Experimental Study on Arm Part Detection for Pose Estimation

Bas Kooiker

Bachelor Thesis Artificial Intelligence

Internal supervisor
Louis Vuurpijl
Faculty of Social Sciences
Radboud University, Nijmegen

External supervisors
Marten den Uyl and Jordi Bieger
Vicar Vision, Amsterdam

Abstract

Pose estimation is very important in fields like forensic research and surveillance applications. Control by gesture and pose are increasingly used in human-computer interaction and since the introduction of Microsoft's Xbox 360 Kinect device (1) it is becoming a standard in the gaming industry. High accuracy of pose estimation can be achieved through the use of high quality stereo camera's, but of course there is still room for improvement. However, good pose estimation using low resolution devices like webcams is still a very difficult task. Vincent van Megen (2) presented a system capable of estimating upper body pose in low resolution images with reasonable accuracy. In this thesis, local classifiers are explored which are trained to improve such a rough estimation by localizing the individual parts of human limbs. The use of different image features are compared as well as the use of different classification algorithms. The different detection and localization methods are evaluated on two different data sets, testing the generalizability of the detectors. HoG features and covariance features combined with neural network based arm part detectors turn out to be the most accurate, fast, and best generalizing on the tested data sets.

Contents

Acknowledgements

I would like to thank Louis Vuurpijl from the Radboud University Nijmegen as my supervisor during the course of this thesis project and his valuable lessons on research and science. I would also like to thank Jordi Bieger, my supervisor at Vicar Vision during my time at the company, for all his help and time spent on this project. I would like to thank Marten den Uyl for giving me the opportunity to work in his company and learn from the company's experience. I thank all the colleagues at Vicar Vision and SMR for their help, support and hospitality.

1 Introduction

Pose estimation and object detection are two research fields of great interest in image processing. There are many kinds of applications of pose estimation and many of these applications require a high performance regarding both accuracy and speed. Especially for control applications as in the game industry, both the criteria are equally important. Pose estimation should happen in real time. If the interaction with a person controlling an application and the application itself are not in sync, the sensation of control will be too unnatural to function properly. Also, the accuracy of pose estimation is very important, because no one wants an application to perform the wrong action, because the wrong motion or pose was detected by the application. In simple applications the gestures corresponding to different actions can be chosen such that they are not very alike and properly distinguishable by the gesture detector. However, as the applications and the control gestures become more sophisticated, the pose estimation also needs to keep up with that.

One goal of Vicar Vision in the development of pose estimation systems, was to make pose estimation work with low resolution image data. HCI application like those that work with the Microsoft Kinect device, use a lot of depth information in order to work properly. By developing pose estimation techniques that work with standard, relatively low resolution camera input the application field spreads from dedicated device environments to home computer-webcam environments or even smartphone applications.

In order to reach this next step in pose estimation development, we focus on local detection and localization of arm parts. Automatic pose estimation systems for single frames have been developed, but their performance may not be accurate enough for sophisticated gesture recognition. By enhancing these systems with local detectors for different body parts, the accuracy of the whole system can be improved. These detectors will have to be fast as there will be detectors for multiple body parts performing on each frame. They will have to be accurate, as they will have to track the exact location of the body part. The detectors will not need to be able to distinguish between everything else in an image and the detected body part, as they will be an enhancement to the system to improve a given pose estimation.

2 Background

2.1 Pose estimation

The research field of automatic pose estimation is very wide as well as the field of applications. Some surveys giving a good overview of the research are (3; 4; 5). Following Moeslund (3), applications of automatic pose estimations can be divided in three categories: surveillance applications, control applications and analysis applications. Surveillance applications are mainly about monitoring locations where many people move around, counting people, recognizing abnormal behavior and recognizing people from different viewpoints. Control applications are the main focus in Human-Computer Interfaces, where people want to control their computer using their body pose and motion. A very popular example of this kind of application is the before mentioned Xbox Kinect (1). Analysis applications can be used to gather information about the physical performance of people like athletes.

Different approaches to pose estimation vary in their use of body models. A widely used approach is model-based analysis-by-synthesis, which uses a model of the proportions and physical constraints of the human body to match on shapes and features found in images (6). Model-free approaches include probabilistic assemblies of parts and example-based methods. Probabilistic assemblies of parts use a bottom-up approach by first finding the individual body parts and then putting them together in a body configuration (7; 8).Finding the body parts is what will be done in this research. Example-based methods directly map 2D images to 3D configurations from train examples.

Some pose estimation algorithms predict 3D body configurations (2; 6), while other algorithms predict the body pose as a set of 2D points or shapes in the image (9; 10). The use of a third dimension in estimating body pose makes it possible to use a body model and its constraints to infer the positions of body parts. However, the third dimension also introduces some ambiguity that has to be resolved, which makes it more difficult.

2.2 Object detection

In object detection, systems are trained to discriminate between objects of different classes. As Papageorgiou and Poggio put it: "*A face detection system knows how to differentiate faces from "everything else", while a face recognition system knows the difference between my face and other faces"* (11). Different kinds of body part detection have been extensively researched. There have been many advances in face detection (12; 13), like the Viola-Jones face detector which uses a boosted cascade of Haar-like feature classifiers (14). Detecting faces can be a goal in itself, it can also be very useful to support other tasks like pose estimation. Van Megen's pose estimation algorithm first localizes the face in an image in order to initialize the search window for the upper body, assuming that the upper body is below and relative in size to the face. Hand detection (9; 15), as well as hand recognition for different poses of the hand (15; 16), is also very much researched for applications in sign language recognition.

2.3 Research goals

The primary research goal is to improve on rough estimates of body part locations through the use of local classifiers. Such rough estimates may be given by detections in earlier frames, in the case of video, or by a rough global estimator, like Van Megen's neural network based pose estimator. This work will focus on the latter, and only still images (i.e. no videos) are considered.

Figure 1 An image with the desired locations of arm part detection.

To accomplish this, good feature extractors and classification algorithms need to be selected, implemented and optimized. This thesis compares the performance of multi-layered perceptron neural networks, linear support vector machines and boosted cascades using (combinations of) pixel intensity, color, histogram of gradient and covariance (of color and first and second order derivatives) features.

Accuracy and computational performance will be considered in the context of improving body part location estimates. These local classifiers can help to achieve this goal in a couple of ways:

- By evaluating the classifier on every location in a window defined by the original estimation, the actual location of the body part can be "detected".
- When fitting a body model to the image, local classifiers can provide information about the goodness of the current fit.
- Some local classifiers may even be able to self-correct and point out a next location that is more likely to contain the body part.

This thesis will pursue the following three research questions:

(1) Which image features are best suited to detect arm parts in images?
(2) Which classification algorithm is best suited for detecting arm parts in images?
(3) Can additional location specific information be learned in order to improve the speed and accuracy of a detection algorithm?

Both accuracy and speed will be considered in all the research questions.

2.4 Organization of thesis

In chapter 3 of this thesis, the proposed method for arm part localization will be described, with in section 3.1 an overview will be given of the used image features: pixel intensities, color, histogram of oriented gradients and covariance features. In section 3.2, the used classification algorithms will be described: multi-layered perceptron neural networks, linear support vector machines and cascades of GentleBoost classifiers. In chapter 4, the data sets used for training and testing will be described. In chapter 5, the method for evaluating the classifier will be described. Chapter 6,7 and 8 will be dedicated to respectively the results, the discussion, and the conclusion of the experiment.

3 Methods

The system that will be used in this experiment will consist of a number of components. The system will get an image and a prior estimate of the body part location as its input in order to calculate a new (and better) estimated location of that body part. Using a sliding window algorithm, image clips will be extracted from the image at different locations. Image clips are small regions from larger images that will be evaluated by the classifier to determine whether it contains the specified body part or not. The image clips all have the same size, but the sliding window algorithm could be extended to detect body parts at different scales. The image clips serve as input for the feature extractor, which will represent the clip as a feature vector. The classification algorithm will calculate an activation value corresponding to this feature vector. The activation pattern resulting from the sliding window will be smoothed out by means of convolution with a Gaussian distribution. This way, the system will be robust against noisy output from the classification algorithms.

$Input = \{image\ I, prior\ estimation\ X_{est}\}$
$System\ components = \{feature\ extractor\ F, a\ classification\ algorithm\ C\}$
1. $Use\ sliding\ window\ based\ on\ X_{est} to\ calculate\ activation\ matrix\ A$
$For\ each\ window\ w$:
$(a) Select\ image\ clip\ I_{clip,w}\ from\ I$
$(b) Extract\ feature\ vector\ V\ using\ feature\ extractor\ F(I_{clip,w})$
$(c) Calculate\ activation\ value\ A_w\ using\ classifier\ C(V)$
2. $Convolve\ A\ with\ Gaussian\ distribution$
3. $Determine\ new\ estimated\ location\ X_{new}\ corresponding\ to\ argmax_w(A)$

The localization algorithm

This thesis is about finding the best combination of feature extractors and classification algorithm for the task of detecting and localizing body parts. The size of the image clips, the sliding window parameters and the size and standard deviation from the Gaussian distribution matrix which is used for convolution can all be optimized for each classifier. During the preparations for this experiment, good settings for all these values were found and kept constant throughout this experiment.

