Orientation Estimation

from ceiling-mounted cameras

Master's Thesis in Artificial Intelligence – Intelligent Systems

August 12, 2011

Author:
Silvia-Laura Pintea
S.L.Pintea@student.uva.nl
Student no. 6109969

Supervisor:
Dr. Gwenn Englebienne
University of Amsterdam FNWI
Science Park 904, 1098XH Amsterdam

UNIVERSITY OF AMSTERDAM

Abstract

This project addresses the orientation estimation of people from a ceiling-mounted camera. Ceiling-mounted cameras have two favorable properties: people do not occlude each other as much as when the camera is positioned at human-level and the segmentation of people might be easier in crowded scenarios. On the other hand, they also entail two less favorable characteristics: occlusion by the background can substantially affect the position estimation and the same person, with the same orientation but situated at different positions with respect to the camera will have vastly different appearances due to the projection in the image plane. Unfortunately, the last issue has a dramatical influence on the orientation estimation task.

In this work, we evaluate a number of supervised learning techniques and data representation methods for orientation estimation. We investigate how these generalize over different people and different locations, and propose a novel use for *Gaussian processes* for this task.

Acknowledgments

I would like to thank my supervisor, dr. Gwenn Englebienne, for his guidance and patience. I greatly appreciate the amount of time he has spent helping me with ideas and, at times, debugging my code.
I would also like to thank all my colleagues and friends with whom I have discussed my project on multiple occasions and from whom I have received valuable suggestions (some of which will be encountered in the data processing part).
Finally, I would like to thank everybody that was there for the recording of the second dataset.

The source code of this project can be found at:
https://github.com/SilviaLauraPintea/OrientationEstimation

Contents

Chapter I

Introduction

I.1 Orientation-Estimation Problem

In the past few years, the task of people tracking and human-body pose detection has been intensely studied and different approaches have been proposed and tested. Until now, the main focus has been on using cameras mounted at human level. Although the majority of the cameras used in surveillance or other real-life situations are offering an elevated sideway view, few articles have been published in which such elevated cameras are used.

The ability of estimating the orientation of people in a video has a large number of benefits. Knowing the direction in which a person is looking can be used for the analysis of human social interaction. It can also be used as a component of the security systems present in airports and other public places. Targeted advertisement is another field that can benefit out of the ability of estimating the orientation of people.

I.2 Goal of This Thesis

The goal of this thesis is to be able to build a system that can estimate the orientation of the people present in a video recorded with an *elevated sideway camera*. This project tries to predict the direction in which a person is looking. The facing direction does not always coincide with the body orientation and when they are different, being able to estimate the former represents a challenge of this project.

To estimate the direction in which a person is looking, we first need to be able to detect the people present in an image/video. Therefore, the first requirement in estimating orientations is the availability of a *people-detection* system. An important aspect to be considered is that the people detection is bound to give imperfect results, a fact that will affect to a large degree the quality of the extracted features and the overall performance.

The positioning of the camera gives rise to variance in the data corresponding to the same person and the same orientation. This fact represents another challenge in fulfilling the goal of this thesis.

I.3 Thesis Overview

Chapter II gives an overview of the available methods for pose and orientation estimation. In chapter III the people-detection system is briefly described and an overview is given of the

different data pre-processing steps that are explored in this work. Chapter IV is composed out of 3 sections: section IV.1 indicates the features extracted from the data, section IV.2 gives a brief description for each learner that was implemented and applied and section IV.3 presents comparative experimental results for different features, learners and datasets. Finally chapter V gives the conclusions and the future work.

Chapter II

Related Work

This chapter provides an overview of the state-of-the-art techniques for *orientation estimation*. Given that more research has been done on estimating body poses rather than the facing direction of people, and that most of the features used for pose estimation are also useful in the task of orientation estimation, articles dealing with either of these problems have been surveyed.

The following two sections outline some of the existing methods used for solving the orientation/pose detection problem. The articles are grouped in two categories: *example-based* approaches and *learning-based* approaches, which in turn are divided into *orientation prediction* versus *orientation tracking* and for the *orientation prediction* we distinguish between *classification methods* and *regression methods*.

II.1 Example-Based Approaches

The goal of *example-based* approaches is to be able to match a stored model against an input image. The labels of the models stored in the database are used for assigning a label to the input image.

In [Rogez and Orrite, 2007] the body pose is detected by matching the body shape and skeleton against the stored models. The orientation is defined by two angles: the latitude φ and the longitude θ which are discretized into a set of 8 *viewing hemispheres* for each angle. For each viewing point a set of *view-based 2D models* containing shape and skeleton are generated. For a new input image, the view-point is determined and the image is projected on the corresponding *training plane*. The models corresponding to this view are, then, used for the feature-extraction step. The projection on the *training plane* is done in two steps: first the input image is projected on the vertical plane and then the result is projected on the plane corresponding to the detected view-hemisphere (for which there are *view-based models* available in the training database). In this case the position of the camera is not important because all images are first transformed to a *frontal view* — the vertical plane is centered on the detected human bodies — and then projected to a certain training plane. Once the image is warped such that it would be aligned with the models present in its corresponding set of view-based models, the shape is retrieved by computing the *Canny edges* [Canny, 1986] and then using *dynamic programming* to keep only the boundary edges in the form of a smooth contour. The next step is retrieving the corresponding skeleton and then projecting back the detected shape and skeleton on the appropriate position in the image. It is rather difficult to acquire a complete

database of shapes and skeletons of poses and this fact makes this method less appealing. On the other hand, the idea of projecting both the training images and the test images on a common plane — that accounts for a form of image normalization — and then extract the features is innovative.

While estimating poses, the ability of determining the coordinates of all important body features could be extremely useful. In [Chen and Wang, 2009] the authors present a method for estimating these coordinates by relying on human body statistics. In their experiments 2 infrared cameras (one top-view and one side-view camera) are used and the silhouettes are extracted. The obtained silhouettes are processed with *dilation* and *erosion* to smooth their shape. After this step the boundary pixels are retrieved using *chain coding*. The *centroid* (geometrical center detected at the intersection of all boundary pixels) is determined and the distance from each boundary pixel to the centroid is computed as a *1D* discrete function. All the maxima and minima of this function are retained. The authors assert that the head and the feet points are always in the maximal set. The feet and head are the closest maximal points to the axis that goes through the *principal point* and the *centroid point*. The *principal axis* — the axis that pases through the coordinates of the camera and its orientation corresponds to the orientation of the camera — meets the image plane at the *principal point*. Another information given by the authors is that the feet are always closest to the principal point and the head is always the farthest away. The hand detection is more difficult because the position of the hands is unrestricted so for this task the authors use the position of the neck which can be approximated using some generally valid human body proportions. Because the target of the authors is to determine the *3D* coordinates of all important body features: position of the hands, head, neck, feet, elbows and knees they use the constraint that the person is standing. The *3D* coordinates of the hands are determined employing *Gaussian-Seidel iteration* [Barrett et al., 1994], the knees' positions are determined with the *gradient descent* algorithm. The rotation of the elbows is formulated with the use of *quaternions* and the rotated elbow point is mapped in the image and an error function is minimized to determine the best fit. Unfortunately, the statistics indicated for predicting the locations of the important points are only valid for the case in which the person is standing and there is an available sideway camera.

There are numerous papers in which the input image is matched against the stored images using different techniques such as: [Jiang, 2009] in which the foreground area is covered by a number of polygons such that each polygon best matches the underlying body part, [Dimitrijevic et al., 2006] that present a method based on fitting *2D* silhouettes of walking people over the input image by using the *Chamfer distance*, [Navaratnam et al., 2005] which model the body as a collection of parts that encode *3D* joint angles and that are represented as an *hidden Markov model (HMM)* [Cappé et al., 2005].

II.2 Learning-Based Approaches

The *learning based* approaches try to learn some parameters that are specific to each pose/ orientation. These methods are usually relying on *2D* (sometimes *3D*) features that are extracted from the images and then used for training. Examples of such features are: *edges* and *contours*, *Haar responses*, *optical flow*, *SIFT* features, etc.

II.2.1 Orientation Prediction

For predicting the orientation/pose of a person in an image two approaches are available: *regression methods* and *classification methods*.

Cases in which the aim is to assign each input vector to one of a finite number of discrete categories, are called classification problems. If the desired output consists of one or more continuous variables, then the task is called regression. [Bishop, 2006]

Given that the orientation problem requires the ability of predicting a continuous variable in the interval *[0, 360)*, for this task it would be preferable to employ *regression* and not *classification*. Classification is nonetheless more common, and we start, therefore, with an overview of these methods.

II.2.1.1 Classification Methods

In [Brekhof, 2005] *neural networks (NN)* [Bishop, 1995] and *local receptive fields* [Bishop, 1995] are employed for detecting the *body-facing direction* of people in videos taken with an elevated sideway camera. In this paper the people detection component is represented by the *Chamfer system*, which is based on detecting the upper part of the body (head and shoulders) of the individuals present in the image. The features considered for detecting the *body facing direction* are: *shoulder width*, *head location relative to the body*, *facial skin and hair contrast*. In the experiments done in this paper only the body orientation is considered; thus, the images in which the head orientation does not coincide with the body orientation are ignored. The possible orientations were discretized to only four: *front*, *back*, *left* and *right*. For each possible combination of two orientations an *NN* was trained giving rise to 6 classifiers. The best performance was obtained by the *front versus back* classifier, using the *skin/hair ratio* as a feature. This approach was extended by labeling the shape models employed for detecting people such that these detection labels would indicate which one out of the two classifiers mentioned above: *front versus back* or *left versus right* should be applied at the next step for the current input image. Although this method uses a single elevated sideway camera, the downside of it is that the orientations are restricted to only the four ones mentioned above. Another drawback of this approach is that it only detects the body orientation and not the head orientation and it does not consider the cases in which these two do not coincide. The choices for the features were constrained by the circumstance that the people-detection system was based on upper-body shapes.

The method described in [Van den Bergh and Van Gool, 2009] employs *2D* and *3D Haarlets* [Haar, 1910] for pose estimation. Here, multiple cameras are used (two side-cameras and a top-camera), that are synchronously capturing frames of the same person. The pixels corresponding to the body of the person present in the image are obtained by doing background subtraction. The resulting silhouettes are then normalized by retaining only a fixed-size bounding box around the person and centering the silhouettes horizontally to their center of gravity. For the *2D* case the obtained frames from the multiple cameras available are concatenated and the result is projected onto a lower dimension using *average neighborhood margin maximization (ANMM)* [Wang and Zhang, 2007] — this method has the advantage that the projection is done such that *neighboring points with the same class label as the current data point are pulled towards it as near as possible, while simultaneously pushing the neighboring points with different labels away from the current point as far as possible* [Van den Bergh and Van Gool, 2009]. In the end an *NN* is used to determine the closest stored match in the database, and this match represent the

output of the system. *Haar wavelets* representing bars with different sizes and orientations are convolved with the underlying pixels at different locations in the input image. Each *Haarlet* is then convolved with the vectors in $\boldsymbol{W}_{opt}$ — the optimal projection. The set of *Haarlets* whose linear combination best approximates $\boldsymbol{W}_{opt}$ (they have the highest response magnitudes) are kept. The coefficients that would result from the *ANMM* transformation are approximated by the determined coefficients used to combine the *Haarlets*. The result for each new silhouette is compared with the mean of the coefficients stored for each class. Rotation invariance is obtained by splitting the training data into 36 groups (depending on their corresponding orientation) and then training 36 different classifiers — one for each group of orientations. This approach requires the use of multiple cameras even for the *2D* case. The performance is greatly enhanced by the use of 3 or more cameras (out of which only one offers a top-view, the rest being positioned at the human-level).

Haar wavelets are also used in [Shimizu and Poggio, 2004] for estimating the walking direction of people in images recorded with a side camera. The *wavelet coefficients* are determined using three orientations: horizontal, vertical and diagonal and multiple scales. A number of 16 classifiers are trained for orientations in the range *[0, 360)*. *Support vector machine (SVM)* [Cristianini and Shawe-Taylor, 2000] is used for training the 16 classifiers. The system chooses the most probable orientation based on a decision function obtained from the predictions of these classifiers. The decision function includes the outputs of the current classifier, whose index is i, and the outputs of its two neighbors $i-1$ and $i+1$.

