Neural Network. RBFs were attempted, but were never found to produce results comparable with NNs. SVMs were suggested as a superior alternative to NNs, but since working SVMs are still rare their use was considered unfeasible.

A Neural Network, as the name implies, is an interconnected group of neurons. A neuron in this context is a mathematical function which takes one or more input signals, sums them according to a *transfer function*, and gives an output signal within a specified range.

In a Feed Forward Neural Network, the connections between the neurons are in a single direction, i.e. no cycles may exist in the network. In this kind of network, it is common to divide the neurons into *layers*.

5.5.2 Layer Design

The main design choice for a Neural Network is the number of layers to use, and the size of each layer. In general, only very simple concepts can be learned with one or two layers, while more than three typically only increases training time. Hence three layers were decided upon, an input layer, a middle (or hidden) layer, and an output layer.

For the input and output layers, the layer sizes are obvious. The input layer connects to the signals given from the feature extraction step, and must therefore have the same dimensionality as the feature subset. The desired output for our purposes is naturally a single value, since what we want is a probability that the input data matches a given class, and hence the output layer should be a single neuron.

The middle layer was experimentally decided to have 10 neurons, as 10 neurons appeared to be sufficient to achieve good training. Less than 10 neurons worsened the recognition rate, while more than 10 appeared not to affect the results.

5.5.3 Transfer Functions

Another important design decision is the transfer function between the layers. Here the choice was quite simple. Between the input layer and the middle layer the traditional *tansig* or "squashing function" was used. This function is a continuous mapping from an infinite input domain to the range [-1, 1].

$$tansig(n) = \frac{2}{1 + e^{-2n}} - 1 \tag{5.1}$$

The second transfer function between the middle layer and the output layer was chosen to be *logsig*, which is similar to *tansig* however has the range [0, 1], which is suitable for our interpretation of the output signal as a probability.

$$logsig(n) = \frac{1}{1 + e^{-n}} \tag{5.2}$$

Both these functions are available in the MATLAB neural network toolbox.

5.6 Chapter Summary

This chapter described the implementation of the theory covered in previous chapters. With a working implementation we are able to measure results and draw conclusions from the work done.

Chapter 6

Experiments and Results

After covering the implementation of the methods discussed in this thesis, we shall now look at the experiments done using the system and the results obtained.

6.1 Experiment Setup

In order to test the methods developed in this thesis suitable experiments had to be devised and executed. The goal was twofold:

- To test the efficiency of the feature extraction methods used when predicting the class, in other words the recognition result.
- To test the efficiency of the two selection algorithms and their influence on the recognition result.

The strategy used was to do two main experiments, which were called the *build-up* experiment and the *tear-down* experiment.

For the build-up experiment we start out with the tracking data as a primitive feature set and then build a large super-set of features using the methods developed in Chapter 3 in order to achieve excellent recognition results.

After this comes the tear-down experiment, where a small subset of highly relevant features are selected per activity, while trying to retain good recognition results.

6.1.1 Activity Selection and Recording

In order to have some data to work with a number of test activities had to be recorded. The activities were chosen with a few considerations in mind. A good activity was decided to be:

Table 6.1. Activities Used for Experiments

#	Name	Description
1	Balance	Balance on left leg
2	Bow	Upper body bow
3	Call	Right hand waving gesture towards body
4	Clap	Hand clap in front of body
5	Flap	Flap with both arms, "attempt to fly"
6	Handshake	Handshake with right hand
7	Kick	Kick with right leg
8	Manipulate	Object manipulation with both hands
9	Sit	Sit down on chair
10	Walk	Walk towards camera
11	Wave	Right hand waving

- a natural activity that people commonly do.
- successfully and reproducibly tracked by the motion capture system.
- similar to at least one other activity, to show granularity of the system.
- dissimilar enough to other activities show range of system.

After some experimentation with tracking results a set of 11 activities that had all of these properties were selected and recorded. These activities were assigned a number which we call *class*. A list of the activities are found in Table 6.1.

To ensure a sufficient data set for both training and testing 10 example sequences were recorded for one male and one female subject for each activity, making 20 sequences per activity, and in total 220 sequences and 21 222 frames.

6.1.2 Sequence Segmentation

The recorded sequences were manually segmented and assigned its correct activity, or *class* as an integer.

In the segmentation phase, frames were divided into three classes: initialization, activity and finalization. Initialization frames were considered all frames up until the activity could be readily and correctly identified by a human. Activity frames were all frames where the activity was clearly being performed. Finalization frames were considered frames where the activity was being halted.

For the training and evaluation, only the actual activity frames were used. The initialization and finalization frames generally tend to overlap the activity frames slightly. However, it was thought that discarding some quantity of training data to achieve higher quality was a sensible thing to do.

6.1.3 Data Set Preparation

The 240 segmented and classified sequences were then prepared into data sets suitable for training and classification. As discussed in Section 5.5 the neural network used as classifier needs tree sets of data, a training set, a validation set and a test set.

Out of the set of 20 sequences available per activity we must therefore create three subsets of appropriate sizes. Here we want the training set to be as large as possible, while having a validation set large enough to prevent overfitting, and still have a sufficiently large test set to draw conclusions. In the literature sets of equal sizes are suggested as a first attempt, implying approximately 7 sequences per set. Based on this it was empirically determined that a validation set of 4 sequences and a test set of 6 sequences was appropriate.

For the training set we may then use at most 10 sequences, but in order to be able to test the recognition rate for varying train set sizes, five different training sets ranging from 2 to 10 sequences at intervals of 2 sequences were prepared.

The sequences were selected randomly from the total set with an equal number of sequences from the male and the female subject. To diminish the problems resulting from a random selection, five complete data sets with training, validation and test sequences were randomly created for each of the 5 training set sizes, making in total 25 sets.

6.1.4 Feature Extraction

With these results as grounding the next step was to attempt to improve them by expanding the feature set in order to facilitate better recognition. The methods developed in Chapter 3 were applied in various ways to the tracking data. This was done in an iterative fashion, where a few new features were extracted and the recognition results were then observed. Further new features were added and once again the results were recorded.

As there is essentially no limit on the number of features we may extract from the model, we need a point at which to stop this process. For this project this point was decided to be when the recognition rate surpassed 98%.

In total this resulted in 118 features, including the 45 in the primitive set. The complete list of features developed are found in Table 6.2.