3.1 Feature extractors

To train the classifiers, there are different kinds of image features that can be used to represent the information in the image clips. These different kinds of feature representations are useful in different kinds of situations. In images, there is information about color, texture, shape, rotation, light intensity, et cetera. Human body parts have certain characteristics. Certain body parts, when people are viewed from a specified angle, always have roughly the same shape. Upper and lower arms are more or less cylinder shaped and shoulders have the same round edge as long as the arms are alongside the body and someone isn't wearing shoulder pads. Hands however, differ very much in shape because there are five fingers that can all point in different directions. People wear all kinds of clothes, so the color and texture of most body parts vary very much between people in different situations. But as long as people do not wear gloves, hands always show similar texture and skin color. As long as the camera does not move and the people stand or sit in regular positions, the shoulders will only rotate in one direction. The upper arms can only rotate in a limited space and the lower arms and hands can rotate in almost any direction. In order to discriminate between body parts with these different kinds of characteristics, different kinds of image features may be useful.

Some of these image features take up more computation time than others. This should also be taken into consideration when selecting image features for classification.

3.1.1 Pixel intensity features

Pixel intensity features are tested because they are also used in Van Megen's system (2) in addition to the optical flow feature information that is unavailable in still images. Pixel intensities are very fast to compute and they can be useful in estimating the rough shape of objects in a certain orientation. In combination with neural networks they have also been used to detect frontal faces (13). Pixel intensity features are sensitive to variations in lighting, color and orientation.

When converting a color image to a grayscale image, the pixel intensity for every pixel p is calculated by

$$intensity_p = red_p \times 0.3 + green_p \times 0.59 + blue_p * 0.11$$

Pixel intensity features are extracted by scaling an image clip down to 20x20 pixels and converting it to a greyscale image, resulting in a feature vector of length 400. All the pixel intensity values are divided by 255 in order to scale them between zero and one. By using a fixed, small size for the greyscale image the feature vector size is kept relatively small and the classifier can be used on different scales in an image.

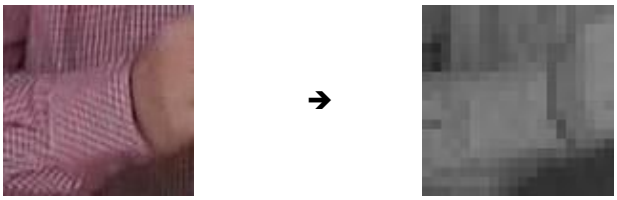

Figure 2: Conversion from image clip to 20×20 greyscale image

3.1.2 Color histograms features

Due to the great variation in color and texture of clothing, color information is not expected to be very useful in detecting body parts like shoulders and elbows. However, color information has been proven to be very useful when detecting hands (9). Skin color does vary between people, but it still spans a relatively small part of the full color spectrum.

Color information is extracted from the images using color histograms. In order to separate information about lighting conditions from the actual color/hue of a pixel the HSV (Hue, Saturation, Value) color space was used. When calculating a color histogram over an image clip, the color spectrum is separated into 8 hue bins. Each hue bin gets a value corresponding to how much of the clip is filled with the colors corresponding to the hue bin. This histogram is then L2 normalized and appended with the means and standard deviations of the saturation and value of the clip. Each image clip is separated into 5x5, non-overlapping color histograms, causing the feature vector to be sensitive to position and rotation.

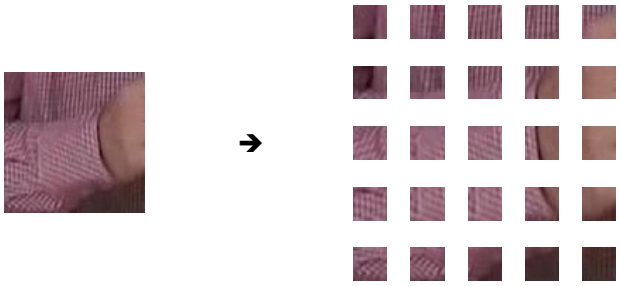

Figure 3: Extracted non-overlapping image clips, used for color histogram computation

3.1.3 HoG features

Histograms of oriented gradients have been used to detect a variety of objects, like pedestrians, bikes, cars and various other kinds of objects in a wide variety of environments (18; 19; 20). HoG features are calculated in a number of steps. First, the image gradient is computed by filtering the grayscale image with the discrete derivative masks [-1, 0, 1] and $[-1, 0, 1]^T$. The evaluated image clip is divided in overlapping blocks, which consist of a number of cells. Each pixel casts a vote in one of the 8 orientation bins for the cell in which it lies. Each cell has the same number of orientation bins. The next step is concatenating the values from the bins in all the cells from one block, and L2 normalizing the resulting vector. This way of normalizing in relation to the surrounding cells, is applied in order to account for changes in lightning. The blocks are organized in such a way that they overlap with half the block size. By calculating HoG features in such a dense grid, it is very useful for object localization.

When extracting the HoG features, the blocks in this work always consist of 3x3 cells, like Dalal and Triggs (18) found as the optimum setting. The number and size of the blocks is varied in order to find the optimum for different kinds of body parts. The image clip is split into $n \times n$ blocks, overlapping by half the size of the block. Therefore, the block size is $2 * clip\ size/(n+1)$. In this experiment, the clip size was always set too 100 pixels.

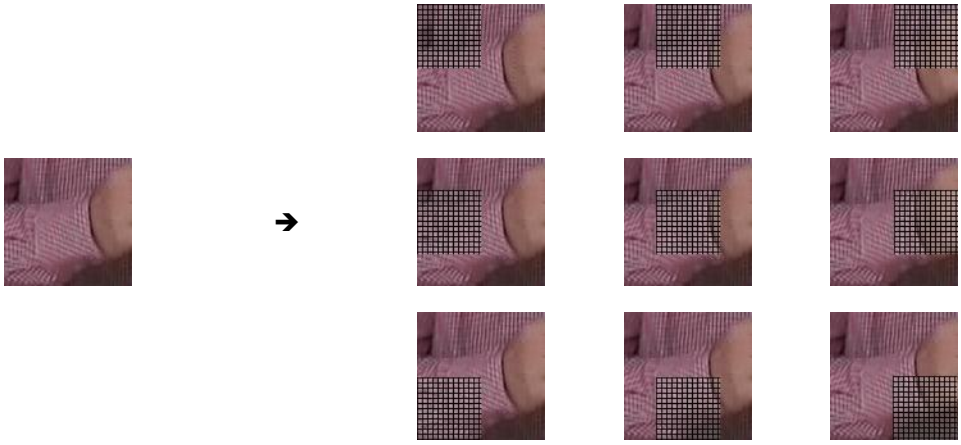

Figure 4: Extracted overlapping HoG descriptor blocks from image clip for n = 3

3.1.4 Region covariance features

Covariance is a measure of linear relation which is much used in statistics and probability theory. It measures how two random variables change together. Tuzel, Porikli and Meer (21) successfully used these image statistics for object and texture recognition. They calculated the covariance between different features in image regions, namely pixel location, RGB-color and the norm of the first and second order derivative of the intensities in the x- and y-direction. Covariance matrices from overlapping image regions were used to describe different kinds of objects including faces and pedestrians. The resulting feature descriptors are to a large extend invariant to changes in illumination and rotation. These region covariance features were also used by (22).

The covariance matrix is represented by $d \times d$ covariance values for a given region R with d different image features. These covariance values are calculated as

$$C_R = \frac{1}{n-1}\sum_{k=1}^{n}(z_k - \mu)(z_k - \mu)^T$$

where, n is the number of points in the region, z_k are the d-dimensional points in R and μ is the mean of these points.

In this experiment, covariance matrices were calculated using similar image parts as in (21). Pixel locations were omitted and an opponent color space is used instead of RGB. Regions were selected on a variable number of scales around the same center. By using multiple regions with the same center on different scales, the feature descriptor preserves its property of being invariant to rotation changes and increases its location sensitivity, which is important for detecting objects accurately. This number of n scales was varied equally with the $n \times n$ number of HoG descriptor blocks. Block B_i had size *clipsize*(n-i)/n*.

Figure 5: Extracted region covariance feature blocks for n = 3

3.2 Classification algorithms

In object detection different kinds of classification algorithm have been used which all have different characteristics. The three evaluated algorithms and how they are used, will shortly be described here.

3.2.1 Neural networks

Neural Networks (NNs) have been applied to many aspects of image processing (23). They have been used for various kinds of object recognition like face detection (13). NNs were also used for Van Megen's pose estimator (2). For this work, multi-layered perceptron neural networks were used because they have been proven to work very well for classification and are simple in use. NNs can be trained such that image clips containing a part of the object may receive more activation than clips not containing any part of the object, but less activation than image clips with the whole object. This makes them very well suited for object localization. NNs can also have multiple output values for one classification network without significantly affecting computation time.