Because the facing direction is mainly represented by the head direction, the previous work done on this task is also relevant for this project. An accurate method of detecting head orientations is presented in [Wu and Toyama, 2000]. The pre-processing step of the data requires the extraction of the head pixels, followed by enhancing the contrast, cropping the head area, rescaling the resulting image to a fixed size and applying a circular mask such that the pixels that do not correspond to the head area are ignored. Finally, *histogram equalization* is applied to the masked image. In the next step, a *Gaussian* at a coarse scale, and 4 rotation-invariant *Gabor wavelets* [Daugman, 1985] are convolved with the resulting image. In each training image, for each pixel location $\boldsymbol{p}_j$, there exists a normalized feature vector $\boldsymbol{z}_j$ corresponding to the concatenated responses at that location. Each vector $\boldsymbol{z}_j$ (for each pixel location $\boldsymbol{p}_j$) contributes to a corresponding *model-point* i_j*. The *pdf* of each such model-point is defined by a *Gaussian* whose mean and covariance coincide with the mean and covariance of the vectors associated with it. For head-pose detection the *maximum a posteriori (MAP)* pose given the observation is estimated. The method is able to predict somewhat accurately the head pose. Unfortunately, it requires images in which the head area can be clearly segmented.

In [Enzweiler and Gavrila, 2010] a unified method for pedestrian detection and orientation estimation is proposed. This is done by decomposing the probability that a sample $\boldsymbol{x}_i$ is pedestrian — w_0 — over a set of K clusters: $\mathcal{P}(w_0 \mid \boldsymbol{x}_i) = \sum_{k \in K} \mathcal{P}(\psi_k \mid \boldsymbol{x}_i)\mathcal{P}(w_0 \mid \psi_k, \boldsymbol{x}_i)$. The cluster membership prior for $\boldsymbol{x}_i$, $\mathcal{P}(\psi_k \mid \boldsymbol{x}_i)$, is modeled using the *Bayesian* approach applied to the residual shape distance between the best matching shape in cluster ψ_k and the input sample $\boldsymbol{x}_i$. The cluster-specific probability that $\boldsymbol{x}_i$ is a pedestrian, $\mathcal{P}(w_0 \mid \psi_k, \boldsymbol{x}_i)$, is modeled using k texture-based classifiers. The orientation is estimated by using a *Gaussian mixture model* such that each cluster, k, has a *Gaussian* component associated with it. For determining the cluster-specific probabilities that the sample $\boldsymbol{x}_i$ is a pedestrian two methods were employed: *histograms of orientated gradients (HoG)* [Dalal and Triggs, 2005] were extracted and *SVM* was used for classification and in the second method *adaptive receptive field* features were used in combination with a *multi-layer neural network*.

II.2.1.2 Regression Methods

The use of *principal component analysis (PCA)* [Pearson, 1901] combined with *support vector regression (SVR)* [Smola and Schölkopf, 2003] is proposed in [Ando et al., 2007] as a method of estimating human poses. The training images represent different instances of a *3D* object (that resembles to some degree a human head) positioned at different orientations in the range $[0, 180]$. The *eigenspace* is derived from the training data which is subsequently projected on this space. In the next step, *SVR* is used to learn the pose function from the training data. To make their system more robust to changes in the illumination the authors have decided to extract features using the *modified gradient method* [Kleinman and van den Berg, 1992] instead of using just the image brightness pattern. Another extension was to split the input image into areas and extract features separately from each such area. A *reliability* measure is calculated based on how similar each area is to the training patterns. The weighted median value of the *reliability* is used to decide for the estimated pose. The performance of the system is estimated on images recorded with a side camera and the face area is detected before the features are extracted.

The *Gaussian process* [Rasmussen and Williams, 2006], combined with *scale-invariant feature transform (SIFT)* [Lowe, 1999] descriptors, is used in [Zhao et al., 2008] for pose estimation. In their experiments the data used for training and testing is recorded with a side camera. *Harris corner detector* [Harris and Stephens, 1988] is employed for locating interest points. A *SIFT* descriptor, $\boldsymbol{d}$, is extracted at each interest point location. The location of the interest point is concatenated with the descriptor, giving rise to the feature: $(x, y, \boldsymbol{d})$. The *bag-of-words* method is used for extracting the final features to be used for training. The calculated *codebook* is not larger than 60 words. For the learning part the authors use a *Gaussian process* with a *squared exponential* kernel function. In this approach the features were extracted from the complete body area.

II.2.2 Orientation Tracking

A more complex method is presented in [Ozturk and Aizawa, 2009] for tracking both the body and the head orientation. A *two level cascaded particle filter tracker* [Yang et al., 2005] is used for people detection in a video. The tracker can not deal with occlusion so this case was excluded also from the task of orientation tracking. The features that were defined for determining the orientation were: *motion vector*, *SIFT optical flow (SOFV)* and the *body orientation*. The *body orientation* is determined by dividing the data into 2 or 4 main classes. If the camera is positioned sideways then only 2 classes are defined: left and right; otherwise, if the camera is positioned centrally on the ceiling then the other 2 classes are added. For the case in which the camera is positioned sideways the data corresponding to the 2 classes is grouped in 5 categories: $\frac{-\pi}{4}$, $\frac{\pi}{4}$, $\frac{-\pi}{8}$, $\frac{\pi}{8}$ and 0. These categories do not represent actual body orientations, but classes of body shapes. *Canny edge detector* is used to retrieve the shape of the upper body (head and shoulders) and *shape context matching* [Belongie et al., 2002] is used to find the corresponding group for each shape. To compute the *motion vector* (or *displacement vector*) the *particle filter* results are employed. This feature gives information regarding the general direction on which the person is moving. The *displacement vector* and the *SOFV* are computed only for every 10^{th} frame. For the evaluation of *SIFT optical flow* the *SIFT* features are computed around the head region for every 3^{rd}, 6^{th} and 10^{th} frame. The *SIFT* features detected at each such interval are matched and the *optical flow* is estimated. The head region is then divided

into four parts marked as: *-R, -L, R* and *L*. For each such region the average vector of the optical flow vectors is determined (the vectors are called: $\overrightarrow{V}_{-R}$, $\overrightarrow{V}_{-L}$, $\overrightarrow{V}_{R}$ respectively $\overrightarrow{V}_{L}$). The average position of the *optical flow* vectors is also computed in each region, noted as: c_{-R}, c_{-L}, c_R respectively c_L. Out of the four vectors the two corresponding to the largest two lengths are kept: $\overrightarrow{V}_1$ and $\overrightarrow{V}_2$. Finally, *SOFV* is calculated by comparing the two average positions corresponding to the two vectors having the largest lengths (c_1 and c_2) and adding these two vectors: $SOFV = \overrightarrow{V}_1 + \overrightarrow{V}_2$. Four simple rules are defined to decide about the orientation changes from one frame to another. The paper as it is presented tries to achieve a method of accurate orientation tracking and not of orientation estimation. Also given that the *SIFT* features and the *displacement vector* are determined every 10^{th} frame, it might not be fast enough to be applied in situations in which real-time estimates are required.

Chapter III

Data Processing & People Detection

Estimating the orientation of the people in a video requires, in the first place, an accurate method for detecting people locations and extracting the foreground area corresponding to each person. The system described in [Englebienne and Kröse, 2010] is applied for this task. The reason for which this people-detection system was chosen is that it has the advantage of providing information about the location of the head and the feet of a person as a side-effect. And this information is employed to a large extend in the feature extraction part. Other advantages of this system are its robustness to changes in illumination and the fact that it does not require sequential data. However, because the detected foreground areas are not always as flawless as desired, different strategies have been developed to improve the quality of the detections.

At the same time, this brought up the research question of which part(s) of the body gives more information about the orientation. Given that the data pre-processing steps have an essential role in the final performance of the system, a number of changes have been applied to the image-data which are described in more details in section III.3.

III.1 People Detection System

A *background model* is built out of a set of images in which there is no person present. This is done by applying *PCA* and retaining the top *3* components. Each input image is, then, converted to a predefined *colorspace*, projected and backprojected on this *eigenspace*. The probability of an image pixel being background is given by a *1*-dimensional *Gaussian* with the mean equal to the mean of the backprojected image and a constant variance: $p_b(\boldsymbol{x}_i) = \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_i, \sigma^2)$. The foreground probability is defined to be uniform over the possible pixel values: $p_f(\boldsymbol{x}_i) = \frac{1}{256}$.

Image templates are generated by projecting $3\mathcal{D}$ bounding boxes in the image plane (figure III.1(b)). Each such box is constructed around a ground location by considering the average human height and a certain width. A mask, $\boldsymbol{m}$, can be associated with each such template to indicate which pixels belong to the background and which to the foreground:

$$p(\boldsymbol{x}|\boldsymbol{m}) = \boldsymbol{m}p_f(\boldsymbol{x}) + (1 - \boldsymbol{m})p_b(\boldsymbol{x}) \tag{III.1}$$

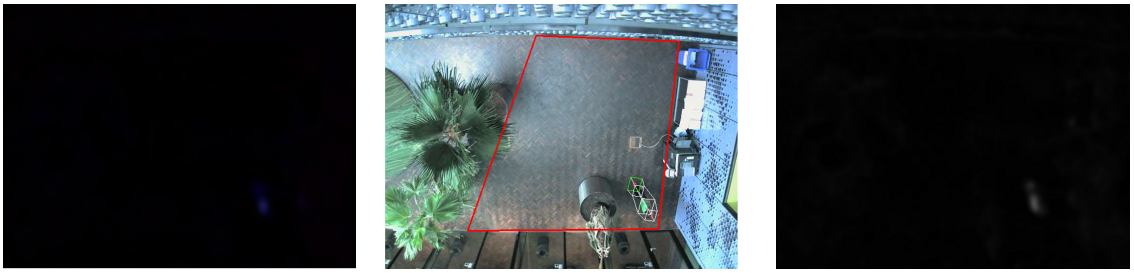
where $\mathbf{x}$ is a vector of pixels contained in the template and $\mathbf{1}$ and $\mathbf{m}$ are also vectors of the same size. Given that the templates depend only on the position of the person in the image (feet location), a set of masks, $\boldsymbol{m}_{\mathcal{L}}$, can be associated with a set of locations in the image:

$\mathcal{L} = \{\boldsymbol{l}_1, \boldsymbol{l}_2, ..\}$.The set of masks is built such that if two or more masks overlap, the pixels are present only once in the masks' set.

The probability of a number of people being visible at locations $\mathcal{L}$ in the image is given by:

$$p(\boldsymbol{x}, \mathcal{L}) = p(\mathcal{L})(\boldsymbol{m}_{\mathcal{L}} p_f(\boldsymbol{x}) + (1 - \boldsymbol{m}_{\mathcal{L}}) p_b(\boldsymbol{x})) \quad \text{where} \quad p(\mathcal{L}) = p(\mid \mathcal{L} \mid) \prod_i p(\boldsymbol{l}_i) p(\boldsymbol{l}_i \mid \boldsymbol{l}_1..\boldsymbol{l}_{i-1}) \tag{III.2}$$

In the formulas depicted in (III.2) $p(\mid \mathcal{L} \mid)$ represents the prior probability that a number of $\mid \mathcal{L} \mid$ people should be seen in a video; $p(\boldsymbol{l}_i)$ represents the prior probability that a person should be positioned at location $\boldsymbol{l}_i$ — for this a polygon is defined that encloses the valid locations of the people in the image; and $p(\boldsymbol{l}_i \mid \boldsymbol{l}_1..\boldsymbol{l}_{i-1})$ indicates the probability that a person should be detected at location $\boldsymbol{l}_i$ given that there are people at locations $\boldsymbol{l}_1, .., \boldsymbol{l}_{i-1}$ — it is set to *0* if the distance between the locations is smaller than half of the width of the template, otherwise is uniformly distributed.