6.2 Build-up Experiment

For both of our experimental goals the significant result we want to calculate is the recognition rate. In essence, this is a measure of the quality of the extracted and

Table 6.2. Complete List of Features

#	Description	#	Description
1-3	Torso angle	82-84	Left hand velocity
4-6	Head angle	85-87	Right foot velocity
7-9	Left upper arm angle	88-90	Left foot velocity
10-12	Right upper arm angle	91	Body position planarity
13-15	Left upper leg angle	92	Right hand position planarity
16-18	Right upper leg angle	93	Left hand position planarity
19-21	Left lower arm angle	94	Right foot position planarity
22-24	Right lower arm angle	95	Left foot position planarity
25-27	Left lower leg angle	96	Right hand position variance
28-30	Right lower leg angle	97	Left hand position variance
31-33	Head position	98	Hand covariance
34-36	Right hand position	99	Foot covariance
37-39	Left hand position	100	Hip covariance
40-42	Right foot position	101	Knee covariance
43-45	Left foot position	102	Hand distance period
46-48	Torso angular velocity	103	Foot distance period
49-51	Head angular velocity	104	Right knee angle period
52-54	Left upper arm angular velocity	105	Left knee angle period
55-57	Right upper arm angular velocity	106	Right hip angle period
58-60	Left upper leg angular velocity	107	Left hip angle period
61-63	Right upper leg angular velocity	108	Right elbow angle period
64-66	Left lower arm angular velocity	109	Left elbow angle period
67-69	Right lower arm angular velocity	110	Right shoulder angle period
70-72	Left lower leg angular velocity	111	Left shoulder angle period
73-75	Right lower leg angular velocity	112	Distance between hands
76-78	Body velocity	113-115	Mean right hand position
79-81	Right hand velocity	116-118	Mean left hand position

selected features, and in effect a measure of the quality of the input data. There are two ways for us to measure how well an activity was recognized: The number of frames correctly classified, and the number of sequences correctly classified.

While the number of frames is simple to count, it is harder to define exactly when a sequence has been recognized. As a vague definition we may say that the majority of the frames must be correctly classified in order to say that a sequence was recognized. As we shall see, with the high recognition rates achieved we can be certain that at least a significant percentage of each sequence was recognized.

Each frame was rated into one of four categories according to the firings of the neural networks:

- *Correct* when the neural network corresponding to the correct activity fired greater than 0.7.
- *Ambiguous* when at least one more neural network apart from the correct fired

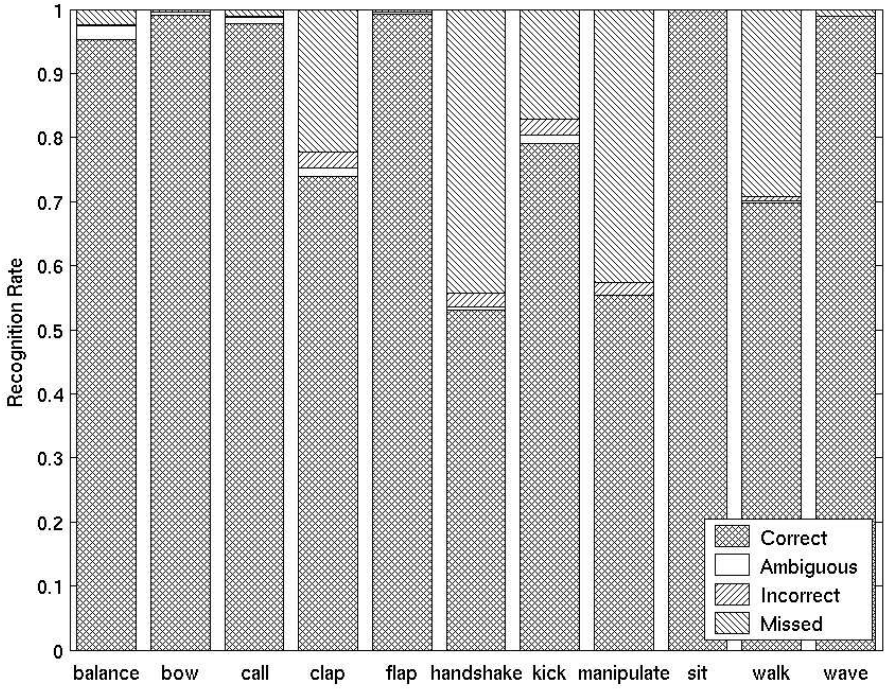


Figure 6.1. Recognition Results Per Activity for the Primitive Feature Set

greater than 0.7.

- *Incorrect* when one or more neural networks corresponding to incorrect activities fired greater than 0.7, but the correct neural network did not.
- *Missed* when none of the neural networks fired greater than 0.7.

Average recognition rate values are found in Table 6.3, while the complete set of results per method and activity is found in Appendix A.

6.2.1 Primitive Feature Set

Figure 6.1 shows a bar diagram over the recognition results obtained per activity using only tracking data. Overall, the results turned out to be surprisingly good, with several activities showing recognition rates over 90 %. Especially noticeable are the activities *balance*, *bow*, *call*, *flap*, *sit* and *wave*, for which we obtain close to 100% recognition.