The output of the NNs will represent the classification and the distance to the body part. This value will lie between zero and one where zero means that the body part to be detected is not close enough to the image clip and one means that the body part is exactly in the center of the image clip. The train set is constructed in such a way that there is a linear relation between the distance from the center of the image clip to the center of the arm part and the output value. The output is set up to be zero at a distance of ¼th of the image clip size shifted from the arm part. In this experiment this is 25 pixels. This way, the network does not need to discriminate perfectly between clips that are shifted five pixels from the body part and clips that are shifted six pixels from the body part. When using the network for classification, a threshold will be set at 80% activation, which corresponds to five pixels distance.

For the second research question, neural networks will be trained with four additional outputs. These outputs will represent search direction information to be provided to a search algorithm. By learning information about in which direction an arm part is to be found, the best estimated position of an arm part can be found in much less steps than with the commonly used sliding window algorithm. Every output value represents one direction: up, down, left or right. When the body part is exactly in the center of the clip, the four the values will be zero. When the clip should be shifted in the down-left direction, the down- and the left-output value will be activated, and the up- and right-output will be zero. These output values will be proportional to the distance between the center of the clip and the center of the body part. When the clip does not contain the body part at all, all four the output values will be one.

All the networks were trained with the same parameters: 10 units in the hidden layer, the learn rate is set to 0.005 for all the weights and momentum is set to 0.001. A pilot study showed that these parameters always gave more or less the best results. After every training epoch, the network was evaluated on the stop set. The network which performed best on the stop set in terms of balanced accuracy (see chapter 5) on body part detection was saved and training was stopped when there has been no better performance on the stop set for 150 epochs. All networks were trained for a minimum of 500 epochs. FamProp, an in-house neural network implementation was used in this experiment.

3.2.2 Support Vector Machines

Support vector machines (SVMs) have also been used in quite some object detection applications (18; 19; 20; 11). SVMs do not suffer from local minima when they are trained and are much less prone to over-fitting than NNs. Therefore, unlike the NNs, SVMs do not need a separate stop set in the training process. This way, the data from the stop set can also be used to train the SVMs additionally to the train set which is used to train the NNs.

In this experiment, SVMs are used for classification, not for regression. Unlike the NNs, the target output of the train samples will not linearly decrease as the sample is farther shifted from the body part. Instead, the samples which are shifted from the body part 5% of the clip size or less (i.e. 5 pixels in this work) get a target output of 1 and the rest of the train samples gets target output 0.

For this experiment, the approach used by Dalal and Triggs (18) will be followed by using a linear soft margining support vector machine with C = 0.001. The implementation of SVM in OpenCV (24) is used in this experiment.

3.2.3 GentleBoost Cascades

Cascades of boosting classifiers have been increasingly used for object detection because of their high classification speed and good performance. The idea behind boosting is that the use of multiple simple classifiers with accuracy at least better than chance can lead to more accurate classification when combined the right way (25). The main idea behind cascading is that classifiers can be chained together so that each classifier tries to reject as many train examples as possible. All the train examples that are not rejected by the current classifier are sent to the next classifiers in the cascade. The examples that are not rejected in any of the cascade steps are classified as positive.

In principle, any kind of classifier can be used in a cascade. A common classification algorithm in cascading is AdaBoost (26), which is also used in the Viola-Jones face detection algorithm (14). In every boosting step a linear classifier using a single feature is trained for every single feature. These classifiers are called weak classifiers. When training these weak classifiers, every train sample gets a weight. Train samples which are misclassified get a higher weight than the correctly classified train samples so the classifiers will focus more on the harder samples. The weak classifier with the smallest error is added to a strong classifier, which will be used as one step in the cascade. A total of M strong classifiers are trained and cascaded.

$$1.\ Start\ with\ weights\ w_i = \frac{1}{N}, i = 1, \dots, N$$
$$2.\ Repeat\ for\ m = 1,2, \dots, M:$$
$$(a)$$

$$1.\ Start\ with\ weights\ w_i = \frac{1}{N}, i = 1, \dots, N$$
$$2.\ Repeat\ for\ m = 1,2, \dots, M:$$
$$(a)\ Fit\ weak\ classifier\ f_m(x) \in \{-1,1\}\ for\ every\ feature\ using\ weights\ w_i\ on\ the\ training\ data$$
$$(b)\ Compute\ err_m = E_w\left[1_{(y \neq f_m(x))}\right], c_m = log\left(\frac{1 - err_m}{err_m}\right)$$
$$(c)\ Select\ weak\ classifier\ f_m\ with\ minimum\ err_m$$
$$(d)\ Set\ w_i \leftarrow w_i\ exp\left[c_m 1_{(y_i \neq f_m(x_i))}\right], i = 1,2, \dots, N, and\ renormalize\ so\ that\ \sum_i w_i = 1$$
$$3.\ Output\ the\ classifier\ sign\left[\sum_{m=1}^{M} c_m f_m(x)\right]$$

Comment [JB1]: What is E_w?

The AdaBoost learning algorithm as in (27).

For this experiment, GentleBoost (27) is used. GentleBoost uses a different update function for the weights and may use a linear combination of multiple single-feature regression classifiers as weak classifiers rather than only single-feature regression classifiers.

$$1.\ Start\ wit\square\ weig\square ts\ w_i = \frac{1}{N}, i = 1, \dots, N, F(x) = 0$$
$$2.\ Repeat\ for\ m = 1,2, \dots, M:$$
$$(a)\ Fit\ t\square e\ regression\ function\ f_m(x)\ by\ weig\square ted\ least - squares\ of\ y_i\ to\ x_i\ wit\square\ weig\square ts\ w_i$$
$$(b)\ Update\ F(x) \leftarrow F(x) + f_m(x)$$
$$(c)\ Update\ w_i \leftarrow w_i \exp[-y_i f_m(x_i)]\ and\ renormalize$$

$$3.\ Output\ the\ classifier\ sign[F(x)] = sign\left[\sum_{m=1}^{M} c_m f_m(x)\right]$$

The GentleBoost algorithm as described in (27).

As these cascades are trained on rejecting negative examples until the evaluated example must be a positive example, it is best to train cascades with as much negative examples as possible. This is done by taking every possible 100×100 clip that is not a train example. This way of training was not possible with the NNs and the NNs because these algorithms would then classify everything as negative and the MSE would still be low.

For this experiment weak classifiers are added to the strong classifiers until 40% of the negative examples are rejected while retaining a performance of 99% on the positive examples. The algorithm stops when the false alarm rate is below 0.001. For The first strong classifier, only HoG and color features are used. This way, the most negative examples can be rejected without calculating the computationally expensive covariance features.

4 Data

For training and testing, two different data sets are used: the *eating se*t (see Section 4.1) and the *daily activities set* (Section 4.2). For training the NNs, the eating set is split into a train set, a stop set and a test set. 60% of the data is used to train the networks and the stop and test set both consist of 20% of the data. When training the SVMs or the boosted cascades, both the train and stop set are used as train samples, as these algorithms are less prone to over fitting. The daily activities set is only used for testing the generalizability of the detectors.

4.1 Eating set

The video material that is used to train the body part detectors was recorded by Vincent van Megen (2). It is the same video material that was used to train his pose estimation system. The videos contain people, sitting on a table, eating something. The setting is similar to people sitting behind a desk, viewed from a webcam. HCI applications for people behind a computer may work under the assumption that the upper body of the subject is always visible, does not move significantly to the front or back and the subjects are always facing the camera. The data is therefore very well suited for the target application of the classifiers.

Figure 6: Example images from the Eating data set

There are a total of 13 different people with different kinds of clothing in the different videos. There is a total of 2095 frames. In their survey, Moeslund and Granum (3) describe the most common assumptions on the conditions for motion capture that are associated with different contributions to the field of human motion capture. Even though this system will not use motion information, some of the assumptions can still be useful to compare the results with the results of other systems.

Assumptions related to movement are:

- There is no camera movement
- There is only one person in the workspace at a time
- The subject faces the camera at all times

An assumption related to appearance is:

- There is static background

All the video material is hand annotated with ground truth for the positions of the seven body parts: hands, elbows, shoulder and neck. This annotation is represented by 3-dimensional coordinates. The x- and y-coordinates are estimated by eye. The z-coordinate is calculated from the x- and y-coordinates, pre-defined body constraints and the prior knowledge that the subject in the video is facing the camera.

In order to train detectors for the individual body parts, specific data has to be extracted from the video material. These data and the corresponding ground truth should make it feasible to classify an image clip as containing the body part as well as using the classifier to localize the body part in a full image. The data that will be used to train the local classifiers will consist of clips with equal size, taken from the video frames. Every clip will have a corresponding ground truth. Classifiers typically have binary output. In this case, positive classification would mean: the body part is close to the center of the image clip. Negative classification would mean: the body part is not close to the center of the image or not in the image at all.