(a) Background model (*Luv colorspace*) (b) Template centered on a person (c) Background model (*Gabor responses*)

Figure III.1: People detection

The probability of a set of person locations $\mathcal{L}$ is defined as:

$$p(\mathcal{L} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x}, \mathcal{L})}{\sum_{\mathcal{L}} p(\boldsymbol{x}, \mathcal{L})} \tag{III.3}$$

The method finds the most likely position of the first person and then it searches for other positions that improve the likelihood. No more positions are added when, by adding the most likely new position, the likelihood decreases. *The most likely next position is detected by considering only positions that did improve the likelihood in isolation* [Englebienne and Kröse, 2010].

For the data used in this project, the background model is obtained by convolving the input image with the *3 Gabor wavelets* displayed in figure III.2. Although — as it can be

Figure III.2: The used *Gabor* wavelets

observed in figure III.1 — the background model obtained by convolving the image with the

3 Gabor wavelets is more noisy than the one using the *Luv* colorspace, it offers an easier way of selecting the foreground areas by thresholding the background model. The image is blurred with a *median kernel* of size 5×5 pixels, convolved with each one of the *3 Gabor wavelets* — with the orientations: $\frac{\pi}{6}$, $\frac{\pi}{2}$, $\frac{5\pi}{6}$ — and the results are copied on the 3 channels of the input image. A *median filter* applied to an unfiltered window has the same values as the window on all positions except for the center of the window where the value from the unfiltered window is replaced with the median of all the other values in the window.

A few processing steps, that have proven to be efficient, are applied to the background model. After the background model is computed, it is smoothed by convolving it with a *Gaussian kernel* of the size 31×31. The result is subjected to a process of *erosion* followed by *dilation* that is repeated a number of times to have the area corresponding to each person resemble a uniform blob.

III.1.1 Projecting and backprojecting templates

The performance of the system depends to a large extent on the quality of the detections — how well the templates fit the people present in a video. Therefore, special care has to be taken in finding reliable camera calibration models.

Initially, the model employed was ignoring the 3 camera angles — pitch, yaw and roll — and was assuming that the *xy* image plane was parallel to the *xy* camera plane. This simplified model was not able to give satisfying results, thus, the generated templates could not be correctly centered on the detected people. The solution to this problem is to use a more complex model that includes the camera angles: *the perspective projection* model. This calibration model is explained in more details in appendix A.

As it can be observed in figure III.1(b), the *people-detection system* relies on the construction of templates around the people present in the image. The position in the image plane at which the template needs to be built is *backprojected* in the real world using the available camera calibration. A *3D* template is created around the backprojected location by considering the average human height and a given width. Finally, the resulting *3D* templates are projected back in the image. The templates are parametrized only by the location in the image plane — feet location, the average human height and a certain width.

Unlike in the previous case when the camera *xy*-plane and the image plane were parallel and the *3D* coordinates of the feet location could be easily retrieved, in the new situation obtaining the backprojected point is more difficult. For this circumstance, we have to apply the inverses of the projection transformations and impose the constraint that the backprojected point should be positioned in the ground plane — its computed *z*-component should be equal to 0. The explanations regarding how we can accomplish this are given in appendix A, in section A.2.

III.2 Annotations

A system for annotating the data was built such that it would allow for the annotation of both the *latitudinal* and *longitudinal* orientation angles for each person present in the data. The *latitudinal* and *longitudinal* angles are indicated in the image with arrows. The *longitudinal* angle can have values in the range *[0, 360)* while the *latitudinal* angle can be defined in the range *[0, 180)*. For the *latitudinal* angle the image is rotated relative to the camera position such that the angle can be annotated easier. Figure III.3 shows how the annotations component looks

like for an example image. A fact worth mentioning is that the annotations are not perfect because some of the images have a lower quality and in those situations guessing the facing direction is a more difficult task.

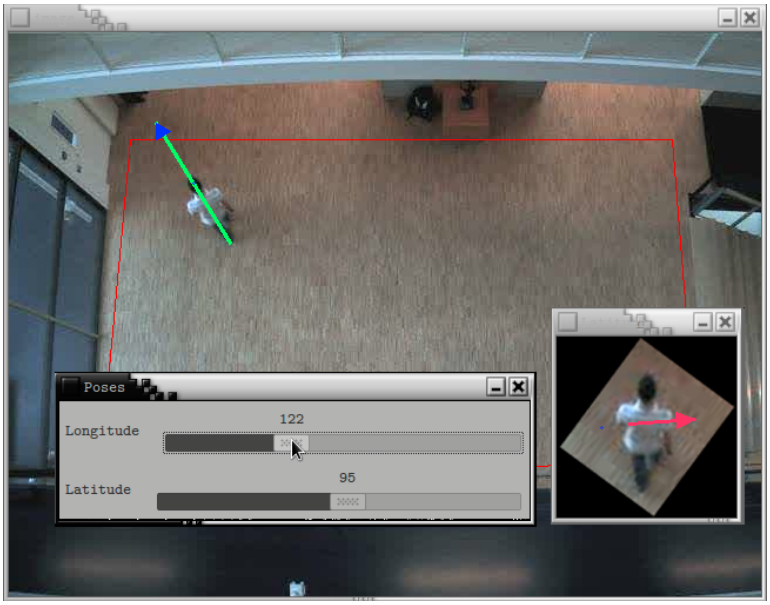


Figure III.3: Annotations on an example image

The idea of annotating *latitudinal angles* was inspired by [Rogez and Orrite, 2007]. Unlike in this paper, the goal here is not to be able to actually predict the latitudinal angles, but to employ them in the learning of the longitudinal angles. This approach can be seen as a form of *transfer learning* and it indicates to the learner that not only the *longitudinal* angle is responsible for how diverse the features can be for the same target orientation, but also the *latitudinal* angle. Experimental results are presented in subsection IV.3.3 where a *neural network* is trained to predict both *longitudinal* and *latitudinal* angles, while having the performance estimated only on the predicted *longitudinal* orientations.

III.3 Data pre-processing

In this section, we present in detail the pre-processing steps that are applied to the extracted foreground areas to give more reliable foreground segmentations and simplify the task of the learners.

Once a good background model is available a threshold is defined and the foreground area, presumably, corresponding to each person is retrieved. Some foreground pixels might not be contained in any template (because the person might be larger than its template or the template is not correctly centered). Such pixels are assigned to the closest template (detected position) as long as the distance to the center of the closest template is smaller than half of the maximum between the width and the height of the template. If, after eroding and dilating, the foreground area enclosed in each template contains multiple blobs, only the *biggest blob* that is closest to the center of the template is kept.

Sometimes it is not possible to generate a proper background model because the lighting

(a) Foreground area (b) Foreground mask (c) Foreground delimited by a template

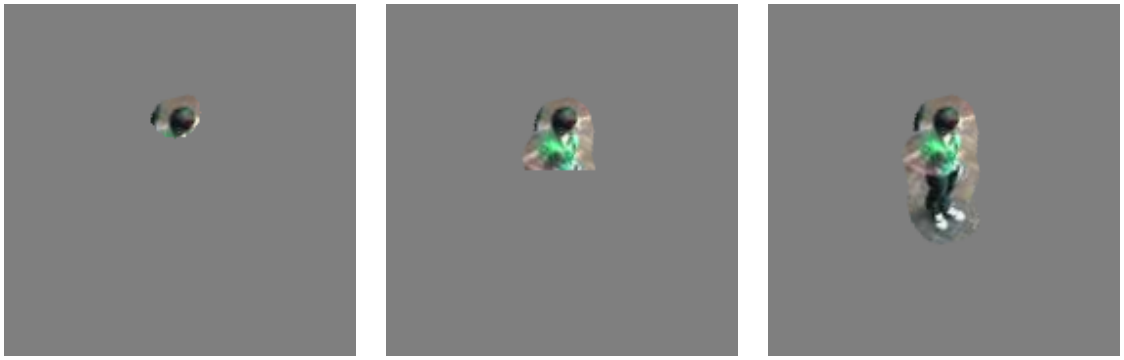
Figure III.4: Foreground area extraction

conditions change too much throughout the video or because the camera is moved from its initial position. For these cases the system was extended to be able to use the *ground truth annotations* instead of the *people-detection*. In the situation in which the *ground truth* is used, only the templates offer information about the pixels corresponding to each person in the image and no background model is available. Given this fact, the image area assigned to each individual is strictly depending on the quality of the parameters set for the camera calibration — this topic is discussed in more details in appendix A. The image locations used (both annotated and detected) are *filtered* such that only those locations corresponding to templates that completely fit in the image, are kept.

The features can be extracted out of the *whole* area corresponding to a person — area delimited by the template's extremities or the area given by the foreground mask. There is a large diversity in the appearances of people when the whole body is considered: different clothing textures, different sizes, different colors. Apart from adding a large amount of variance in the data corresponding to the same target orientation, the use of the complete body also increases the dimension of the data. For these reasons it was made possible to have features extracted from only parts of the body: *the upper half of the body* or from only the *predicted location of the head*. In the case in which ground truth and templates are used to retrieve the foreground area, the location of the head is estimated by considering the pixels enclosed in the top-most parallelogram, $\mathcal{P}$. Here, *the top-most part* refers to the area that is the farthest away from the projection of the camera position in the image plane. As it can be noticed in figure III.4(c) this solution is not always guaranteed to produce the desired outcome. For the case in which a foreground mask is available, a circle is defined in the topmost part of the foreground; the diameter of this circle corresponds to the maximum between the 2 sides of the parallelogram $\mathcal{P}$ and the circle is centered on the x-axis of the foreground area. Even in the second situation, when the foreground mask is available, the predicted head area is not guaranteed to be the correct one because the background model is not always accurate.

As we can observe in figure III.5, instead of using a black background around the extracted foreground, a gray one is used — the average gray color: *(127,127,127)*. This has the advantage of being the mean value for each channel and it will not affect, to a large degree, the outcomes of different image processing steps such as image normalization or histogram equalization.

A window around the template corresponding to each location is extracted as it is depicted in figure III.4. Sometimes the people can be located around the borders of the image — to



(a) Head area (b) Upper half of the body (c) Complete body area

Figure III.5: Body parts used for feature extraction — these correspond to the rotation of figure III.4(a)

make sure that the template is always completely enclosed in the window and centered on the feet location, *a fixed border* (half of the window size) is added around the original input image. The obtained window is rotated relative to the camera position. This step represents a form of normalization that makes the task of the learner less complicated. Figure III.5 illustrates the rotated versions (with different body parts retained) of the foreground area displayed in figure III.4(a). The window is rotated around the feet location with the angle θ' computed as follows:

$$\theta' = \frac{\pi}{2} + \theta \quad \text{where} \quad \theta = \arctan\left(\frac{y_{head} - y_{feet}}{x_{head} - x_{feet}}\right) \tag{III.4}$$

The head location, $\boldsymbol{H} = (x_{head}, y_{head})$, and feet position, $\boldsymbol{F} = (x_{feet}, y_{feet})$, of the person in the image are used in this formula because the projection of camera location in the image plane, $\boldsymbol{C} = (x_{cam}, y_{cam})$, is collinear with the feet and the head location in the order: C, F, H. The fact that the point $(0, 0)$ is positioned in the top-left corner of the image implies that the coordinate system of the image is vertically flipped as displayed in figure III.6.

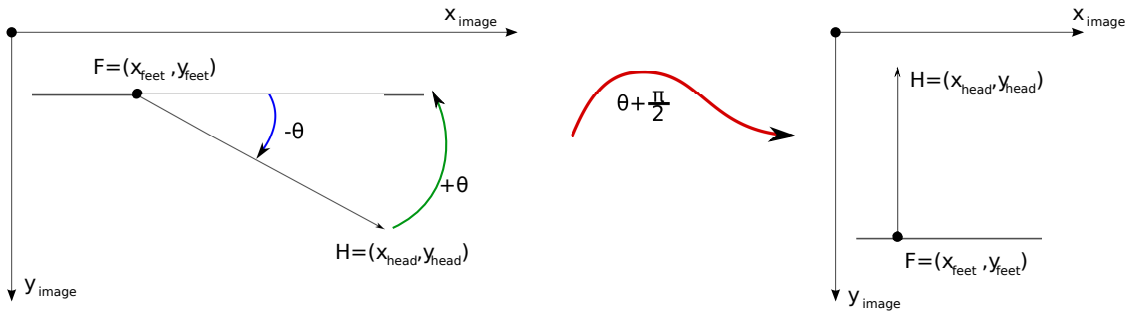


Figure III.6: Determining the rotation angle

For consistency reasons the *target longitudinal angles* also have to be changed such that they would be *relative to the camera position* in the image plane. This is acquired by adding to the annotated angle, α, the rotation angle:

$$\alpha' = (\alpha + \theta') \mod 2\pi \tag{III.5}$$

where θ' is the angle defined in equation (III.4).