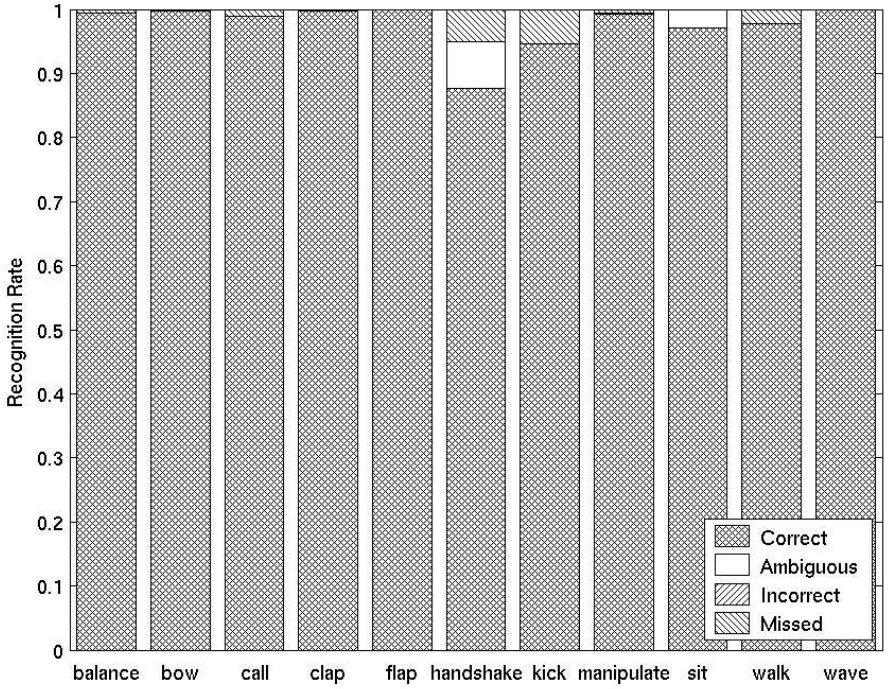


Figure 6.2. Recognition Results Per Activity for the Complete Feature Set

Some activities do not fare quite as well, especially *handshake* and *manipulate* where the recognition rate is around 50 %. Also *clap* and *kick* show less than 80 % recognition, and *walk* less than 70 %.

In order to explain why some activities were exceptionally well recognized we can take a look at which properties we can easily notice from tracking data itself. For the six activities what have over 90% correct classification we notice that there is one or more model angles that should readily tell the activity: *balance* (torso), *bow* (hip, torso), *call* (elbow), *flap* (shoulder), *sit* (hip) and *wave* (elbow).

6.2.2 Complete Feature Set

Figure 6.2 shows a bar diagram over the results obtained using the complete set of features extracted as described in Section 6.1.4.

Here we obtain nearly perfect recognition, with all activities having more than 90% recognized frames. For the activity *handshake* we have some problems with ambiguity for approximately 5 % of the frames. Only *handshake* and *sit* now have more than 5% not recognized frames.

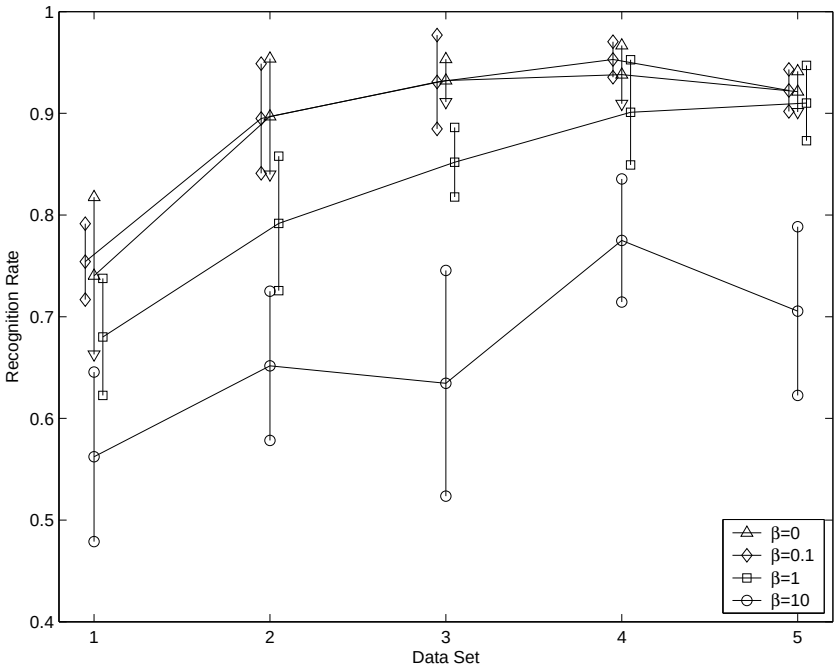


Figure 6.3. Recognition Rates for MIFS Subsets With Varying β Value

6.3 Tear-down Experiment

Convinced by the results of the build-up experiment that the features extracted were sufficient to successfully recognize the activities, we shall make use of the feature subset selection techniques described in Chapter 4.

The two selection algorithms, MIFS and HCFS, were run on the data sets. For both selectors a maximum of 5 features were chosen. This limit was placed mainly to try to confine the time needed for the HCFS algorithm. In the case of MIFS we could in practice let it choose all features until the mutual information is exhausted, but to make its results comparable to HCFS the same limit was applied here.

Despite this limit, the run time needed by HCFS to select subsets proved to be enormous. On a 3.0 GHz Pentium 4 the selection of a single feature for one single activity took on average approximately 4,900 seconds. In order to finish in reasonable time the work was hence distributed on several computers.

As above, average recognition rate values are found in Table 6.3.

6.3.1 MIFS Parameter Selection

The MIFS algorithm takes one parameter, β, which determines the discounting of features based on their mutual information with already selected features. In order to find the optimal β for our purposes, different values were tested and the resulting recognition rate recorded.

Figure 6.3 is a plot of recognition result for the various data sets, with standard deviation shown as error bars. From the plot we immediately see that lower values of β show strong results for the smaller data sets.

The primary effect of β is that for larger values the feature set selected will generally be smaller, since we stop the algorithm once new features do not add information according to our estimation. This is the reason why for the largest value of β we get extremely poor results, as the subsets selected contained at most two features.

As we increase the data sets the difference becomes smaller, and for our largest data set the results approach a common value, agreeing with the conclusions of Kwak and Choi [25]. Noticeable is that the largest data set show worse results than the second largest. This phenomenon is discussed in detail further on.

Based on these results the value chosen was $\beta = 0.1$ for all further MIFS runs.

6.3.2 MIFS Feature Subset

Figure 6.4 shows the recognition result per activity using the feature subset MIFS–5. The average recognition rate of 92.3 % places the MIFS–5 set in between the primitive and the complete feature sets.

The selected feature subset enables excellent recognition of many activities, above 95 % for eight of the 11 activities. However two activities, *call* and *handshake* show very poor results with only around 70 % recognition.

In order to see of a slightly larger feature set would improve on these results, another MIFS run was done, this time producing the set MIFS–10, with 10 features. Figure 6.5 shows the recognition result per activity using the feature subset MIFS–10.

The average recognition rate of 95.6 % shows a slight 3.3 % improvement compared with MIFS–5. We may notice from the figure that *call* has reached above 90 %, but *handshake* is still only around 80 %.