The data is split up in a train, stop, and a test set. Data from videos of eight different people, containing 1285 frames, are used as train set and data from three other people containing 400 frames is used as stop set. Data from the remaining two people, containing 410 frames, is used for testing the networks and the performance of the local classifiers for localization.

4.2 Daily activities set

Another data set will be used to test the generalizability of the detectors. If people will be using HCI applications in front of their computer, there will probably be background clutter. Illumination conditions will vary between users and between different sessions with the same user or even within one session. This second data set is used to test the performance on images with previously unseen conditions and subjects. The data set consists of frames taken from the University of Rochester Activities of Daily Living data set, which was used in (28). The data set consists of five people, each performing a number of activities three times. The activities include answering a phone, dialing a phone, looking up a phone number in a telephone dictionary, writing a phone number on a whiteboard, drinking a glass of water, eating snack chips, peeling a banana, eating a banana, chopping a banana, and eating food with silverware.

Figure 7 Example images from the Rochester Daily Activities data set

There are a number of differences in assumptions on the classifiers related to this data set with the Eating data set. There is still:

- no camera movement

- only one person in the workspace at a time
- a static background

However, there are some differences.

- There are some objects in the background, so the classifiers will somehow have to differentiate between the foreground and the background.
- The subjects in the videos sometimes move forward or backward, so the scales of the body parts are not always the same.
- The subjects sometimes turn around. Frames, in which the subject is turned away from the camera, are not used for testing, but there are frames in which the subject is not exactly facing the camera.

For testing the classifiers, a total number of 128 frames were taken from five of the activities performed by two different people. Two different people per activity were it would be possible to analyze whether the performance of a classifier varies due to the different activities or due to the different people. A number of 128 frames is sufficient to get a reasonable performance measure of the generalizability. The data was hand annotated the same way as the eating set was annotated.

4.3 Training data

In order to train the classifiers, clips are extracted from the frames of the train set. Clips are small pieces of an image with a dimension of 100x100 pixels. A total of five clips per frame are extracted. The clip with the body part exactly in the center is the first clip to be extracted. The clip is shifted with a randomly chosen distance with a maximum of half the clip size, 50 pixels, in either the x- and y-direction. In this new position and the second clip is extracted. This is repeated for the same body part on the other side of the subject's body where the clips are mirrored. A fifth clip is taken randomly in the image, not near the body part, as a negative train example. Not more than one random clip is taken because that could cause the classifier to learn to classify everything as false.

By extracting image clips at different distances up to 50 pixels in both the x- and y-direction, the trained classifiers should be able to locate the exact body part location in a window with the same size as the image clip. The maximum pixel distance of the train samples can thus be $\sqrt{50^2 + 50^2} \approx 70.1$ pixels. The non-body part examples can be further away. All the train samples with a pixel distance of over 25 pixels will have a corresponding target output of zero and train samples with a pixel distance of 5 pixels or less will have a target activation of 0.8 or more. Therefore, these will be classified as true.

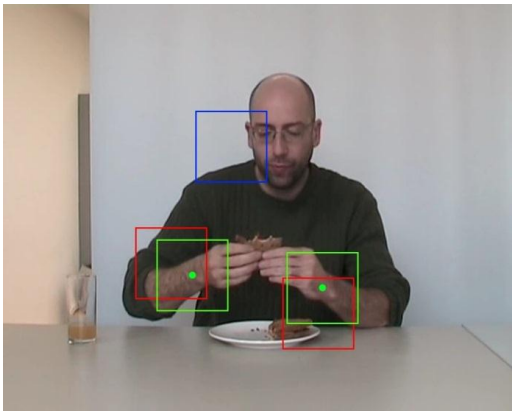

Figure 8 Extracted image clips for one frame for training a right wrist detector: the green dots represent the annotated points, the green squares are the positive train examples, the red squares are the negative train example, and the blue square is the additional negative example. Clips taken near the left wrist are mirrored.

The training data for the direction prediction network outputs is generated with the same samples. The four direction outputs can represent a total of ten different states: do not look any further; look in one of eight directions (up, up-right, right, down-right, etc.); unknown where to look. Figure 9 show the output activation patterns that correspond to the different classes of shifted image clips.

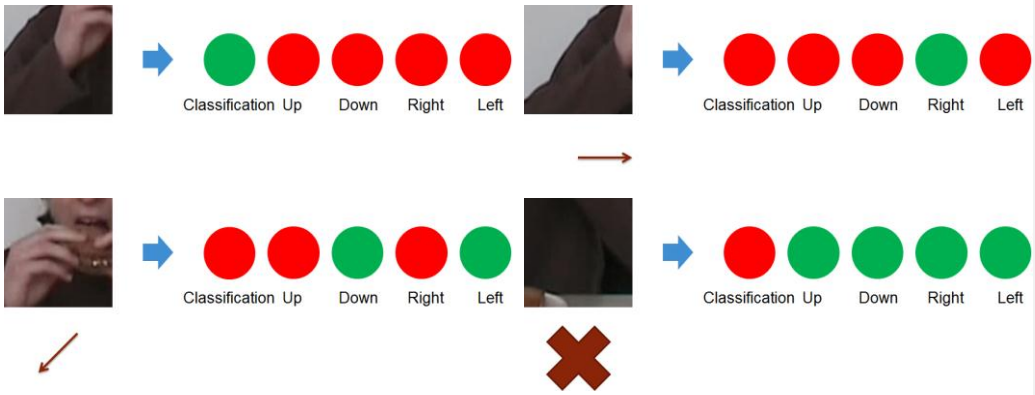

Figure 9 Four train examples with the corresponding target output.

5 Evaluation

The performance of all the classifiers will be evaluated on the train set, the stop set, the test set and the additional test data from the daily activities set regarding accuracy and computation time. The accuracy will be defined as the balanced accuracy, which is the average between the sensitivity and the specificity. This way, the classifiers are as much evaluated on their performance on positive image clips as on their performance on negative image clips. Balanced accuracy is defined as follows:

$$balanced\ accuracy = \frac{sensitivity + specificity}{2}$$
$$= \frac{0.5 \times true\ positives}{true\ positives + false\ negatives} + \frac{0.5 \times true\ negatives}{true\ negatives + false\ positives}$$

The performance will also be measured in localization error. This is the distance between the location of the annotated point in the ground truth and the estimation of the location of that point by the search algorithm. The search algorithm utilizes a sliding window around the location of the annotated point in the ground truth. The sliding window has a range of 75×75 pixels around the point and a step size of 3 pixels bringing the total number of evaluations to $25 \times 25 = 625$ evaluations per frame. This creates a matrix of classifier outputs. This activation matrix is convolved with Gaussian distribution matrix of size $7 \times 7$with $\sigma = 1.5$ to filter out outliers and to find the center of more densely activated areas. The position corresponding to the point with the highest activation in the activation matrix is presented as the estimated position for the arm part.

The computation time will be measured as the required time to perform the sliding window algorithm on one frame. This includes transforming the color image to a greyscale intensity image, extracting the feature vectors at 625 different positions, using the classification algorithm to get an activation for the position, and proposing an estimate position for the body part.

6 Results

6.1 Feature extractors

Different combinations of feature extractors to provide the input for the classification algorithm, will result in different performances of the final detector. Figure 10 shows the behavior of the neural network based detectors on an image taken from the train set. These are the activation maps for hand detectors, since hands are most important in many applications like action recognition and HCI.

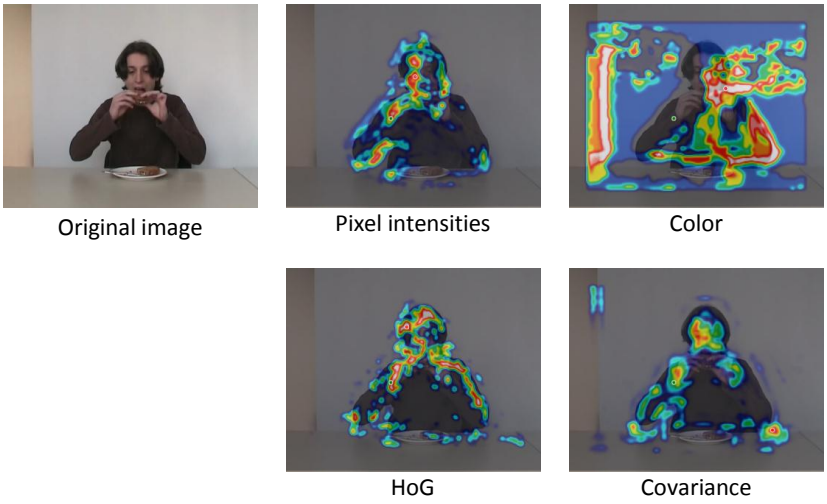

Figure 10 The activation patterns of the NN based hand detectors each using one of the four feature extractors.