For the features that are not based on the detection of key-points, after the window is rotated, *a sub-window* is cut out. This smaller window is delimited by the extremities of the template/foreground mask. The sub-window is resized to a fixed size that depends on the body part that is used for extracting features: *whole body area, upper half of the body or head area.*

Because the orientation data is rather scarce, an idea for extending the dataset is to, also, add to the training set the *horizontally flipped* versions of the images. For the case in which this option is included the target angles need to be changed to retrieve their flipped equivalents.

$$\alpha'' = \begin{cases} \alpha' + (\pi - 2\gamma) \mod 2\pi & \text{if } \alpha' \in (0, \frac{\pi}{2}] \text{ or } \alpha' \in (\pi, \frac{3\pi}{2}] \\ \alpha' - (\pi - 2\gamma) \mod 2\pi & \text{otherwise} \end{cases} \tag{III.6}$$

where α' is given by equation (III.5) and γ is defined in equation (III.7).

$$\gamma = \begin{cases} \alpha' \mod \frac{\pi}{2} & \text{if } \alpha' \in (0, \frac{\pi}{2}] \text{ or } \alpha' \in (\pi, \frac{3\pi}{2}] \\ \frac{\pi}{2} - (\alpha' \mod \frac{\pi}{2}) & \text{otherwise} \end{cases} \tag{III.7}$$

A more clear explanation of these equations is given in figure III.7 which depicts the angle that needs to be flipped — α', the flipped angle — α'' defined in equation III.6 and the angle γ defined in equation (III.7), for all 4 possible situations.

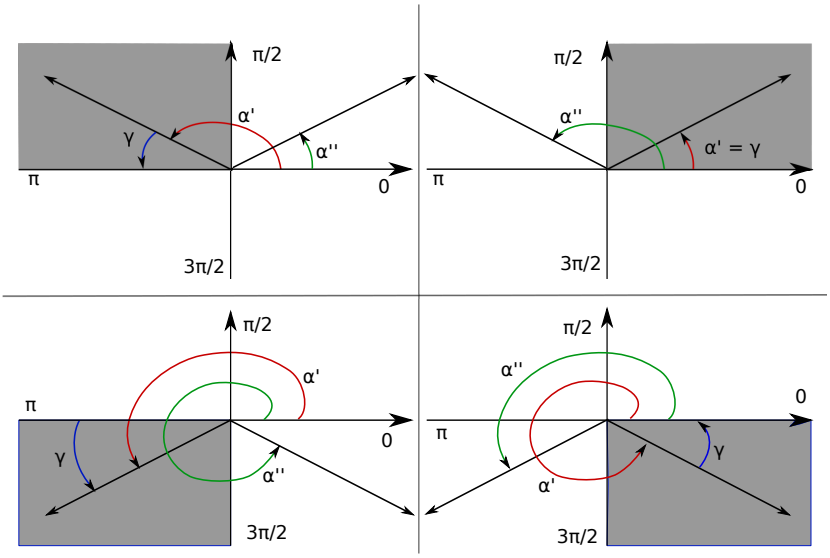


Figure III.7: Determining the flipped target angle

Chapter IV

Orientation Estimation

The task undertaken in this project is determining the direction in which a person is looking — the head orientation which does not necessarily need to coincide with the body/feet orientation. Thus, extracting features from the head area should be more informative than using the rest of the detected foreground body. The difficulty encountered in the problem of *orientation estimation* for data recorded with a *single elevated side-camera* is that it is rather difficult to obtain a good alignment for the detected foreground areas. Given the low quality of the images and the positioning of the camera, it is not possible to use face detection and predicting the head area is not always guaranteed to give good results. Due to the setup of the camera the assumption that the head is located at the top-most part of the foreground area will not hold anymore. Another reason for which extracting the head-area is difficult is that the background model is not always reliable: sometimes there are a large amount of background pixels added around the head area of a person, or some other times the pixels corresponding to the head are missing because their values are too close to the ones of the underlying background. These issues will generate shifts in the extracted foreground areas, and for the learners it will be hard to discriminate between similar orientations and orientations that greatly differ from each other.

Ideally, we would like to find features that are *invariant to shifts* and, at the same time, can encode the orientation information in a manner that would yield satisfactory prediction results. The learners applied have to be able to return *continuous outputs* and learn to predict an orientation that is not present in the training dataset by making use of the seen orientations.

In chapter II are overviewed a few methods that have been employed for the task of head/body orientation estimation or pose estimation. The features used in these methods are: *histograms of orientated gradients*, *shoulder width*, *head location relative to the body* and *facial skin and hair contrast*, *edges*, *body contour*, *Haar responses*, *SIFT optical flow*, *SIFT and bag-of-words*, *Gabor responses*. In the majority of the papers reviewed the results are reported on images taken with multiple cameras out of which one is a side-camera, or with cameras positioned at human-level or just slightly above. This simplifies the task of aligning the extracted foreground areas. Another characteristic that simplifies the problem in these cases is that the geometry of body (position and distance of the head relative to the shoulders and feet) remains predictable regardless of the position of the people with respect to the camera. This is not the case for top-down camera views, as it can be seen in one of the examples presented in figure IV.14 where the head area is positioned in the center of the foreground blob. In the situation in which an elevated camera is used, the positions of the feet and the head change depending on

the location of the person with respect to the camera.

The task on which this paper is focused, on the other hand, is more difficult and the features mentioned above might not be sufficient for obtaining good orientation estimations. In section IV.1 some of these features are described in more detail and modifications are suggested, when applicable, to make them suitable to our setup.

IV.1 Feature Extraction

To be able *to reuse the extracted features*, and thus, increase the speed of the system, we compute the features over the whole image and store them locally. Before the learning phase, the feature matrix is loaded for each image. The *region of interest* corresponding to each window centered on a person location is cut out and rotated relative to the camera position. For the features that are not based on the detection of interest points, non-linear dimensionality reduction is applied in the form of image-region rescaling. The same form of dimensionality reduction is also traditionally applied to other image-recognition problems such as character recognition [Lecun et al., 1998]. Depending on the feature, a window of size 5×5 to 20×20 pixels is obtained. For the rest of the features only the descriptors extracted from the area contained in the template/foreground mask are retained.

In the case in which the data is more or less sequential, the *direction on which the person is moving* can be retrieved from the *people-detection system*. This information can be useful in the situation in which the data contains images of walking people. This feature is based on the assumption that humans tend to look more in the direction in which they are walking than sideways. It is highly improbable that a person would be walking and looking in opposite directions. We can concatenate this feature to the final vector of features. The *distance from the person location* to the projection of the *camera location* in the image-plane can also be added to the final vector of features because it gives a clue regarding the causes of the variance present in the data.

To avoid numeric problems (multiplying a great number of pixel values can give rise to large numbers) the data should be normalized. This is done by transforming the complete training matrix such that is has *0-mean and identity covariance*. The features are stored on each row of the data matrix such that it has N rows — the number of datapoints and D columns — the number of dimensions. The *mean*, $\boldsymbol{\mu}$, and the *standard deviation*, $\boldsymbol{\sigma}$, are computed over each dimension of the training matrix. The same vectors $\boldsymbol{\mu}$, respectively $\boldsymbol{\sigma}$ obtained from the training data are also used to transform the test data. This transformation is depicted in equation IV.1.

$$\begin{aligned} \mu_d &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} feature_{id}, \quad \forall d \in \{1,..,D\} \\ \sigma_d &= \sqrt{\tfrac{1}{N}\textstyle\sum_{i=1}^{N}(feature_{id}-\mu_d)^2}, \quad \forall d \in \{1,..,D\} \\ \boldsymbol{feature}_i &\leftarrow \tfrac{(\boldsymbol{feature}_i-\boldsymbol{\mu})}{\boldsymbol{\sigma}}, \quad \forall i \in \{1,..,N\} \end{aligned} \tag{IV.1}$$

where $feature_{id}$ represents the element on row i, column d of the data matrix and $\boldsymbol{feature}_i$ is the i^{th} row of the data matrix.

To reduce the dimension of the data and allow for a faster classification by considering only the parts of the features that are actually important, *principal component analysis* — [Pearson, 1901] can be applied. This is done by defining the principal subspace from the

training data matrix, choosing a number of n principal components (dimensions to be retained) and, then, projecting both training and testing on this subspace.

The possibility of using *multiple concatenated features* is also made available. In this case the features are concatenated one after another for each row of the data matrix. For the situation in which the dimension of the resulting data is too large, *PCA* can be employed.

IV.1.1 Grid of Interest Points

The positioning of the interest points in the image could contain important local information. For example we would expect that the face area will contain a larger amount of interest points compared to the head area when the person is looking away from the camera. Based on this assumption the use of the *grid of interest points* was investigated. The input image is converted to a *grayscale* image and *interest points* are detected. For the detection of the salient points in the image, a set of important image coordinates are discovered using *Harris corner detector* ([Harris and Stephens, 1988]) as well as the points situated on the boundary of the estimated *maximally-stable extremal regions (MSERs — [Matas et al., 2002])*.

Harris corner detector is based on the evaluation of a matrix H defined by the 2 eigenvalues of the matrix, Q, depicted in equation IV.2:

$$Q = \begin{bmatrix} S_x{}^2 & S_x S_y \\ S_y S_x & S_y{}^2 \end{bmatrix} \tag{IV.2}$$

where $S_x = G_\sigma \otimes I_x$ is a smoothed partial derivative of the image, I_x is the partial derivative of the image on the x-axis, G_σ is a *Gaussian kernel* and $S_y = G_\sigma \otimes I_y$, I_y is the partial derivative of the image on the y-axis. The selected corner points are the local maxima of H.

For determining the *MSER*s, the image intensities are thresholded with several increasing values of the threshold. The connected components are extracted after each such process. Thresholds are found such that each considered region is maximally stable — the current threshold gives a local minimum of relative growth. The boundary points of each such region can then be retrieved.

(a) All the interest points (b) Corresponding foreground region (c) Retained foreground (d) Retained interest points

Figure IV.1: Detection of interest points: red points — *MSERs*, blue points — *Harris corners*

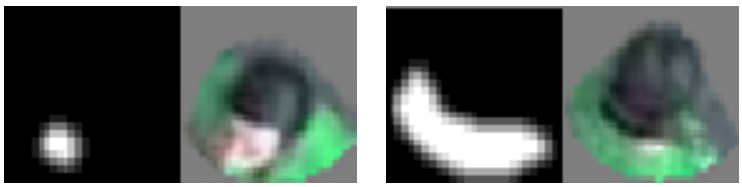
Figure IV.1 illustrates the detected interest points for a typical image. The interest points are extracted over the complete image as depicted in figure IV.1(a). Using the templates, only the desired foreground area is retained and all the interest points that are more or less enclosed in this area are retrieved IV.1(d).

The templates employed for detecting people locations (as depicted in figure III.4(c)) are used to extract a window around the person. For the case in which the features should be extracted from only a specific part of the detected foreground area (e.g. head area or upper half of the body), the template is changed such that its extremities would enclose only that area. The window that is delimited by the template extremities is divided into a grid of $n \times n$ cells. In each such cell are binned the interest points falling in that area. Finally, the feature is normalized — divided by the total number of points enclosed in the template.