6.3.3 HCFS Feature Subset

Figure 6.6 shows the recognition results per activity using the feature sets selected by HCFS. The average recognition rate of 95.2 % places the HCFS–5 set in between

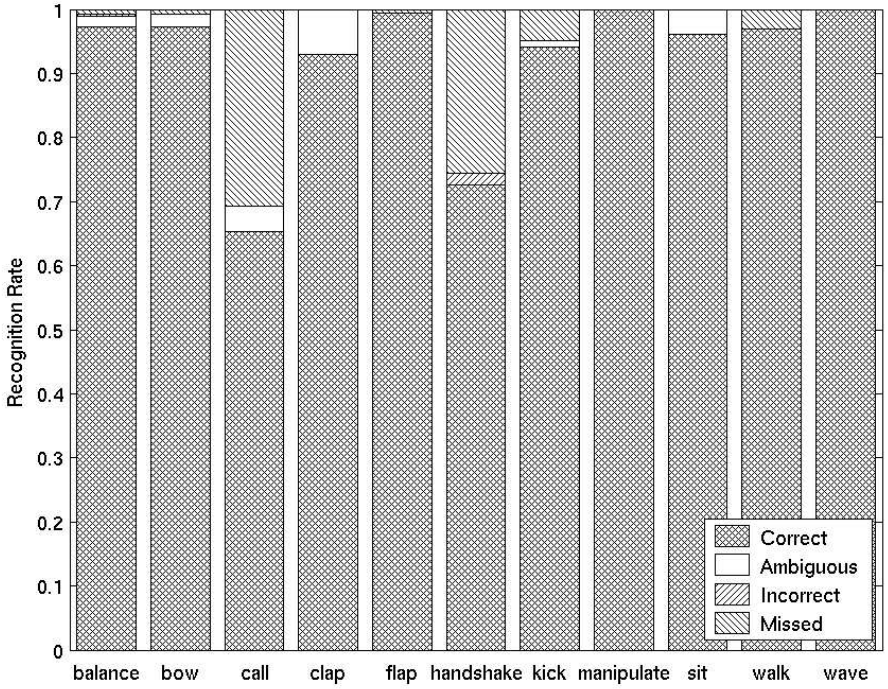


Figure 6.4. Recognition Results Per Activity for the MIFS–5 Feature Set

the MIFS–5 and the complete feature sets.

6.4 Feature Subset Analysis

After covering the results obtained from various feature sets, we may try to analyze wherein differences lie, in other words what the subsets look like.

6.4.1 Recognition Rates for Various Training Set Sizes

One measurement is how well the system generalizes, that is, how well the classifier is able to learn a concept instead of just the very examples used in training. As was stated in Chapter 4, one of the desired properties of feature selection is quicker generalization. In other words, we should be able to train the system with fewer examples.

In order to test this a series of tests was carried out with increasing number of training sequences. For each activity training was done with 2, 4, 6, 8 and 10 sequences,

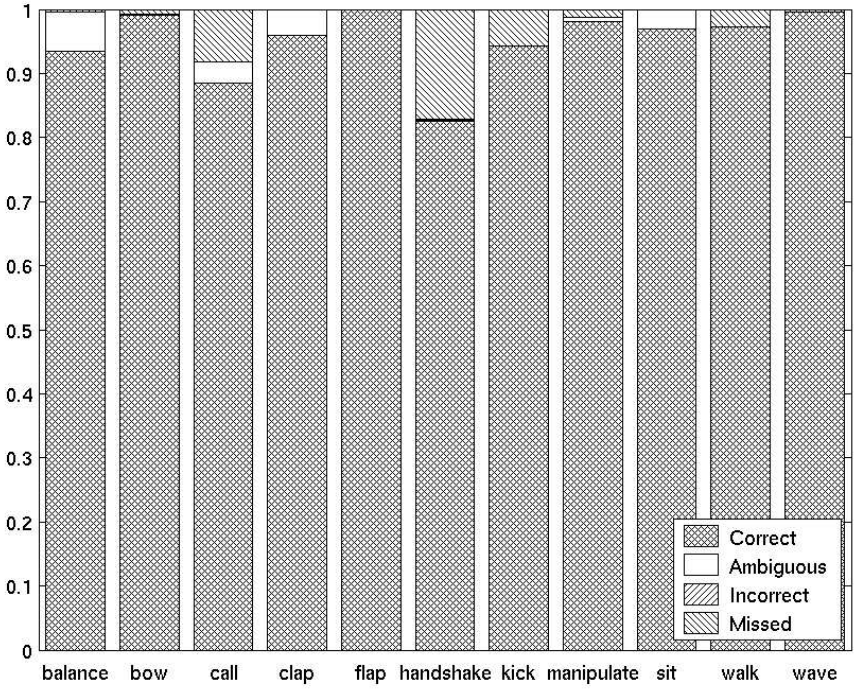


Figure 6.5. Recognition Results Per Activity for the MIFS–10 Feature Set

after selecting features with MIFS and HCFS. As before, 4 validation sequences and 6 test sequences were used. All sequences were selected randomly from the 20 recorded and like before, in order to alleviate the problems with randomization, five sets of each size were created.

Figure 6.7 is a plot of average recognition rate versus number of training sequences for the primitive features, the entire set of features, and for subsets selected by MIFS and HCFS.

As can be seen the complete set of features is constantly superior to the other sets. Already with one training sequence the recognition rate is above 90%. With this small set we have a noticeable region with ambiguous results and even a few percent incorrect classification. As we add more training sequences we essentially eliminate everything but the correct and missed sets.

The primitive set is able to match the subsets selected by our two algorithms with the smallest training set for the recognition rate. Using more training data however improves recognition very marginally. Additionally, both the miss rate and the error rate are considerably higher than for the three other sets.