The pixel intensity based detector has an activated peak close to the center of the hand, but it is also activated anywhere else in the image on the edge between a darker and a lighter area. The face is almost the same darkness as the hand and therefore gets a lot of activation. The activation pattern of the color based detector is very messed up. This is not as much because of the color features. The color feature extractor normalizes the values of the hue bins, but it also adds a value to the feature vector which is higher than one. Since neural networks have a hard time dealing with very large values and large scale differences between them, all NN detectors that use color features are incapable of producing acceptable accuracy rates. Also, these networks weren't trained on the whole image, which may result in bad performance on regions that were not in the train set. The activation pattern of the HoG based detector is also inconclusive. It has very strong activation in the region of the hand, but it also has very strong activation in a lot of other regions. The region covariance based detector does not have too much excessive activation, but the activation peak that lies close to the hand is not too strongly activated either.

Figure 11 shows the performance error of the SVMs based on different feature extractor combinations. For instance HoG-2 shows the average of all the pair wise combinations of feature extractors containing the HoG feature extractors (HoG+PI, HoG+Color, HoG+Covariance). These were also averaged over the all the values for parameter n. The error rates are all averaged over the five evaluated body parts. For both the classification algorithms, the graphs show that on average, the error is constantly increasing when adding more feature extractors to the detectors. The HoG feature based detectors increase in performance when one additional feature extractor is added. The color

feature extractor performs quite bad on average. This is in part explained by the bad normalization, but also because it is only useful when detecting skin, which is only present in the hand detector, and not with the four other detectors.

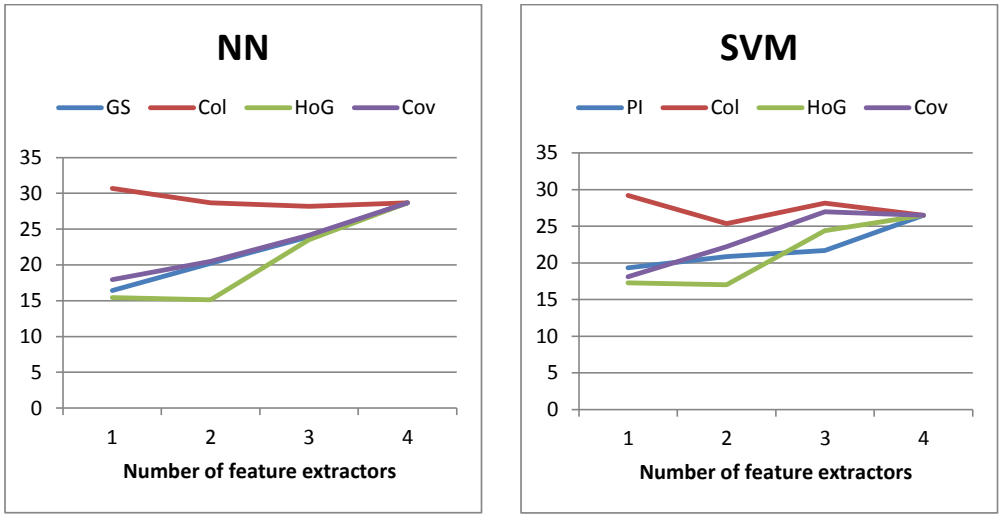

Figure 11 The average performance of the different feature extractor combinations with different number of combined extractors for the different feature extractors.

A difference was expected to be found between the two classification algorithms, but this does not really seem to be the case. One would expect that adding more information would always lead to better performance with the SVM based detectors, as they should be less prone to overfitting than NNs, but this is not reflected in the results.

6.2 Classification algorithms

The results for the different classification algorithms will be presented here. First, the performance results of the different algorithms will be presented individually. Then, their performance will be compared to see which performs best. Their generalization ability will be discussed as the different classification algorithms performance will be measured on the second data set.

6.2.1 Neural networks

As stated before, the neural networks did not work well with the un-normalized data from the color feature extractor. Therefore, this may not be a fair comparison, as these features do seem to be useful as they are used with some of the best SVM based arm part detectors. Nevertheless, the detectors perform quite well when localizing the different arm parts, as the table below shows. The table shows the arm part detectors trained on the feature extractor set with the best performance in terms of pixel distance for each individual arm part. N represents the number of descriptor blocks for the covariance feature extractor and the square root of the number of descriptor block for the HoG feature extractor as explained in section 3.1.3 and 3.1.4. All but one of the detectors use HoG features. Three out of five use region covariance. It would be expected that more feature extractors give more information about an image, so it would result in better performance, but for the neural networks this is clearly not the case. As shown in table 1, the best performing neural network based detectors use combinations of one or two different kinds of feature extractors.

			Test 1		Test 2		
Arm part	Feature extractors	N	Pixel error	Accuracy	Pixel error	Accuracy	Time
Right shoulder	Greyscale, Covariance	3	6.08	0.86	23.57	0.54	0.18
Right upper	Covariance, HoG	3	8.41	0.68	25.53	0.46	0.25
Right elbow	HoG	5	6.79	0.76	23.96	0.52	0.18
Right lower	Covariance, HoG	5	10.26	0.70	23.55	0.49	0.33
Right hand	Greyscale, HoG	6	9.45	0.69	22.57	0.56	0.27

Table 1 The performance of the best performing NN based arm part detectors.

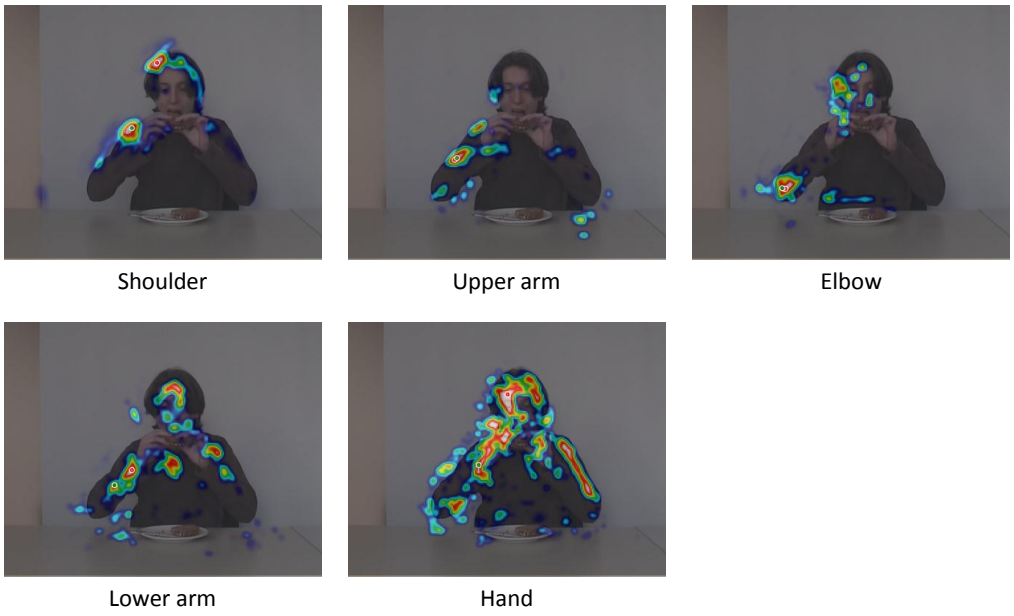

Figure 12 The activation maps of the best performing NN based arm part detectors.

6.2.2 Support vector machines

The best set of feature extractors for each arm part detector based on SVMs are shown in table 2 and figure 14. All SVM-based detectors use HoG features. The SVMs are better capable of dealing with un-normalized data, so the color features are also used well in detecting hands. The training data for the SVMs was set up such that it would be able to discriminate between points that lie close to the arm part and points on the arm part itself. Therefore, there are a lot of areas that get activated by the detector. The SVMs seem to learn to deactivate when evaluating a clip near the body part, but not on the body part. All the points more than 50 pixels away from the body part are therefore not deactivated. The shoulder detector and the elbow detector have a very compact activated region on the body part, which is good for localization. The activation peak on the hand is much smaller, but it is a peak nevertheless.

			Test 1		Test 2		
Arm part	Feature extractors	n	Pixel error	Accuracy	Pixel error	Accuracy	Time
Right shoulder	Greyscale, Covariance, HoG	3	7.10	0.84	18.46	0.61	0.26
Right upper	Greyscale, HoG	6	18.86	0.77	26.45	0.58	0.20
Right elbow	Greyscale, HoG	6	13.45	0.76	28.13	0.55	0.20
Right lower	Color, Covariance, HoG	6	19.32	0.68	25.31	0.58	0.39
Right hand	Color, HoG	5	17.61	0.73	23.55	0.63	0.28

Table 2 The performance of the best performing SVM based arm part detectors.