IV.1.2 Mask of Skin-Pixels

Being able to detect the face — the region corresponding to skin in the image — can be extremely helpful in the task of orientation estimation. For the skin pixel extraction, the image is converted to the *hue-saturation-value (HSV)* colorspace. Because it is possible that the background color resembles the skin color, the foreground mask is *eroded* such that there are fewer background pixels around the boundary of the foreground. The pixels values corresponding to skin are determined using the method proposed in [Lu et al., 2007] by splitting the image into its 3 channels and computing:

$$\boldsymbol{skin}_{\boldsymbol{image}} \leftarrow \frac{\boldsymbol{channel}_S + \boldsymbol{channel}_V}{\boldsymbol{channel}_H} \tag{IV.3}$$



(a) Good detection (b) Erroneous detection

Figure IV.2: Skin pixels extraction

The resulting skin-image is processed using *dilation* followed by *erosion* and blurred with a median kernel. Finally, the skin matrix is thresholded to keep only the highest values. Figure IV.2 displays two examples of the obtained skin masks. As we can observe, although the correctly detected skin region can be very informative of the orientation, this method does not alway give a reliable indication regarding the location of the face.

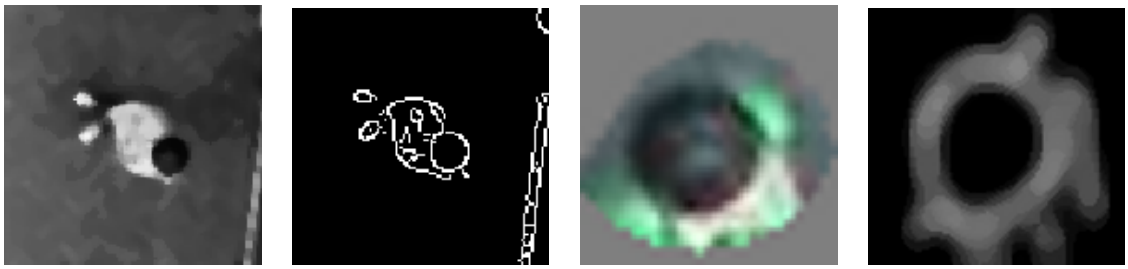
IV.1.3 Edges

The edge information has the advantage that it reduces the variance caused by the people's appearances. *Canny edge detector* [Canny, 1986] is used in [Rogez and Orrite, 2007] and in [Ozturk and Aizawa, 2009] for extracting the edge information. For the *edge* extraction the image is converted to *grayscale*. *Histogram equalization* is applied to enhance the details and increase the contrast of the grayscale image. The process of *histogram equalization* reassigns the intensities of the pixels such that the histogram of intensities becomes uniform. To remove the small details, the image is blurred with a *median kernel* of size 5×5.

In the next step, *Canny edge detector* [Canny, 1986] is employed to retrieve the edges. *Canny edge detector* is based on a few processing steps: the noise is reduced by convolving the image

with a *Gaussian filter*, the *image derivatives* in the horizontal and vertical directions are estimated and the edges' gradients and directions are computed. The next stage is called *non-maximum suppression* and it retains only those edge points that are characterized by gradient magnitudes larger than their neighbors on the direction of the edge. The last step in determining the edges uses *two thresholds*: a high one and a low one. The high threshold selects the strong edges and starting from these, the edges that are continuous (positioned on a curve containing strong edges) and over the low threshold are traced on the image.

After the edges are detected, the window corresponding to the desired foreground area is cut and the image is blurred with a median kernel such that when computing the difference between two edge-images the result will be less affected by small shifts in the position of the edges. The idea of applying this processing step to the edge image is inspired by [Stenger et al., 2004] where the authors assert that: *when using edges as features, robust similarity functions need to be used when comparing a template with the image, i.e. ones that are tolerant to small shape changes. One way to achieve this is to blur the edge image or template before correlating them. Other methods, which are tolerant to small shape deformations and some occlusion are the (truncated) chamfer and Hausdorff distance functions* [Stenger et al., 2004].



(a) Grayscale image (b) Detected *Canny* edges (c) Foreground area (d) Retained edges

Figure IV.3: Edge detection

Figure IV.3 illustrates the detected edges on an example image. The foreground mask corresponding to the desired foreground area (IV.3(c)) or the people-detection template is used to retain only the wanted edge information (IV.3(d)).

Contours can also be retrieved from the detected edges. The longest contour — representing the longest vector of connected pixels — is retained and the x and y coordinates of the points of the contour are stored as a single row and returned.

IV.1.4 SURF Features

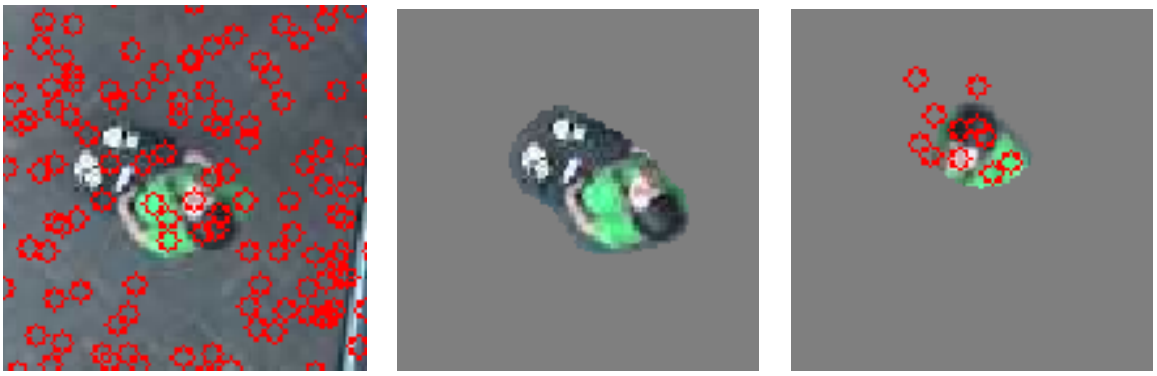
The *speeded up robust features (SURF)* — [Bay et al., 2006] descriptors rely on the estimation of a distribution of *Haar-wavelet* responses at a neighborhood around the interest points. The idea of using *SURF* descriptors is inspired by [Ozturk and Aizawa, 2009] in which the authors assert that they have also performed experiments on these features. Furthermore, the *SURF* descriptors are based on the estimation of *Haar* responses which are also employed in [Van den Bergh and Van Gool, 2009] and in [Shimizu and Poggio, 2004].

The input image is converted to *grayscale*. The interest points are determined using the *multi-scale Hessian detector [Mikolajczyk and Schmid, 2002]*. This detector resembles to a certain degree the *Harris corner detector*. As in the case of the *Harris corner detector* the detection

of interest points can be done iteratively for different window scales. For this, the image derivatives are smoothed by a Gaussian kernel with a specific standard deviation, σ_i: $S_x = G_{\sigma_i} \otimes I_x$, $S_y = G_{\sigma_i} \otimes I_y$. The *Hessian detector* is based on the estimation of the *Hessian* matrix displayed in equation IV.4;

$$H = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{yx} & S_{yy} \end{bmatrix} \tag{IV.4}$$

where $S_{\alpha\beta}$ represent the second derivatives of the smoothed image.



(a) Detected interest points (b) Corresponding foreground region (c) Retained interest points

Figure IV.4: Detected *Hessian* interest points

Figure IV.4 displays the salient points discovered by the *multi-scale Hessian detector*. As in the case of the *grid of interest points*, we only retain the points that are enclosed in the people-detection templates (figure IV.4(c)).

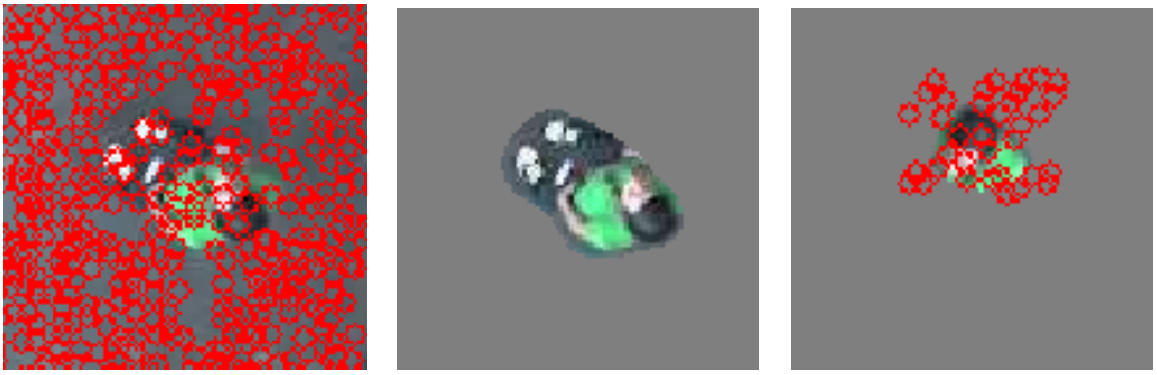
In the *SURF* method, the *Hessian detector* is modified to use *integral images* — which reduce the computational time — and it is called *fast-Hessian* [Bay et al., 2006]. The *integral image*, also called *summed area table*, defines the value at each point (x, y) as the sum of the values of all pixels positioned above and to the left of (x, y) including the current pixel. *SURF* descriptors are extracted at each detected interest point.

The *SURF* descriptors are sorted in descending order by their responses. Only the top n descriptors that are enclosed in the detection template are kept. The templates are modified to surround only the desired foreground area. The x and y coordinates of the interest points are concatenated to the extracted features. This idea is borrowed from [Zhao et al., 2008] where the same technique is applied to *SIFT* features.

IV.1.5 SIFT Features & Codebook Method

The idea of using *SIFT* descriptors and the *codebook method* is inspired by [Zhao et al., 2008] in which this technique is employed for human pose estimation. For this, the input image is converted to a grayscale image. *The keypoints' locations are selected at maxima and minima of a difference of Gaussian function applied in scale space* [Lowe, 1999]. To find the interest points that are stable, a *Laplacian pyramid* is used and the points that correspond to high variations in scale and region are retained. The *Laplacian pyramid* is obtained from the *Gaussian pyramid* by computing the *difference-of-Gaussian* images. This is done by estimating the difference between two consecutive layers of the *Gaussian pyramid* where the coarser layer is upsampled.

Scale invariant feature transform (SIFT) [Lowe, 1999] descriptors are computed by taking a 16×16 pixels square window around the keypoints and estimating the edge orientation for each pixel in the window as: $\theta - 90^o$, where θ is the gradient orientation. The gradient magnitudes are thresholded and only the strong edges are retained. The window is divided into a 4×4 grid and an *orientation histogram* of size 8 (the orientations are binned every 45^o) is built out of each cell. The resulting histograms are concatenated, giving rise to a feature of size *128*.



(a) Detected interest points (b) Corresponding foreground region (c) Retained interest points

Figure IV.5: SIFT interest points

The detected *SIFT* keypoints for an example image are displayed in figure IV.5. Figure IV.5(a) illustrates all the interest points detected over the input image. The retained interest points — figure IV.5(c)— are chosen such that they are enclosed in the people-detection template.

For the *bag-of-words* approach a *codebook* or dictionary is built by extracting *SIFT features* out of a separate set of images. The features are extracted only over the desired area of the foreground: complete body area, upper half or head area. The whole set of features is clustered into a given number, n, of clusters, also called *visual words*. Each *visual word* is then normalized such that it has length 1.

Given a new window corresponding to a person, the *SIFT features* are extracted and then normalized such that they also have length 1. Distances are computed from each new *SIFT* feature to each *visual word*. A histogram of size n is computed by assigning each new feature to its closest dictionary word. Each bin contains counts representing how many times each dictionary word is present in the input window. Finally, the histogram is normalized such that all its values add up to *1.0*. Similar to the case of *SURF features*, the idea from [Zhao et al., 2008] is also employed here and the x and y coordinates of each keypoint are concatenated at the end of each *SIFT feature*.

IV.1.6 Histograms of Oriented Gradients

Histogram of oriented gradients (HoG) [Dalal and Triggs, 2005] descriptors are employed in the system of [Enzweiler and Gavrila, 2010] where the orientation estimation and the pedestrian detection is done in a unified manner. Based on this work, we have decided to also investigate the use of *HoG* descriptors in our task. The estimation of *HoG* descriptors requires that the image is first converted to grayscale. The *HoG descriptors* are extracted over the resized window delimited by the template/foreground extremities.