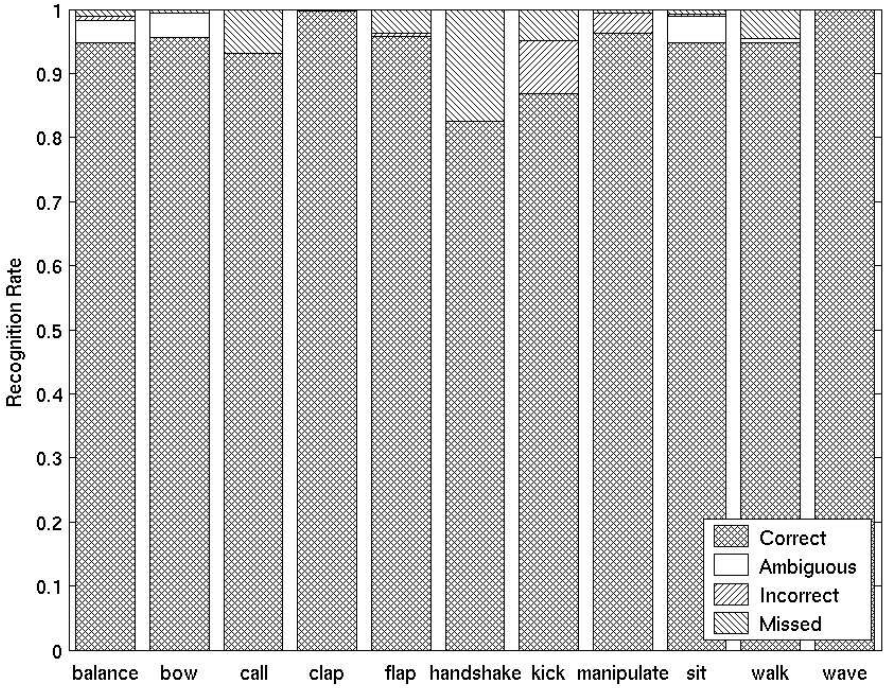


Figure 6.6. Recognition Results Per Activity for the HCFS–5 Feature Set

Table 6.3. Average Recognition Rate Over Five Runs

Method	Correct	Ambiguous	Incorrect	Missed
Primitive	84.0 %	0.8 %	0.8 %	14.4 %
All features	98.3 %	0.6 %	0.0 %	1.1 %
MIFS 5 features	92.3 %	2.1 %	0.1 %	5.5 %
MIFS 10 features	95.7 %	1.6 %	0.0 %	2.7 %
HCFS 5 features	95.2 %	0.9 %	0.8 %	3.1 %

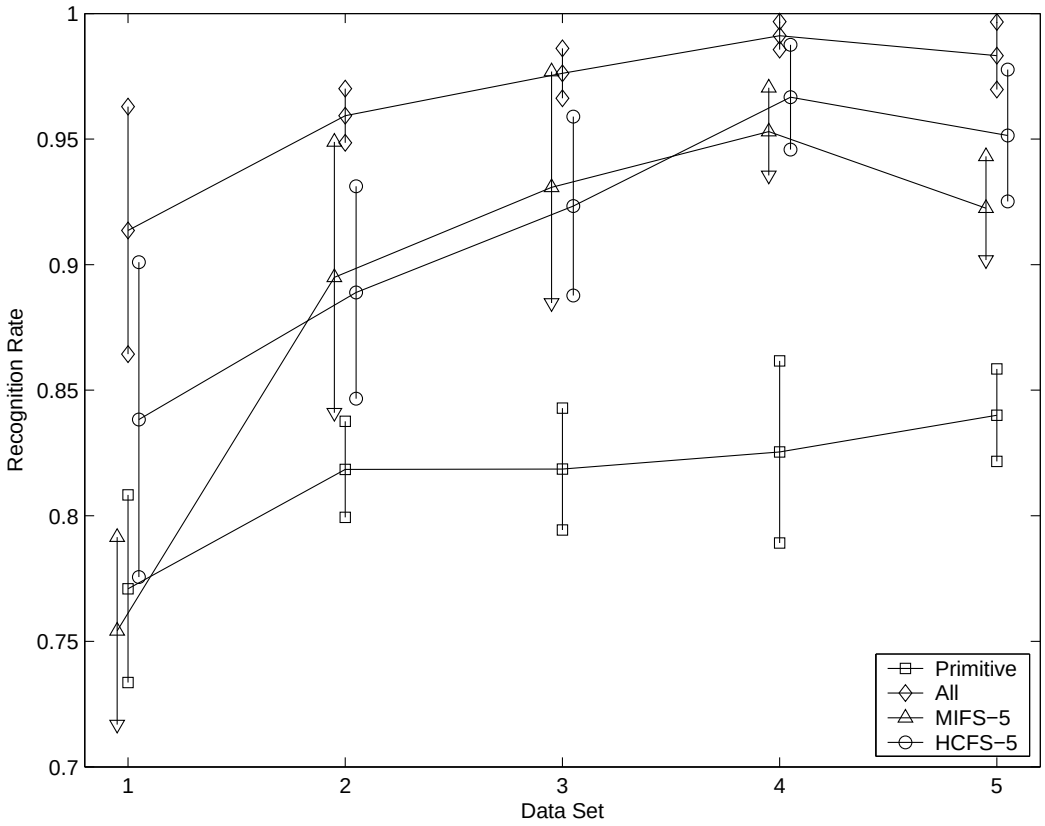


Figure 6.7. Subset Comparison

The two selection algorithms perform similarly at the beginning. While HCFS has a noticeable lead with the smallest set, MIFS improves significantly when more data is available for its entropy calculations, and taking into account the relatively large error bars the two algorithms perform comparably with data sets of 2 sequences and more.

6.4.2 The Largest Set Anomaly

An unexpected phenomenon observed in all plots of recognition results versus training set size, such that Figure 6.7 is that the largest set of 5 sequences appears to produce worse results than the second largest set of 4 sequences. The difference is very marginal, but appears for all methods except for the primitive set.

The most sensible explanation for this phenomenon lies in the creation of the data sets, as described in Section 6.1.3. Since the sets are constructed by randomly selecting sequences, it is probable that the training and test sets were accidentally assigned very dissimilar sequences for the largest set. The fact that the standard