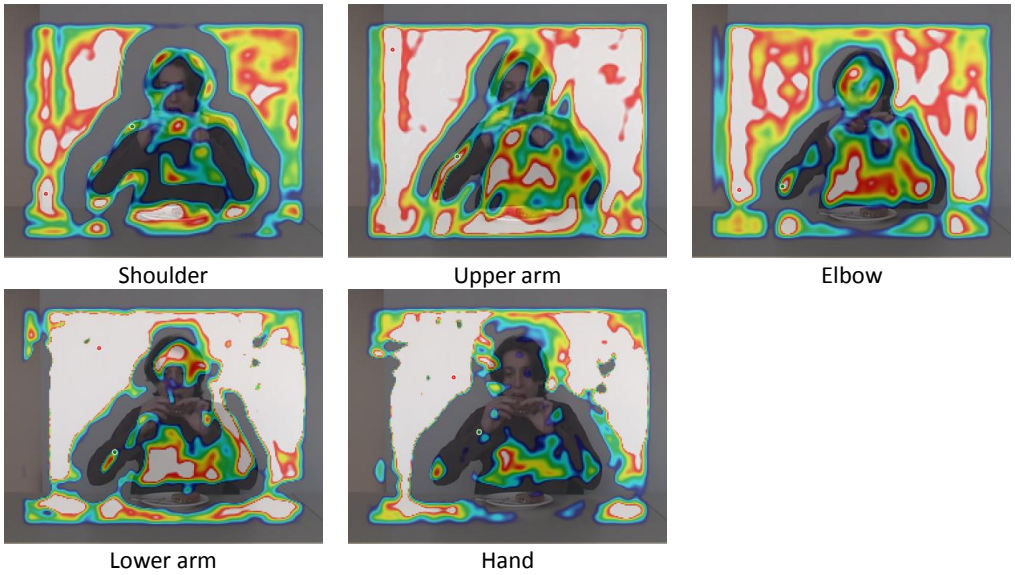

Figure 13 The activation patterns of the best performing SVM based arm part detectors.

The SVMs are indeed able to learn to distinguish between a point on a body part and a point close to a body part, but in order to use SVMs as body part detectors without good initial guesses, they require to be trained with more non-person data, and data from other parts of the body.

6.2.3 Boosted cascades

Table 3 and figure 15 are the performance table and the activation maps corresponding to the boosted classifier detectors. There is not too much activation in the images other than in the region of the detected arm part. This fits the expectation because the cascades were trained with all the background patches as negative examples since the cascade learning algorithm deals much better when training on a lot of negative examples than the NNs or the SVMs. However, this way the cascade is not especially trained on distinguishing body parts from near-body parts. The shoulder has one big white area and two small white areas and the big white area belongs to the shoulder. For the elbow, the same holds, although the elbow detector does have more areas with a little bit of activation than the shoulder detector. The hand detector does have quite a lot of activation in the wrong areas, but the detection error is still rather low, which is good. The detectors of the upper and the lower arm have a long stretched activated area alongside the arm. This makes sense, because a detector cannot know whether it is detecting the center of a cylinder shape without detecting the endings of the cylinder shape.

	Test 1		Test 2		
Arm part	Pixel error	Accuracy	Pixel error	Accuracy	Time
Right shoulder	7.54	0.75	26.53	0.63	0.64
Right upper	17.61	0.67	55.95	0.50	0.83
Right elbow	14.02	0.59	39.09	0.52	0.74
Right lower	25.55	0.52	37.28	0.47	0.92
Right hand	18.48	0.53	30.11	0.62	1.00

Table 3 The performance of the boosted cascade based arm part detectors

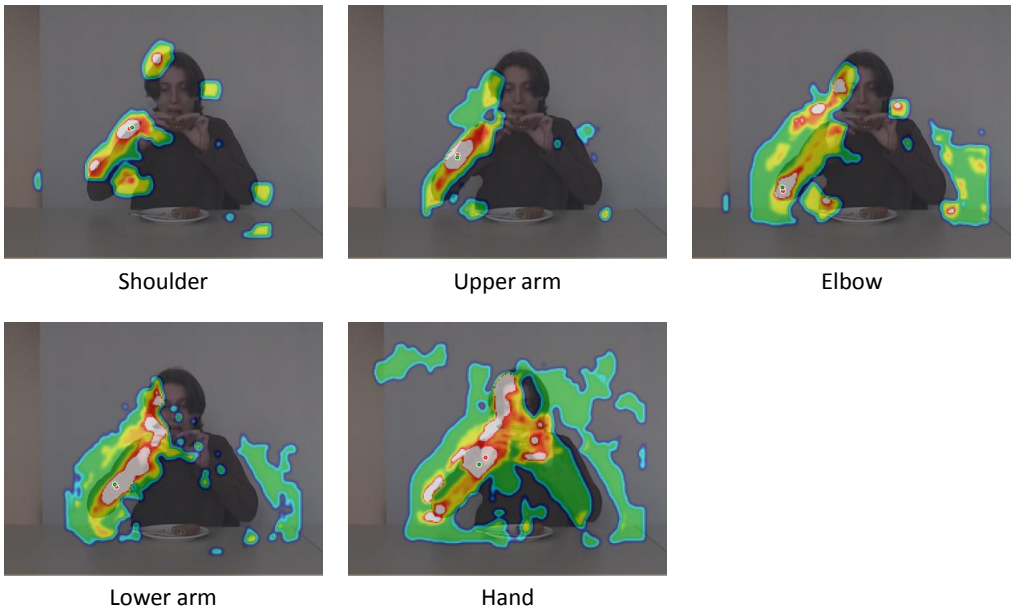

Figure 14 The activation patterns for the boosted cascade based arm part detectors.

One of the ideas behind a boosted cascade is that the learning algorithm itself figures out which feature extractors to use and at which positions in the image clip. Therefore, there will not be a comparison between feature extractors. An overview will be given of the feature extractors that were selected by the learning algorithm.

The cascades for the different arm parts were all trained to have a maximum of ten strong classifiers. Each strong classifier was allowed to have a maximum of 25 weak classifiers, except for the first two strong classifiers, which could hold up to two weak classifiers. This way, the first two steps of the cascade are guaranteed to be fast, so many of the negative clips, over 84%, will not take up too much computation time of the detection algorithm.

To illustrate the behavior of the boosted cascade learning algorithm for different arm parts, table 4 shows the number of weak classifiers for each type of feature extractor in each cascade. Covariance features are shown to be most important in all five of the detectors. Color features are used more for learning hand detection, than for the other cascades.

	Shoulder	Upper arm	Elbow	Lower arm	Hand
Color	3	4	10	16	22
HoG	15	29	29	36	22
Covariance	142	154	132	139	150
Total	160	187	171	191	194

Table 4 The number of used feature extractors of each type for the different arm part detectors.

6.2.4 Comparison

Figure 16 shows the performance of the arm part detectors for the different performance measures. The best feature extractor set for the NN based and the SVM based detectors were selected based on their performance in terms of pixel accuracy.

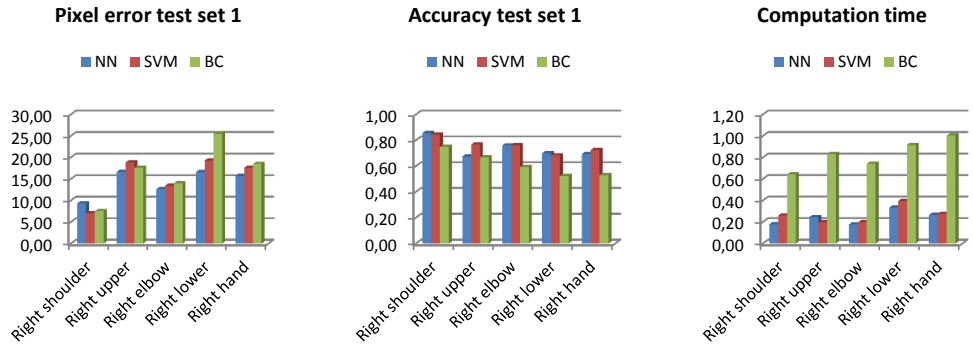

Figure 15 Pixel error, balanced accuracy and computation time for the best selected arm part detectors on the different arm parts. The best feature extractor sets were selected based on pixel distance performance on the eating test set.

In terms of pixel error, the NN based detectors perform best on four of the arm parts. Only when detecting shoulders, the SVM based detector performs best. Remarkably, in terms of balanced accuracy, the NN based detectors perform better on the shoulder and in three of the four other cases, the SVM based detector scores better than the NN based detector. This means that a well-trained classifier is not guaranteed to give good results in pixel distance, which is the measure that tells something about your practical performance. This measure also depends on the used detection algorithm, which is the exact same for all three of the classification algorithms throughout this experiment.

In terms of computation time, the only notable difference is that the cascade based detectors are much slower than both the NN based and SVM based detectors. This is quite remarkable, since one of the strengths of cascaded classifiers should be their speed. There are multiple explanations for this. Maybe the classifier fails to reject many negative examples fast enough. Maybe the weak classifiers were so weak, that such a large number of them was required to get acceptable performance. Maybe the classifier does not need to be able to use this many covariance feature extractors, as these require much computational power.