As the name indicates it, the *HoG descriptors* rely on the distribution of intensity gradients (direction of the edges). To compute these descriptors, the input image is separated into smaller regions that are called *cells*. A histogram of edge orientations is built for each cell. The final descriptor represents the combination of all these histograms. Results that are more robust to changes in illumination are obtained by defining a *measure of local histogram "energy" over somewhat larger spatial regions ("blocks") and using the results to normalize all of the cells in the block* [Dalal and Triggs, 2005].

In our system, the region-of-interest corresponding to the retained foreground area is relatively small. Due to this fact, we set the *block* size and the *cell* size, to be employed for the *Hog* descriptor extraction, to equal values. This means that the no *blocks* are used in our implementation.

IV.1.7 Gabor Responses

As we have seen in chapter II, *Gabor responses* can be extremely useful in the task of head-orientation estimation [Wu and Toyama, 2000]. For this reason we also research the use of *Gabor wavelets* in our work. The input image is converted to a grayscale image and then convolved with a number of *Gabor filters* [Daugman, 1985]. The *Gabor filters* respond strongly to components in the image that have a specific orientation and frequency. A *Gabor wavelet* represents a product of a *Gaussian* with a *sine* or a *cosine* which gives rise to a symmetric or antisymmetric filter. The formulas for the *Gabor filters* are depicted in equation IV.5, respectively IV.6.

$$\boldsymbol{gabor}_{symmetric}(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \cos(2\pi\frac{x'}{\lambda} + \psi) \tag{IV.5}$$

$$\boldsymbol{gabor}_{antisymmetric}(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \sin(2\pi\frac{x'}{\lambda} + \psi) \tag{IV.6}$$

where x' and y' are given by:

$$x' = x\cos\theta + y\sin\theta \quad \text{and} \quad y' = -x\sin\theta + y\cos\theta \tag{IV.7}$$

The parameter λ represents the frequency of the stripes in the wavelet, θ gives the angle of the stripes, ψ is a translation parameter, σ gives the size of the stripes while γ indicates how elliptical they are. For the feature extraction part only symmetric *Gabor filters* have been

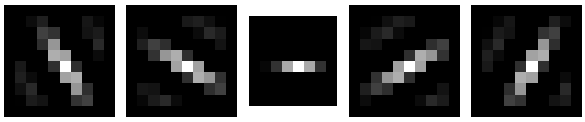
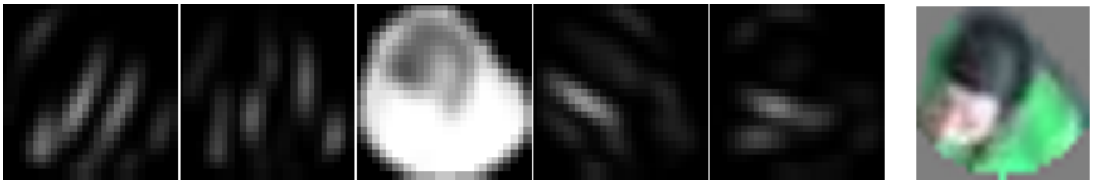


Figure IV.6: The employed *Gabor* filters

employed with the orientations: $\frac{\pi}{6}$, $\frac{\pi}{3}$, $\frac{\pi}{2}$, $\frac{2\pi}{3}$ and $\frac{5\pi}{6}$. Because the head area is rather small compared to the rest of the image, only small scale *Gabor wavelets* are employed. Figure IV.6 displays the used *Gabor wavelets*.

The responses of the above 5 filters (in the same order) and the underlying foreground window can be observed in figure IV.7. Note that the responses depicted in figure IV.7 are



(a) *Gabor* responses for the *Gabor filters* depicted in figure IV.6

(b) Foreground window

Figure IV.7: *Gabor* responses and the foreground window

the rotated responses with respect to the camera position and only the areas delimited by the template/foreground-mask extremities are retained. The results are concatenated and reshaped as a row.

IV.1.8 Template-Matching Responses

The head area contains a larger amount of information regarding the direction in which a person is looking. To make use of this knowledge, we match head-templates over the input images. A set of 8 *head-templates* are generated corresponding to the orientations: 0, $\frac{\pi}{2}$, π, $\frac{3\pi}{2}$. The head templates are created by segmenting out the head area of a random person present in one of the datasets. Different orientations are obtained by flipping the image horizontally and editing it. The generated head-templates are blurred with a *Gaussian filter* at a large scale. Each such *head-template* is resized to the head-size given by the detection template. The templates are, then, compared against overlapping regions from the input image that correspond in size with the *head-template*. For the template-matching process the *OpenCV* library is used.

To compare the image regions and the templates, multiple methods are available and for this task the *CV_TM_CCOEFF_NORMED* method is chosen. For this, equation IV.8 [OpenCV,] is used.

$$\boldsymbol{Response}(x,y) = \frac{\sum_{x',y'} \boldsymbol{T}'(x',y')\boldsymbol{I}'(x+x',y+y')}{\sqrt{\sum_{x',y'} \boldsymbol{T}'(x',y')^2 \sum_{x',y'} \boldsymbol{I}'(x+x',y+y')^2}} \tag{IV.8}$$

where $\boldsymbol{I}'$ ans $\boldsymbol{T}'$ are given by:

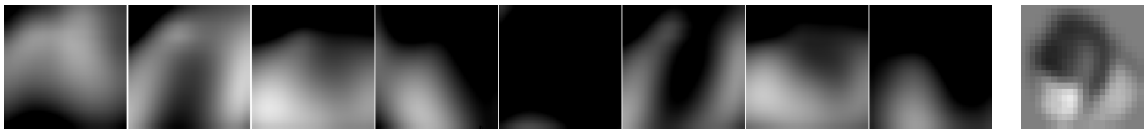
$$\boldsymbol{I}'(x+x',y+y') = \boldsymbol{Im}(x+x',y+y') - \frac{1}{w_{\boldsymbol{Templ}} h_{\boldsymbol{Templ}}} \sum_{x''y''} \boldsymbol{Im}(x+x'',y+y'') \tag{IV.9}$$

$$\boldsymbol{T}'(x',y') = \boldsymbol{Templ}(x',y') - \frac{1}{w_{\boldsymbol{Templ}} h_{\boldsymbol{Templ}}} \sum_{x''y''} \boldsymbol{Templ}(x'',y'') \tag{IV.10}$$

where $w_{\boldsymbol{Templ}}$ represents the width of the head-template and $h_{\boldsymbol{Templ}}$ is the height of the head-template. The head-templates used are depicted in figure IV.8. The obtained responses of all 8 comparisons are concatenated and reshaped as a single row. Figure IV.9 displays the responses resulting from matching the grayscale image illustrated in figure IV.9(b) with the head-templates shown in figure IV.8.

(a) Head templates for dark hair (b) Head templates for light hair

Figure IV.8: Head templates for dark hair and light hair

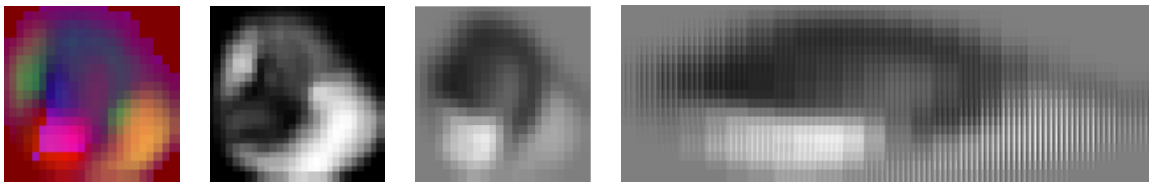


(a) Responses obtained by matching the head-templates from figure IV.8 with the foreground area (b) Foreground area

Figure IV.9: Templates-matching responses and the input image

IV.1.9 Raw Pixel Values

The actual pixel values can be extremely informative. A large difference should be obtained when subtracting the pixel values of a person looking towards the camera and the ones corresponding to a person looking away from the camera. The shifts in the position of the foreground area, as well as the variance in the clothing colors and hair colors of the people will affect the results. The pixel values are also used as a feature. For this case multiple situations have been tested: using the grayscale image, using the *saturation channel* and using the color image.



(a) HSV (b) Saturation (c) Grayscale (d) Concatenated channels

Figure IV.10: Raw-data used as a feature

Given the that the *saturation* channel of the *hue-saturation-value (HSV)* colorspace is known to be invariant to *shadows* and *shading*, this channel can be employed instead of the grayscale image. The final image is reshaped as a row and used as a feature in the learning step. An example is displayed in figure IV.10 showing how the grayscale image, saturation-channel and the concatenated color channels look like, for a certain input window.

IV.2 Classification & Regression

The problem of orientation estimation is a regression problem and not a classification one because the output is a continuous variable in the interval: *[0, 360)* degrees. Given this, the most appropriate approach to solve this problem is to look at the techniques that can

be employed in the task of predicting continuous variables. Two such methods are: *neural networks — multi-layer perceptron (MLP)* and the *Gaussian process (GP)*.

Although the undertaken task requires *regression* as a method for predicting the orientations on new images, classification can also be used if the space is discretized into a number of, say, *36* classes, each trained on orientations that differ from each other with at most 10 degrees. The problem with this approach is that the error will be at least 10 degrees because the classifier will not be able to differentiate between orientations that fall in the range of the same class. Although this is a considerable downside of this approach, a few classifiers have been considered, mainly for the purpose of comparing their performance with the performance of the tools used for regression.

The *Gaussian process* is based on computing a kernel function over combinations of features. The kernel functions used are *stationary* — they are functions of the differences between the input features. The advantage of the *stationary kernels* is that they are known to be *invariant to translations* in the input space. Thus, it is to be expected that it actually can outperform the other methods. For the rest of the methods a possible improvement can be obtained by computing *4 PCA models*. The models are built for each feature type by splitting the data into 4 clusters: orientation $\in [0, \frac{\pi}{2})$, orientation $\in [\frac{\pi}{2}, \pi)$, orientation $\in [\pi, \frac{3\pi}{2})$ and orientation $\in [\frac{3\pi}{2}, 0)$ and computing the corresponding *eigenspaces*. The input feature is projected on all *4 eigenspaces* and distances are computed from the original feature to the backprojection of the feature on each eigenspace. The resulting distance matrices are used as training data.

An idea that was investigated is *combining the outputs of multiple learners* trained over different features by using a voting system. The motion vector associated with each person is used to filter out the predictions that differed from this direction with more than *90* degrees. Here it is used the assumption that people look at most *90* degrees away on either side from the direction in which they walk. In consequence, there are only *180* valid orientations left, around the walking direction. The predictions of the learners are binned in *18* bins (every *10* degrees) and the angle corresponding to the bin containing the largest number of votes is returned as the final prediction. Unfortunately, this method is time consuming and it is only applicable in the cases in which the data is sequential and the motion vector determined from the tracks' information is accurate. Another downside of this approach is that it actually returns discretized predictions, although the learners might be able to predict continuous orientations by using regression rather than classification.

IV.2.1 Regression with angles

Despite that the orientation angles represent continuous variables in the interval *[0, 360)* there is a discontinuity at the transition from *360* degrees to *0* degrees and this fact can substantially affect the predictions because although the difference between the angles at the beginning of the first quadrant and the angles at the end of the last one is the largest, the corresponding images and features are very similar. A solution for this problem is to convert the target angles into their corresponding *sine* and *cosine* components because they have the property of being continuous functions of the input angle.