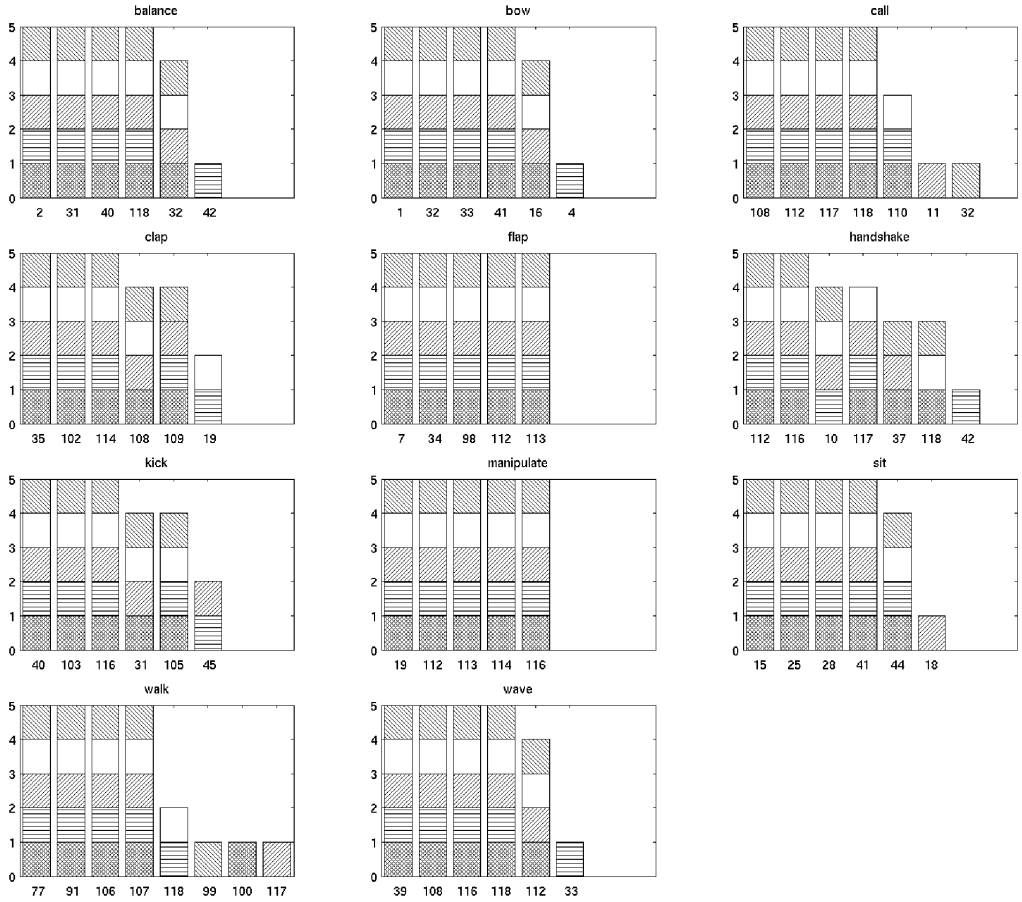


Figure 6.8. MIFS Feature Subset Breakdown

deviation is also significantly higher lends credibility to this interpretation.

6.4.3 Strong and Weak Feature Separation

Keeping in mind the distinction we made between strong and weak relevance it would be interesting to see if this distinction holds up in our test results. The problem here is naturally to separate the two classes. One way to define a strong feature in terms of the results obtained is as a feature that is invariably selected by the algorithms.

Figure 6.8 is a pattern coded bar diagram breakdown of the features selected by the MIFS algorithm. For each activity the features are listed on the horizontal axis, with the numbers corresponding to the feature list found in Table 6.2. The vertical axis shows the number of times it was selected out of the five test sets. Additionally,

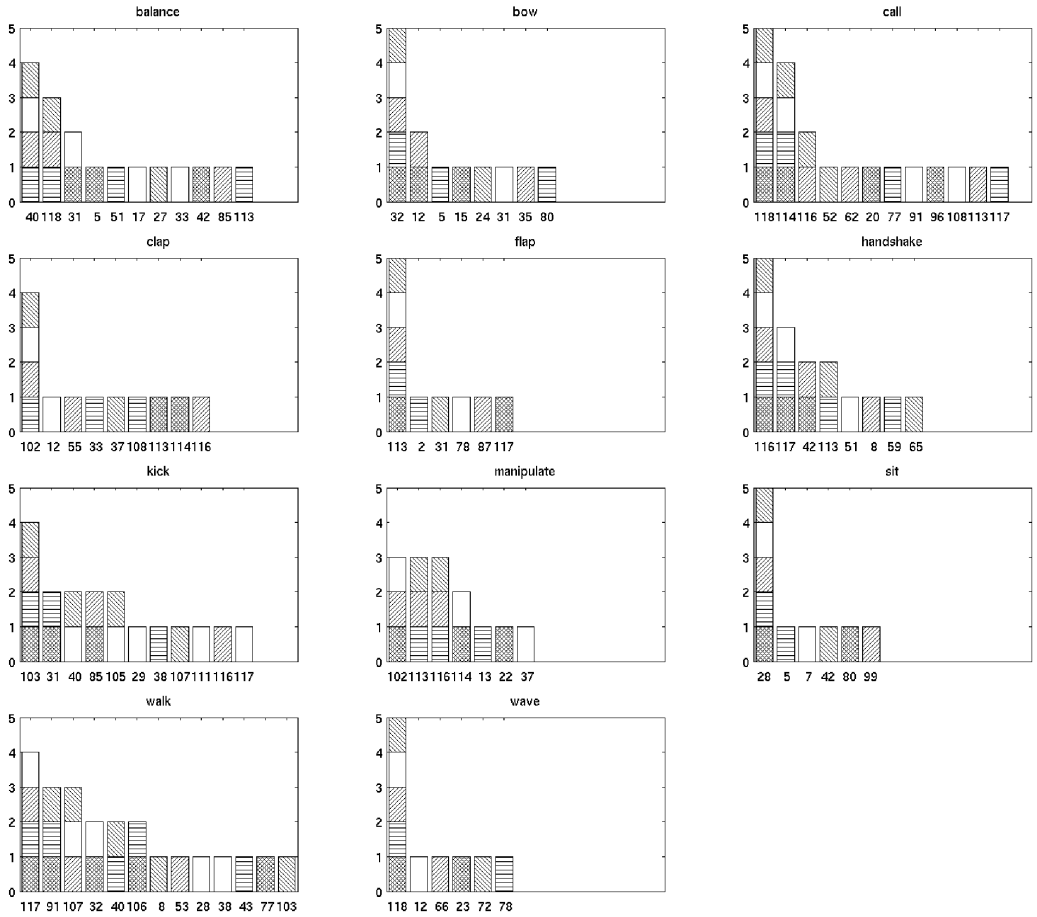


Figure 6.9. HCFS Feature Subset Breakdown

the patterns show the different set compositions, so that all blocks having the same pattern were selected together.