6.3 Generalization

Figure 17 shows performance results of the same arm part detectors as discussed above, but this time their performance is tested on the Daily Activities data set. All the detectors perform much

worse, but the same differences between the detectors are seen as on the second test data set as on the first test data set. The NN based detectors perform best in terms of pixel distance on all the arm parts except on the shoulder, where the SVM based detector performs best. The cascade based detectors perform much worse than the other two classification algorithms in terms of pixel distance.

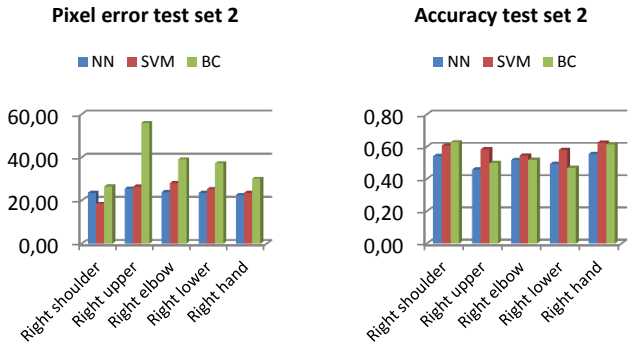

Figure 16 The performance in terms of pixel error and accuracy of the different classification algorithms on the arm parts. The best feature extractor sets were selected based on pixel distance performance on the eating test set

A notable difference in effect is seen when comparing the balanced accuracy between the different classification algorithms. On the first test data set, the cascade based detectors performed much worse than the other two kinds of detectors in terms of balanced accuracy. On the second test set however, the cascades perform about equally well (or bad) as the other two kinds of detectors.

For the sake of feature extractor comparison, the best performing feature extractor combinations for each arm part was also selected in terms of pixel distance on the second test set. The results are shown in table 5.

	Arm part	Feature extracor	n	Average Pixel error	Accuracy
NN	Right shoulder	HoG	3	17,49	0,64
	Right upper	HoG	6	21,30	0,53
	Right elbow	Greyscale, Covariance	3	22,09	0,50
	Right lower	HoG	6	19,52	0,58
	Right hand	Greyscale, Covariance, HoG	3	21,19	0,57
SVM	Right shoulder	HoG	4	15,08	0,76
	Right upper	HoG	6	22,38	0,57
	Right elbow	Covariance	3	22,88	0,50
	Right lower	Covariance	3	20,40	0,50
	Right hand	Color, HoG	4	22,05	0,60

Table 5 The feature extractor sets and performance for the best generalizing arm part detectors based on pixel distance

It seems like the best generalizing arm part detectors rely on less different kinds of feature extractors. The HoG feature extractors still seem to be the most important feature extractors in describing the arm parts. The covariance features still aid in better performance when detecting the more rotating and moving arm parts. Although, not too much analysis will be done on these results

as the performance is not even considerably good. Even though these are the best detectors in this experiment for detecting arm parts on the second data set, their performance is still quite bad.

6.4 Direction prediction

The neural networks that were trained had four additional outputs. These outputs can represent one of ten states for the search algorithm: do not look any further; look in one of eight directions (up, up-right, right, down-right, etc.); unknown where to look. The best performing neural networks for each arm part was selected and evaluated on accuracy of predicting the search direction. Note that during training, the network was selected which performed best on the stop set in terms of accuracy of the classification output. The classification accuracy was the most important performance measure, but therefore, the network was not optimized for the direction task. For each of the 10 states, the fractions of correctly positively predicted states, the true positive rates, are shown in table 6.

Target \ Prediction	Good	Up	Up-right	Right	Down-right	Down	Down-left	Left	Up-left	Dont know
Good	0.64	0.05	0.04	0.03	0.02	0.05	0.03	0.03	0.02	0.09
Up	0.25	0.23	0.10	0.01	0.01	0.03	0.04	0.04	0.11	0.19
Up-right	0.15	0.06	0.39	0.12	0.05	0.01	0.01	0.01	0.02	0.18
Right	0.22	0.01	0.10	0.31	0.20	0.03	0.01	0.01	0.00	0.10
Down-right	0.12	0.00	0.03	0.17	0.52	0.04	0.01	0.00	0.00	0.12
Down	0.30	0.02	0.02	0.04	0.15	0.30	0.07	0.02	0.00	0.09
Down-left	0.15	0.02	0.01	0.01	0.02	0.09	0.34	0.14	0.05	0.18
Left	0.20	0.03	0.01	0.00	0.00	0.02	0.08	0.29	0.18	0.18
Up-left	0.08	0.07	0.01	0.00	0.00	0.00	0.03	0.10	0.42	0.27
Dont know	0.09	0.03	0.09	0.05	0.14	0.04	0.07	0.04	0.09	0.37

Table 6 Confusion matrix showing the direction prediction performance

The detectors' classification performance is reasonable, so the prediction performance for the 'good' state is reasonably good as well. The prediction performances on all 9 other states are well above chance, so the information may be usable to support a search algorithm. However, the accuracy is insufficient to guide a search algorithm completely on the directional output of these outputs. As stated before, the trained classifiers were not optimized on the direction task, but on the classification task. Therefore, performance could be improved selecting the best trained network on MSE or accuracy of all the outputs. Also, a robust search algorithm that incorporates the information from the direction outputs, but does not completely rely on it, may improve detection time.

7 Discussion

The performance achieved in this experiment is encouraging to continue research in this direction of improving pose estimation. The detection speed and accuracy of the developed detectors, mostly from the NN based detectors, seem very promising. However, here still are quite a few questions open for further research.

7.1 The results

The results of the different experiments do neither point at one particular "winning" feature extraction technique nor a "winning" classification method. The cascade based detectors clearly turn out to be the worst performing detectors in this experiment but this could be due to method of training chosen for this experiment. The performance measures of pixel distance of the detection algorithm and balanced accuracy of the classifiers are contradicting. One would expect a detector that uses a classifier with a higher accuracy performance to perform better on localization than a detector that uses a classifier with lower accuracy performance. One can argue that it does not matter for a detector if a classifier gives a lot of false positives or negatives as long as the classifier gives the strongest response to the real detection. On the other hand, the training set was constructed in such a way, that it represents a certain problem, localization in this case. The classifier that learns the train set best should perform the task best. When setting up this experiment, I presumed that SVMs were less prone to over fitting and therefore did not need a separate stop set. I also presumed that training on less data would result in worse performance, so the NN would have a disadvantage to the SVM. These two factors may explain the difference in results between the NNs and the SVMs. The NNs perform better on the actual problem as they were trained to not overfit by using a separate stop set. Another explanation would be the fact that NNs were trained to predict the gradual distance based output. The pilot study of this experiment showed that the NNs performed significantly better when trained on continuous target outputs. The common linear SVM requires binary outputs, so this method was chosen for the SVMs. The SVMs perform better in terms of accuracy as they were trained using more train data.

Another problem in the interpretation of the results is the following. Quite a lot of the settings in both the training and the detection pipeline were set on fairly arbitrary values that seemed to work fine in the prior pilot study. This starts early in the pipeline, when cutting out the training clips. The clip size, which was set to 100×100 pixels, can be optimized for every arm part. The same goes for the learning algorithm setting: the neural network learn rate and momentum, the number of hidden units, the number of training epochs and the SVM's C parameter. The train set was put together from clips on the body parts, clips at a varying distance around the body parts and randomly located non-body part clips. This method of constructing the train set seemed to work for the NNs, but the SVMs seemed to require a wider variety of negative examples.

The methods for extracting the feature blocks were also chosen quite arbitrarily. The number of color histogram blocks was always set to 5×5. The number of HoG descriptors was $n \times n$ with n varying from 3 to 6. The number of n covariance feature blocks was varied along with the number of HoG feature blocks. Therefore, if a certain setting for n seemed to work for a combination of HoG and region covariance features, further investigation should point out whether the good performance was due to the number of HoG feature descriptors or the number of covariance feature descriptors. The HoG feature blocks were placed in a densely overlapping grid because Dalal and Triggs (18)

showed that this resulted in better performance in their experiment. The covariance feature blocks were placed with different scales on the same center to maintain their rotation invariance. Whether one of the two feature extractors results in better performance than the other, could either be due to the kind of descriptor blocks or due to the method of placing the descriptor blocks.

The covariance feature descriptor can also be used in different ways. For this experiment, the region covariance descriptor describes the covariance between the set of fundamental features, position, color, and first and second order derivative. These covariance values may not all be important in detecting all different arm parts. It could be interesting to analyze the weights of a NN or SVM trained on these covariance features, to see which values contain most information. The speed of such a detector could be improved by eliminating near-zero-weighted covariance values and not compute these in the feature extractor.

8 Conclusion

8.1 Classification algorithms & feature extractors

In this study, three classification algorithms were compared for detecting arm parts: neural networks, support vector machines and GentleBoost cascades. The neural networks turned out to be the most accurate, the fastest and the best generalizing of the three, when localizing the arm parts. On the other hand, the SVMs seem to be the most the most accurate when simply looking at the classification performance. Both the quantitative performance results and the visual activation patterns show that the neural network based detectors perform best and generalize best to unseen data. However, there were some differences in training methodology. Using the same (or the optimal) training methodology for all three of the classification algorithms, may lead to different results.