For this case two *learners* are trained separately and the final prediction is obtained by estimating the *arctangent* of the ratio: $\frac{\sin\theta}{\cos\theta}$ and taking into account the quadrant of the angle indicated by the signs of the *sine* and *cosine*. We could choose instead the angle whose quadrant is in accordance with the information given by both the *sine* and *cosine*. The reason for which we use the *arctangent* of the radio between the *sine* and *cosine* is that it represents the optimal

prediction — the prediction that minimizes the *sum-of-squares error*. The probability of the predicted angle given the predictions is: $p(\alpha \mid x, y) = \mathcal{N}(\sin\alpha; y, \sigma_s^2)\mathcal{N}(\cos\alpha; x, \sigma_c^2)$ where y is the prediction for the sine and x is the prediction for the cosine. In consequence, we want to minimize the following function:

$$E_{SSE}(\alpha) = \frac{1}{2}(\sin\alpha - y)^2 + \frac{1}{2}(\cos\alpha - x)^2 \tag{IV.11}$$

We can accomplish this by setting the derivative of this function with respect to the prediction angle, α, to 0:

$$\frac{\partial E_{SSE}(\alpha)}{\partial\alpha} = 0 \iff (\sin\alpha - y)\cos\alpha - (\cos\alpha - x)\sin\alpha = 0 \tag{IV.12}$$

$$x\sin\alpha - y\cos\alpha = 0 \iff x\tan\alpha - y = 0 \tag{IV.13}$$

$$\tan\alpha = \frac{y}{x} \iff \alpha = \arctan\left(\frac{y}{x}\right) \tag{IV.14}$$

Furthermore, we would like to be able to use the information given by the variance of the prediction for both the *sine* and *cosine* to determine the optimal prediction angle, α. Unfortunately, when predicting the *sine* and the *cosine* and maximizing the *log-likelihood function*, the resulting equations are particularly complex and have a number of 8 solutions.

An alternative idea is to learn the $\sin^2$ and the $\cos^2$ instead of just the *sine* and the *cosine* of the target angles. This idea was investigated because for this case we could combine the mean and the variance of the predictions to find an optimal prediction angle. This would be done by optimizing the probability of the prediction angle given the predictions: $p(\alpha \mid x, y) = \mathcal{N}(\sin^2\alpha; y, \sigma_s^2)\mathcal{N}(\cos^2\alpha; x, \sigma_c^2)$ where, again, y is the prediction for the squared sine and x is the prediction for the squared cosine. Thus, we need to determine the optimal prediction angle α for which: $\frac{\partial log\ p(\alpha|x,y)}{\partial\alpha} = 0$; this implies that the equation from formula IV.15 should hold.

$$\frac{y\cos^2\alpha}{\sigma_s^2} - \frac{x\sin^2\alpha}{\sigma_c^2} = \left(\frac{1}{\sigma_s^2} - \frac{1}{\sigma_c^2}\right)\sin^2\alpha\cos^2\alpha \text{ where } \sin^2\alpha + \cos^2\alpha = 1 \tag{IV.15}$$

Unfortunately, the experiment was not effective because the quadrant information is lost when learning the *squared sine* and the *squared cosine* instead of the *sine* and *cosine*.

IV.2.2 Gaussian processes

In the Gaussian process viewpoint, we dispense with the parametric model and instead define a prior probability distribution over functions directly. It might seem difficult to work with a distribution over the uncountably infinite space of functions. However, for a finite training set we only need to consider the values of the function at the discrete set of input values $\mathbf{x}_n$ corresponding to the training set and test set data points. [Bishop, 2006].

In order to predict a continuous variable, we would like to determine a function, f, such that we can estimate $f(\mathbf{x})$ for any possible input value of $\mathbf{x}$. Instead of trying to determine the parameters of this function by making an assumption about how the function looks like (e.g. a linear function of the input), the *Gaussian process gives a prior probability to every possible function where higher probabilities are given to functions we consider more likely.* [Rasmussen and Williams, 2006]

The *kernel trick* is employed to allow the use of large feature spaces compared to the number of datapoints. The *kernel trick* permits us not to define the mapping into the higher dimensional feature space, but instead, to directly define the *kernel function (covariance function)*: $k(\boldsymbol{x}, \boldsymbol{x}')$. In

consequence, the functions can be more complex; they do not need to be restricted to linear functions of the input.

The *Gaussian process* establishes a distribution over functions evaluated at a set of points, X_*: $f^* \sim \mathcal{N}(0, K(X_*, X_*))$ and which is completely specified by the mean function, $m(x)$, and the covariance function, $k(x, x')$ — for two datapoints x and x'. The covariance matrix K is a squared matrix whose element on the position (i, j) is: $k(x_i, x_j)$. The mean function is usually taken to be zero.

The joint distribution of the observed targets, $\mathbf{y}$, and the function's values at the test locations is given by equation IV.16.

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \tag{IV.16}$$

where the observations, $\mathbf{y}$, are assumed to be noisy observation with the variance of the noise σ_n^2, $\mathbf{X}$ is the set of training datapoints and X_* represents the set of test points.

The predictive distribution will have the mean and the covariance indicated in equation IV.17, respectively IV.18.

$$mean(f_*) = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} y \tag{IV.17}$$

$$cov(f_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*) \tag{IV.18}$$

For the task of this project multiple *kernel functions* have been investigated:

- *squared-exponential* — $k(x, x') = \exp(-\frac{1}{2l}\sum_i (x_i - x_i')^2)$
- *exponential covariance (Matérn covariance with* $\nu = 0.5$*)* — $k(x, x') = \exp(-\frac{1}{l}\sqrt{\sum_i (x_i - x_i')^2})$
- *Matérn covariance with* $\nu = 1.5$ — $k(x, x') = (1 + \frac{\sqrt{3}\sqrt{\sum_i (x_i - x_i')^2}}{l}) \exp(-\frac{\sqrt{3}\sqrt{\sum_i (x_i - x_i')^2}}{l})$
- *Matérn covariance with* $\nu = 2.5$ — $k(x, x') = (1 + \frac{\sqrt{5}\sqrt{\sum_i (x_i - x_i')^2}}{l} + \frac{5\sum_i (x_i - x_i')^2}{3l^2}) \exp(-\frac{\sqrt{5}\sqrt{\sum_i (x_i - x_i')^2}}{l})$

where l is the *lengthscale* and represents *the distance you have to move in the input space before the function value can change significantly* [Rasmussen and Williams, 2006].

The *Matérn covariance function* becomes simpler if the $\nu = 1/2 + p$; p is an integer and has the property $p \geq 0$.

$$k_{\nu=p+1/2}(r) = \exp\left(\frac{-\sqrt{2\nu} r}{l}\right) \frac{\Gamma(p+1)}{\Gamma(2p+1)} \sum_{i=0}^{p} \frac{(p+i)!}{i!(p-i)!} \left(\frac{\sqrt{8\nu} r}{l}\right)^{p-i} \tag{IV.19}$$

where $r = \sqrt{\sum_i (x_i - x_i')^2}$ and $\Gamma(n) = (n-1)!$ [Rasmussen and Williams, 2006].

As it was previously mentioned, for the task of orientation estimation two *Gaussian processes* are trained: one that learns to predict the *sine* of the target angle and one for the *cosine* of the target angle. We would like to be able to predict both the *sine* and *cosine* using the same *Gaussian process* and having a two dimensional target for each orientation. Unfortunately, this is a very difficult task and few researchers have tried to tackle it.

Because there are two different *Gaussian processes* trained for each orientation, their *lengthscales* and *noise variances* do not need to coincide. Given that the latitudinal angles also influence the aspect of the people and the resulting features, it would be desirable to be able

to include the latitudinal angles along with the longitudinal angles as training targets for the *Gaussian process*. This poses the same problem as above: training on multi-dimensional targets.

It is to be expected that some parts of the features are more important than others: we would expect that the features extracted from the head area should play a more important role than the rest of the features. It was previously mentioned in section IV.1 that the distance from the person's location to the projection of the camera's location in the image plane is also concatenated as a feature. This information could be more important than the rest of the feature because it explains some of the variance present in the data. It is possible to weight this feature more than the rest and, thus, indicate its importance by changing the value of the lengthscale only on its corresponding position in the feature vector. For example, the *squared exponential* in this case would be rewritten as:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-\frac{1}{2}\left(\frac{\sum_{i\neq j}(x_i - x_i')^2}{l^2} + \frac{(x_j - x_j')^2}{l'^2}\right)\right] \tag{IV.20}$$

where the value corresponding to the distance to the camera is situated on position j in the vector and l' is the *lengthscale* defined for it.

Because the only requirement that the kernel functions have is that their equivalent matrix form is *positive definite* — a matrix, $\boldsymbol{M}$, is *positive definite* if it is symmetric and has the property that $\boldsymbol{\alpha}^T\boldsymbol{M}\boldsymbol{\alpha} > 0$ for all vectors $\boldsymbol{\alpha}$ with real elements — other kernel functions can also be defined. The idea of using the *shape-matching function* offered by the *OpenCV library* was investigated. For this situation the features are represented by the *contours* that are obtained from the edge information as explained in subsection IV.1.3. Unfortunately, this idea did not give satisfactory results which indicates that either the feature or the kernel function is not sufficient for this task.

IV.2.3 Neural Networks — Multi Layer Perceptron

The *multi layer perceptron (MLP)* or *feed-forward neural network* represents another tool that can be employed for solving regression problems. *Multi layer perceptrons* define a fixed number, M, of *basis functions* that are adaptive. M linear combinations of the inputs are computed using a set of weights $w_{ji}^{(0)}$ that give rise to the *activations* a_j. *An activation function* $h(\cdot)$ is applied to the *activations* to obtain the hidden units, z_j. The hidden units are again linearly combined using the weights $w_{kj}^{(1)}$ and the activation function is applied again to the result. This process can be repeated multiple times depending on the number of hidden layers. Thus, the basis functions are adaptive: the parameters of these functions (the weights) are changed during the training stage.

Neural networks have the quality of being able to learn a mapping from a set of input vector to a set of target vectors whose dimensions can be larger than 1. For the case of *neural networks* we can actually learn to predict both the *sine* and *cosine* of the orientation angle on the same network. Provided that there is enough training data we can also add the sine and the cosine of the *latitudinal* angles next to the *longitudinal* ones and try to learn a mapping from the feature space to target vectors of size 4.

Although we can define a network as complex as we want, the more hidden layers we add to the network, the more training data we should have available and the longer the training process will take. For the task of orientation estimation the defined architecture of

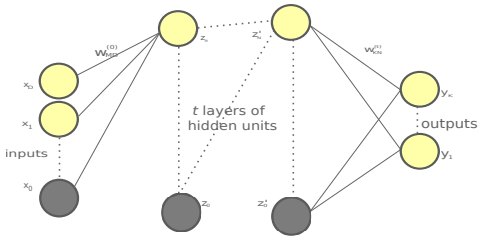


Figure IV.11: Multi Layer Perceptron

the network embodies 2 layers, each comprising 100 hidden units. The activation function used is the *sigmoid function* which is depicted in equation IV.21.

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{IV.21}$$

IV.2.4 Regressive K-Nearest Neighbors

To check the validity of the features, two *k-nearest neighbors (k-NN)* [Cover and Hart, 1967] classifiers are also trained: one that learns to predict the *sine* of the target angle and the other one for the *cosine* of the target angle. This method is not useful for regression problems because it can only return the label of the datapoint that is closest to the test datapoint. It can not learn to predict unseen orientations. Also, from the computational point of view it is rather expensive because it requires a search through the complete training set of points for each test point. *k-NN* is not guaranteed to give satisfactory results unless the distribution that we are trying to learn permits a clear separation in the feature space between the classes.

Although there are numerous reasons for which *k-NN* is not the most indicated choice, it can offer a lower boundary for the error. We do not know how the distribution of the data looks like so it could be possible that a simple *k-NN* classifier would render a correct separation of the classes. On the other hand, our problem is a *regression* problem so we would expect that the *multi layer perceptron* and the *Gaussian process* can outperform the *nearest neighbor* classifier. The classifiers are not actually *regressive* despite their name: to predict the final orientation each classifier searches for the closest 3 datapoints to the current one and, then, it returns the average over their target values.

IV.2.5 Eigen-Orientations

A method very similar to the *k-nearest neighbors* that is based on generating *eigen-orientations* was also investigated. Unfortunately the data is too complex for this approach and there are not enough examples for every orientation range such that it would allow for good results to be obtained.

For this attempt *eigen-orientations* are computed by concatenating vertically all the training images whose target angles are within *10* degrees from each other, giving rise to a set of *36* matrices for the longitudinal angles. Each matrix is, then, used to compute an *eigenspace* that is stored.

During the evaluation, each row of the test matrix is projected on all possible *36 eigenspaces*. The result is backprojected afterwards, to reconstruct the image. Finally, the distances from

each backprojection to the original image are computed. The predicted angle is chosen to be the angle corresponding to the *eigenspace* that generates the smallest backprojection error.