The most striking feature of this plot is how often MIFS finds very similar sets of features, in other words the strong features. This is very promising results, since it shows that the feature selection really manages to capture the features of the activity, and not just those of the very sequence used. For two activities, *flap* and *manipulate*, the exact same set is selected for all five runs, and for another three activities, *balance*, *bow* and *wave*, only one set differs. The most difference is found for *handshake* where no two sets are identical, however we still invariably find combinations of only seven unique features.

Similarly, Figure 6.9 is a breakdown of the features selected by the HCFS algorithm. Here the variance is far greater than for MIFS. Particularly, the sets are generally smaller, as the selection was aborted as soon as sufficiently good results

were achieved, whereas MIFS never got close enough for this to happen with only five features.

What is notable is that HCFS seems to create sets consisting of *one* strongly relevant feature, combined with various combinations of features that may be either strongly or weakly relevant. We may try to decide such features by comparing the two subsets.

For all activities except *manipulate*, the single clearly strongly relevant feature according to HCFS is also found among the strongly relevant features as found by MIFS suggesting that the feature is indeed strongly relevant. If the features found by MIFS were truly strongly relevant, we would expect them to show up for HCFS as well. This is indeed the case for most activities, where many of the features that were chosen by MIFS for all sets turn up in the subsets created by HCFS. The overlapping ranges from only one out of five features for *flap*, up to five out of six features for *kick*.

6.5 Chapter Summary

In this chapter the experiments done for the thesis were described, along with the results obtained. In general, the results agree well with the hypotheses presented in previous Chapters, and the outcome of the experiments were considered to be a success. We shall now go on and draw conclusions from the results in the final chapter of the report.

Chapter 7

Summary and Conclusions

In this chapter we look back at the problem defined in the introduction and summarize the work done in this project towards solving it. Conclusions are drawn from the work, particularly the results presented in the last chapter. Finally, possible future work is suggested.

7.1 Problem Definition Revisited

The problem behind this project was to construct an activity recognition system and evaluate its effectiveness. Given an existing tracking and body modeling system, the problem was to create an adaptive machine learning system that performs reliable recognition of various human activities based solely on the motion of the body model. The recognition part was expected to use an inference engine to allow training of new activities.

The activities to be classified were specified as mainly larger scale motions, and only activities that can be recognized purely by observing the motion of the human body, examples being walking, sitting and waving.

7.2 Project Summary

From the initial literature review, several working approaches to the problem were identified. The Voodoo tracking system developed here at the Universität Karlsruhe seemed a natural starting point, which in turn decided the course of the project: to develop a machine learning system that could be trained from the data obtained by this tracking system.

The author's familiarity with the Neural Network, as well as its availability in MATLAB became strong arguments in favor of that particular machine learning

system. The initial naive attempts to apply the Neural Network directly to the tracking data showed promising results, but it was apparent that pre-processing would greatly improve the results.

Referring back to the literature the pre-processing was done through the extraction of abstract *features* from the tracking data. This turned out to be relatively simple, and results improved considerably using statistical and analytical methods.

After extracting a very large number of features, the recognition results were approaching 100%, and focus then moved more towards finding out which features were actually responsible for the results. Mutual Information turned out to be a very effective method to evaluate features, and based on this the Mutual Information Feature Selector (MIFS) was applied in order to find good subsets of features.

The final part of the project consisted of experimentation. A large training and testing set of various activities was recorded, and used to train the system using various subsets of features. In order to relate the effectiveness of MIFS a brute force method was implemented as well. Significant analysis of the data obtained was done, and conclusions drawn

7.3 Conclusions

This thesis has shown how a combination feature extraction, evaluation and selection can work together to provide a high quality data set for use with an inference engine.

Feature extraction is not necessarily a difficult problem. In fact, as the feature evaluation step has shown, many of the best features turn out to be model data or very simple functions of it. For some activities, more complex features are needed.

Feature selection can be done using a variety of methods. A few such were attempted in this project, and the idea behind selection turns out to be sound.

To summarize, we may draw the following conclusions:

- Feature extraction is effective and relatively simple.
- Feature evaluation can be done automatically by statistical means.
- Feature subset selection is sensible and useful.
- Extraction, evaluation and selection work well together.

Especially notable is how well the combination of feature extraction and selection works. With careful selection, even simple features may contribute significantly to recognition results.

7.4 Suggested Future Work

The feature extraction step can be expanded almost without limits. It is certainly possible to apply more advanced statistical means to extract more complex features. However, it should be kept in mind that the evaluation tends to suggest that complex features are not generally exceptionally good.

The evaluation and selection step could also benefit from more advanced statistics and information analysis. Sadly, it has been shown that the problem is computationally hard, and all efficient algorithms are bound to make use of heuristics or estimations. A relatively simple method to improve on the algorithms used would be to use a more dynamic subset size, so that activities that are poorly recognized are allowed to use more features.

The greatest single point of improvement upon the recognition results could well be using an inference engine with intrinsic time modeling, for instance a state-space model such as a Hidden Markov Model.

Bibliography

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, March 1999.

[2] A. Ali and J. K. Aggarwal. Segmentation and recognition of continuous human activity. In *IEEE Workshop on Detection and Recognition of Events in Video (EVENT'01)*, pages 28–, 2001.

[3] Hussein Almuallim and Thomas G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1):279–305, 1994.

[4] N. I. Badler and S. W. Smoliar. Digital representations of human movement. *ACM Comput. Surv.*, 11(1):19–38, 1979. ISSN 0360-0300.

[5] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, July 1994.

[6] A. Bobick and J. Davis. Real-time recognition of activity using temporal templates. In *Proceedings 3rd IEEE Workshop on Applications of Computer Vision*, pages 39–42, December 1996.

[7] L. Campbell and A. Bobick. Recognition of human body motion using phase space constraints. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 624–, 1995.

[8] C. Cédras and M. Shah. Motion-based recognition: a survey. *Image and Vision Computing*, 13(2):129–155, March 1995.

[9] Z. Chen and H. J. Lee. Knowledge-guided visual perception of 3-d human gait from a singleimage sequence. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):336–342, 1992.

[10] A. Fod, M. J. Matarič, and O. C. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12(1):39–54, January 2002.

[11] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, January 1999.

[12] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, pages 73–, 1996.

[13] Y. Guo, G. Xu, and S. Tsuji. Understanding human motion patterns. In *Proceedings of the 12th IAPR International. Conference on Pattern Recognition Conference B: Computer Vision & Image Processing*, volume 2, pages 325–329, 1994.