For the feature extractors, there is an even less clear cut answer. One would expect that for the simple shaped, little-movement arm parts like the shoulders and upper arms, the densely overlapping HoG descriptors would perform best and that for the more movable and shape-changing parts like the elbows, lower arms and hands, the rotation invariant covariance features would perform best. The results do not really confirm this expectation. HoG features are indeed preferred for detecting the shoulders and upper arms, but both the neural networks and the support vector machines also seem to perform better on HoG features than on covariance features when detecting hands, while hands are the most complex shaped arm parts of all five evaluated arm parts. Further investigation should be done in the different ways of spatial distribution of the descriptor blocks over the evaluated image clips for both the HoG features and the covariance features. The pixel intensity features were useful only in combination with either one of the before mentioned feature extractors. They did seem to add quite some locational information about the objects in the image. The color features, as expected, only added useful information when detecting hands.

8.2 Localization improvement

The initial question behind this study was whether local arm part detectors could be used in order to improve a given pose estimation. This experiment was set up with Vincent van Megen's pose estimator (2) in mind and the arm part detectors were trained and tested using the data that was also used for training and testing his system. His system used one neural network to predict the locations of seven body parts: head, ahoulders, elbows and hands. As input, optical flow and greyscale information was used. His system also used a dedicated hand tracker to improve accuracy performance for the hands.

In table 7, the pixel distance performance of Van Megen's system is compared to the pixel distance performance of the local detectors developed during this study. For the shoulder this turned out to be one of the SVM based detectors and for the hand and the elbow detectors NN based detectors turned out to perform best. As the table shows, the accuracy on localization of the hands can be increased significantly by using the NN based hand detector. Elbow localization and shoulder localization is not increased at all. This may be due to noisy annotation of the data and the fact that the local detectors cannot use contextual information like the positions of other body parts. As stated before, for applications in action recognition and HCI, hand tracking is most important as most movement is done by the hands, so improvement on hand detection is a good achievement for this experiment.

	Set	X error	Y error
Hand	Pose estimator	15.7	14.4
	Hand tracker	13.2	13.4
	Detector (NN)	9.4	10.6
Elbow	Pose estimator	7.1	3.1
	Detector (NN)	6.8	9.5
Shoulder	Pose estimator	5.3	3.9
	Detector (SVM)	4.8	4.8

Table 7 Performance of the local arm part detectors compared to the performance of Van Megen's pose estimator (2).

8.3 Future research

8.3.1 Setting and parameters optimization

As stated in the discussion, a lot of settings and parameters were quite arbitrarily chosen for this experiment. A next step in developing local arm part detectors would be to further investigate all the optimal settings and parameters. Every component in the training and evaluation pipeline can be optimized and have effects on the other components. An interesting research would be to take all the settings from the complete data-set-to-detector pipeline and optimize it using some kind of genetic algorithm. As this is probably too complex to begin with, optimizing the feature extraction settings using a similar approach could lead to some interesting results. The feature descriptors, the number of descriptor blocks, the degree of overlap between the descriptor blocks and the spatial organization of the descriptor blocks all have a great effect on the performance. Applying the same method for finding optimal detector settings to a more challenging data set is also a very interesting research topic.

8.3.2 Detection algorithm improvement

For fast object detection, it is important to efficiently search for the object. There are many possible improvements on the sliding window detection algorithm. An efficient detection algorithm is even more important when using computationally expensive feature descriptors like the region covariance features or descriptor blocks in a dense grid like the HoG descriptors. It would be very interesting to use the information from the additional neural network outputs to guide a detection algorithm in the right direction.

8.3.3 Motion features

When tracking body parts in videos, there are various ways to use the extra information provided by frame sequences to improve the performance of the body part detectors. The estimated location of a body part in frame t can be used as the prior estimated location of the body part in frame $t + 1$. Multiple passed frames and estimated locations can be used to determine whether the body part is moving and in which direction it is moving. In videos it is also possible to extract motion features like optical flow (2) from the image clips and train the body part detectors using these feature descriptors, combined with other feature descriptors.

8.3.4 Part-based pose estimation

Bottom-up pose estimation and pose tracking could be achieved by using the estimated locations of the individual body parts. A skeleton can easily be fitted through the estimated body part locations in order to get a pose estimation. Constraints on a body model can even support in localizing the individual body parts. Information about possible and impossible locations of the body parts can be

propagated from the pose estimator back to the body part detectors. This way, low-level mistakes by the part detectors can be compensated for by the high-level pose estimator.

9 References

1. **Sung, K.** *Recent Videogame Console Technologies.* Bothell, Washington, US : sn, February 2011.

2. **Megen, V. Van.** 3D pose tracking using optical flow. Nijmegen : sn, 2010.

3. *A survey of computer vision-based human motion capture.* **Moeslund, T.B. en Granum, E.** sl : Computer Vision and Image Understanding, 2001, Vol. 81.

4. *A survey of advances in vision-base human motion capture and analysis.* **Moeslund, T.B., Hilton, A. en Krüger, V.** sl : Computer Vision and Image Understanding, 2006, Vol. 104.

5. *The visual analysis of human movement: a survey.* **Gavrilla, D.M.** sl : Computer Vision and Image Understanding, 1999, Vol. 73.

6. *3D human pose estimation using 2D-data and an alternative phase space representation.* **Moeslund, T.B. en Granum, E.** sl : Workshop on human modeling, analysis and synthesis at CVPR, 2000.

7. *Hierarchical part-based human body pose estimation.* **Navaratnam, R., et al.** sl : British Machine Vision Conference, 2005.

8. *Human pose estimation using learnt probabilistic region similarities and partial configurations.* **Roberts, T.J., McKenna, S.J. en Ricketts, I.W.** sl : European Conference on Computer Vision, 2004.

9. *Pose estimation and tracking of eating persons in real-life settings.* **Zhang, L., et al.** Veldhoven : ICT-Open, 2011.

10. **Eichner, M., et al.** *2D articulated human pose estimation and retrieval in (almost) unconstrained still images.* sl : Image Rochester, 2010.

11. *A trainable system for object detection.* **Papageorgiou, C. en Poggio, T.** sl : International Journal of Computer Vision, 2000.

12. *Face recognition: a literature survey.* **Zhao, W., et al.** sl : ACM Computing Surveys, 2003, Vol. 35.

13. *Neural network-based face detection.* **Rowley, H.A., Baluja, S. en Kanade, T.** sl : PAMI, 1998.

14. *Rapid object detection using a boosted cascade of simple features.* **Viola, P. en Jones, M.** sl : Computer Vision and Pattern Recognition, 2001.

15. *View-independent recognition of hand postures.* **Wu, Y. en Huang, T.S.** sl : Computer Vision and Pattern Recognition, 2000.

16. *Long term arm and hand tracking for continuous sign language tv broadcasts.* **Buehler, P., et al.** sl : British Machine Vision Conferenece, 2008.

17. *Hand pose estimation using hierarchical detection.* **Stenger, B., et al.** sl : International Workshop on Human-Computer Interaction, 2004.

18. *Histograms of oriented gradients for human detection.* **Dalal, N. en Triggs, B.** sl : Computer Vision and Pattern Recognition, 2005.

19. *Object detection with discriminatively trained part based models.* **Felzenzwalb, P.F., et al.** sl : Pattern Analysis and Machine Intelligence, 2010.

20. *Multiple kernels for object detection.* **Vedaldi, A., et al.** sl : International Conference on Computer Vision, 2009.

21. *Region covariance: A fast descriptor for detection and classification.* **Tuzel, O., Porikli, F. en Meer, P.** sl : European Conference on Computer Vision, 2006.

22. *Fast human detection from videos using covariance features.* **Yao, J. en Odobez, J-M.** sl : European Conference on Computer Vision Visual Surveillance, 2008.

23. *Image processing with neural networks – a review.* **Egmont-Petersen, M., Ridder, D. de en Handels, H.** sl : Pattern Recognition, 2002, Vol. 35.

24. **Bradski, G. en Kaehler, A.** *Learning OpenCV.* sl : O'Reilly Media Inc, 2008.

25. *The boosting approach to machine learning – an overview.* **Schapire, R.E.** sl : Workshop on Nonlinear Estimation and Classification, 2001.

26. *A decision-theoretic generalization of on-line learning and an application to boosting.* **Freund, Y. en Schapire, R.E.** sl : Computational Learning Theory: Eurocolt, 1995.

27. *Additive logistic regression: a statistical view of boosting.* **Friedman, J., Hastie, T. en Tibshirani, R.** sl : The Annuals of Statistics, 2000.

28. *Activity recognition using the velocity histories of tracked keypoints.* **Messing, R., Pal, C. en Kautz, H.** sl : International Conference on Computer Vision, 2009.