[14] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(1):1157–1182, 2003.

[15] D. C. Hogg. Model-based vision: A program to see a walking person. *Image and Vision Computing*, 1(1):5–20, 1983.

[16] T. Inamura, Y. Nakamura, H. Ezaki, and I. Toshima. Imitation and primitive symbol acquisition of humanoids by the integraded mimesis loop. In *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, pages 4208–4213, 2001.

[17] Odest Chadwicke Jenkins and Maja J. Matarič. Deriving action and behavior primitives from human motion capture data. In *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2002)*, pages 2551–2556, 2002.

[18] G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception Psychophysics*, 14:201–211, October 1973.

[19] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129, 1994.

[20] Steffen Knoop, Stefan Vacek, and Rüdiger Dillmann. Modeling joint constraings for an articulated 3d human body model with artificial correspondences in icp. In *Proceedings of the International Conference on Humanoid Robots (Humanoids 2005)*, 2005.

[21] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997.

[22] A. Kojima and T. Tamura. Natural language description of human activities from video images based on concept hierarchy of actions. *International Journal of Computer Vision*, 50(2):171–184, 2002.

[23] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning*, volume 1, pages 284–292, 1996.

[24] Igor Kononenko. Estimating attributes: Analysis and extensions of relief. In *Proceedings of the European Conference on Machine Learning*, volume 1, pages 171–182, 1994.

[25] Nojun Kwak and Chong-Ho Choi. Improved mutual information feature selecor for neural networks in supervised learning. In *Proceedings of the International Joint Conference on Neural Networks, 1999*, volume 2, pages 1313–1318, 1999.

[26] K. K. Lee and Y. Xu. Modeling human actions from learning. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2787–2792, 2004.

[27] Y. Li, T. Wang, and H. Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3):465–472, 2002.

[28] Huan Liu and Rudy Setiono. Feature selection and classification – a probabilistic wrapper approach. In *Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES*, 1996.

[29] T. Mori, Y. Segawa, M. Shimosaka, and T. Sato. Hierarchical recognition of daily human actions based on continuous hidden markov models. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 779–784, 2004.

[30] C. Myers, L. Rabiner, and A. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(6):623–635, 1980.

[31] R. Polana and R. Nelson. Low level recognition of human motion (or how to get your man without finding his body parts. In *Proceedings of IEEE Computer Society Workshop on Motion of Non-Rigid and Articulated Objects*, pages 77–82, 1994.

[32] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, 1989.

[33] Pedro Canotilho Ribeiro and José Santos-Victor. Human activity recognition from video: Modeling, feature selection and classification architecture. In *Proceedings of the International Workshop on Human Activity Recognition and Modelling 2005*, volume 1, pages 61–78, 2005.

[34] G. Rigoll, A. Kosmala, and S. Eickeler. High performance real-time gesture recognition using hidden markov models. In *Proceedings of the International*

Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction, pages 69–80, 1997.

[35] C. Rose, B. Bodenheimer, and M. Cohen. Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. *IEEE Computer Graphics & Applications*, 18(5):32–40, 1998.

[36] M. Rosenblum, Y. Yacoob, and L. S. Davis. Human expression recognition from motion using a radial basis function network architecture. *IEEE Transactions on Neural Networks*, 7(5):1121–1138, September 1996.

[37] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 702–718, 2000.

[38] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of the International Symposium on Computer Vision, ISCV'95*, pages 265–270, 1995.

[39] L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern Reconition*, 36(3):585–601, 2003.

[40] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 379–385, 1992.

Appendix A

Unabridged results

Table A.1. Recognition Rates for the Primitive Data Set

Activity	Correct	Ambiguous	Incorrect	Missed
bal	95.2	2.1	0.3	2.4
bow	99.0	0.5	0.0	0.5
call	97.7	1.1	0.2	1.1
clap	73.9	1.3	2.6	22.2
flap	99.2	0.4	0.0	0.4
hshake	53.1	0.4	2.2	44.3
kick	79.1	1.3	2.5	17.1
manip	55.4	0.1	1.9	42.7
sit	99.9	0.1	0.0	0.0
walk	69.8	0.3	0.6	29.3
wave	98.9	0.0	0.0	1.1

Table A.2. Recognition Rates for the Complete Data Set

Activity	Correct	Ambiguous	Incorrect	Missed
bal	99.5	0.0	0.0	0.5
bow	99.7	0.0	0.0	0.3
call	99.0	0.0	0.0	1.0
clap	99.7	0.0	0.0	0.3
flap	100.0	0.0	0.0	0.0
hshake	87.6	7.4	0.0	5.0
kick	94.6	0.0	0.0	5.4
manip	99.3	0.0	0.1	0.7
sit	97.1	2.9	0.0	0.0
walk	97.8	0.0	0.0	2.2
wave	100.0	0.0	0.0	0.0

Table A.3. Recognition Rates for the MIFS-5 Data Set

Activity	Correct	Ambiguous	Incorrect	Missed
bal	97.3	1.7	0.2	0.8
bow	97.3	2.0	0.0	0.8
call	65.2	4.0	0.0	30.8
clap	92.9	7.1	0.0	0.0
flap	99.4	0.0	0.0	0.6
hshake	72.5	0.0	1.9	25.6
kick	94.2	0.9	0.0	4.9
manip	99.8	0.0	0.0	0.2
sit	96.2	3.7	0.0	0.1
walk	97.0	0.0	0.0	3.0
wave	100.0	0.0	0.0	0.0

Table A.4. Recognition Rates for the HCFS-5 Data Set

Activity	Correct	Ambiguous	Incorrect	Missed
bal	94.8	3.5	0.6	1.2
bow	95.5	3.9	0.0	0.6
call	93.1	0.0	0.0	6.9
clap	99.7	0.0	0.0	0.3
flap	95.8	0.0	0.6	3.6
hshake	82.5	0.0	0.0	17.5
kick	86.9	0.0	8.2	4.9
manip	96.2	0.0	3.3	0.5
sit	94.8	4.2	0.3	0.7
walk	94.8	0.6	0.0	4.6
wave	100.0	0.0	0.0	0.0

TRITA-CSC-E 2006:028
ISRN-KTH/CSC/E--06/028--SE
ISSN-1653-5715

www.kth.se