IV.3 Experiments & Results

The experiments strive to answer 7 main research questions: which one is the best learning method, the best feature, the generalization over positions and people, the most informative body part for our task, the importance of different processing steps, the generalization of the system over orientations, and the performance on new data. For the experimental part special attention has to be payed such that the results are not biased in any way: if very similar frames are present in the training and in the evaluation set, then, the process could be biased in our advantage because the learner would have already seen input points that are very close to the evaluation ones and would be able to predict the labels correctly. On the other hand, having frames that resemble the ones from the training in the evaluation can also be in our disadvantage: for example if, in the training set, there is a frame of a certain person oriented *90* degrees and in the evaluation set there is a similar frame but, now, the person is oriented *180* degrees, the learner might be driven to predict an orientation of *90* degrees for the frame that appears in the evaluation set.

IV.3.1 Data description

The difficulty of our task is mainly given by the data used, which is a challenging one. The images are recorded with a *ceiling-mounted camera*, the quality of the images is low and they are characterized by a large amount of image noise. The fact that the camera is positioned at a high altitude makes it difficult to observe the faces of the people in the images, especially if they are gazing downwards.

IV.3.1.1 Dataset 1

(a) Position 1 (b) Position 2 (c) Position 3 (d) Position 4

Figure IV.12: Examples of images from *dataset 1* — presenting the 4 locations with respect to the camera available in this dataset

For the experimental part, the results are reported on three datasets. The first dataset — *dataset 1* — contains images of two people, positioned at different locations in the ground plane and having a large number of different orientations in the range *[0, 360)*. Figure IV.12 illustrates a few images from this dataset. *Dataset 1* contains a number of *1345* images with orientations at most *15* degrees from each other in the range *[0, 360)*.

IV.3.1.2 Dataset 2

(a) Person A (b) Person B (c) Person C (d) Person D (e) Person E (f) Person F (g) Person G

(h) Person H (i) Person I (j) Person J (k) Person K (l) Person L (m) Person M (n) Person N

Figure IV.13: Examples of images from *dataset 2* — presenting all 14 people that appear in this dataset

The second dataset on which experiments are performed is *dataset 2*. In this dataset there are *14* people present that are walking in 8 main directions: $\{0^o, 45^o, 90^o, 135^o, 180^o, 225^o, 270^o, 315^o\}$ and endeavor to cover the whole ground plane area in the common view. This dataset is more diverse than *dataset 1*, as it contains different people with distinct sizes and appearances. It has the disadvantage, however, that the quality of the images is lower compared to *dataset 1* because the camera was positioned at a higher altitude. This fact makes the learning task more difficult. Although there are more people present in this dataset, there is still not enough variance in it and some of the people that appear in the video could be very distinct from the rest and, thus, their orientations will be harder to estimate. Figure IV.13 depicts a few examples of images from this dataset. We can notice in figure IV.14 that the position with respect to the camera affects greatly the appearance of the people. The same person, with the same orientation but at different positions looks completely different.

Figure IV.14: Variations in the data due to the position with respect to the camera

IV.3.1.3 Dataset 3

The data used in [Ozturk and Aizawa, 2009] represents the last dataset employed in the experimental part. It was also created using an elevated sideway camera. This dataset was recorded in an airport and it contains images of people walking in different directions. Unfortunately, the images contained in this dataset are not aligned (the camera was shifted from its initial position during the recording) so extra effort had to be put in aligning the images such that we could retrieve a good background model and have good detections. Usually the people

present in this video are walking in a specific direction throughout the whole sequence in which they appear. Given this reason, this dataset is more fit for evaluation than for training. The experiment performed on this dataset uses *dataset 2* for training, fact which may influence the results in an undesired way — the lighting conditions and the quality of the images are different as well as the background.

Figure IV.15: Examples of images from the dataset used in [Ozturk and Aizawa, 2009]

IV.3.1.4 Artificial data

To make sure that the examples present in the data cover the range *[0, 360) degrees* as much as possible, *artificial data* was generated using $3\mathcal{D}$ models[1] of men and women orientated every 5 degrees in the range $[0, 360)$. The models have been created with different body poses and the latitudinal angles are in the range: *[60, 110]* degrees. A good calibration for the artificial data can be easily found such that the bodies of the people are accurately centered in the templates. The artificial data is separated into *clusters*: *605* images of people positioned close under the camera, *593* images of people located a bit farther away from the camera and *592* images of people positioned far from the camera's location. Each set contains *10* models of people — *2* models of men and *2* models of women with *5* different appearances (different clothing and hairstyle). To gain a certain degree of shift invariance and to make the artificial data closer to the real one, the artificial data can be augmented with images in which the annotated template locations are slightly shifted to the top, bottom, left and right. A few examples of images comprised in this dataset are displayed in figure IV.16

Figure IV.16: Examples of images from the *artificial dataset*

IV.3.2 Experimental setup

For the performance evaluation three methods have been implemeted:

- the *RMS (root-mean-squared error)*: $\sqrt{\frac{\sum_i |\theta_i - \theta_i'|^2}{N}}$

[1]The models were generated by Jeroen Groen — $3\mathcal{D}$ Engineer: http://www.jeroengroen.com/

- the *normalized RMS*: $\sqrt{\frac{\sum_i \frac{|\theta_i - \theta'_i|^2}{\pi^2}}{N}}$
- the *average difference* between the target angles and the predictions: $\frac{\sum_i |\theta_i - \theta'_i|}{N}$

where θ_i is the target angle for the i^{th} datapoint, θ'_i is the predicted angle and N represents the total number of evaluation datapoints. Due to the discontinuity of the angle, in the case in which the difference between the predicted angle and the target angle, $\Delta_i = \mid \theta_i - \theta'_i \mid$, is larger than π, the quantity: $(2\pi - \Delta_i)$ is used, rather than Δ_i. If the difference between the prediction and the target angle is larger than π when the difference is computed in the trigonometric direction, then we can use, instead, the difference between the angles in the clockwise direction. Figure IV.17 describes more clearly this situation. To be able to examine the tendency in the error, the differences between the target angles and the predictions — Δ_i — are *binned* in 18 bins (every *10* degrees) and plotted as histograms.

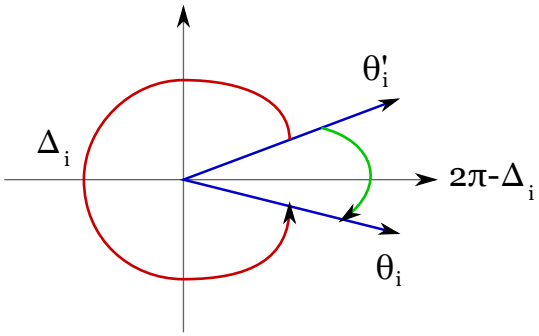


Figure IV.17: Estimating the difference between the predicted angle and the target angle

It is also possible to *train on multiple datasets* (with different camera calibrations and different background models) and then evaluate on a completely different set. The change in the illumination between the training set and the evaluation set affects the overall performance, but the advantage of this idea is that there is more training data available.

Because *dataset 1* is more limited, the only experiments reported are: training on images of one person at all locations and evaluating on different frames corresponding to the same person, and training on a person at different positions with respect to the camera and then evaluating on the same person but at a new position. As already stated, it is desirable to avoid biasing the evaluation when researching the generalization of the system over people. For *dataset 2* one way of achieving this is to make sure that no person present in the training set will also occur in the evaluation set. Thus, the dataset containing images of *14* different people is split such that the training is done on images in which *13* people are present and the evaluation is done on the images of the person that is not included in the training. The performances estimated over the *14* rounds are averaged such that we can obtain an overall performance of the system for this dataset. This method can be regarded as a *14-fold cross-validation* with the only extra constraint added to it that no frames containing people present in the training set should occur in the evaluation set. For each person 72 images were annotated — *9* different positions with respect to the camera for, somewhat, the same orientation.

For the dataset used in [Ozturk and Aizawa, 2009] we train on *dataset 2* and evaluate on this dataset alone. The downside of this method of evaluation is that the lighting conditions

are different in the training and in the evaluation and this fact can dramatically influence the results. From this set *50* images were annotated and used, containing *150* people in total.

Due to the fact that the appearance of a person changes depending on the position of that person with respect to the camera, an idea is to split the data in 3 classes. The projection of the camera coordinates in the image plane: $(X_{cam}, Y_{cam}, Z_{cam}) \xrightarrow{projection} (x_{cam}, y_{cam})$ is computed and the euclidian distance from this point to the person's location is estimated. Each detection is, then, assigned to one of the 3 classes: *close, medium* and *far* based on the measured distance. Unfortunately, this idea did not prove to be efficient because the number of training images assigned to each class was insufficient.

For the *Gaussian process* the *squared exponential* kernel function was found to be more effective than the rest, therefore this kernel function is employed in all experiments in which the *Gaussian process* is used. Estimating the location of people with the *people-detection system* is slower than using the annotated locations of the people — *ground truth* — and selecting the foreground are enclosed in the templates. But the second method is more prone to error (more background is selected along with the pixels corresponding to a person).

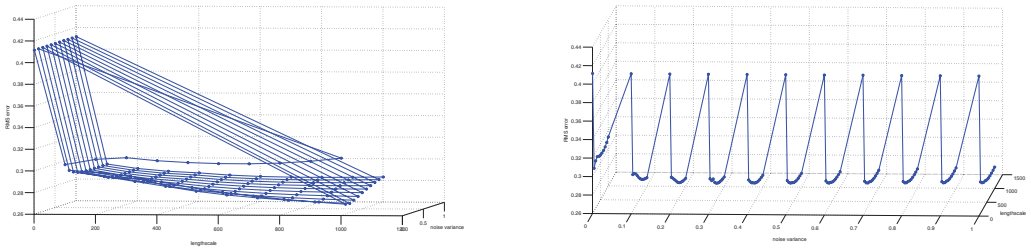
IV.3.3 Results & Performance Analysis

In this subsection we present the results of all our experiments and give a brief analysis of these outcomes. The first thing that we want to investigate is which learning method is more suitable for our task. In the next part, we present comparative results over different features for the best method. We indicate, afterwards, how the *Gaussian process* generalizes over different people and the generalization of the *Gaussian process* over different positions. The performance of the *Gaussian process* on different body parts is, also, researched. Next, we estimate the performance by considering or removing different processing steps such as rotation with respect to the camera. We are mainly interested if we can learn orientations and this is presented in the next part. Finally, we present the results obtained by the *Gaussian process* on the dataset employed in [Ozturk and Aizawa, 2009].

IV.3.3.1 Experiment 1 — Comparison of learning methods

To determine the most appropriate learning method for the task of orientation estimation, the *concatenated color channels* of the upper half of the body are employed as a feature. All the experiments are performed on *dataset 2*. In this dataset, some of the people are very distinct from the rest. For this reason, the two people corresponding to the largest two errors — figure IV.13(d) and figure IV.13(e) — were removed form the dataset. In consequence the training is done on *11* people and the evaluation on the 12^{th} person. For each person there are *72* images annotated, thus, for each fold we train on *792* images and test on *72* images.

The parameters of *Gaussian process* (the *lengthscale* and the *noise variance*) are estimated experimentally by looping over different settings of the parameters and evaluating the predictions on a separate validation set. Figure IV.18(a) shows how the normalized *RMS* error changes with the lengthscale and figure IV.18(b) depicts the change in the error when the noise level is varied. The error was estimated separately on the *sine* and *cosine* and it was observed that when, in both cases, the parameters are set to the same values determined experimentally, the *sine* had an error slightly larger than the *cosine*. To make the error corresponding to the *sine* closer to the one obtained for the *cosine*, the *lengthscale* of this *Gaussian process* is adjusted.



(a) Lengthscale (b) Noise

Figure IV.18: Estimation of *lengthscale* and *noise variance* for the *concatenated pixel values* on the validation set

The *multi layer perceptron* is trained on network architectures comprising 2 layers of hidden units, each composed out of *100* nodes. The training is done using the *error backpropagation* method. The errors are reported using different combinations: training the network by including both *sine* and *cosine* targets in the same network — using a target vector of size 2 for each input sample, training two different networks and combining the results, training on *latitudinal* and *longitudinal* targets in the same network. For the error estimation on the *k-nearest neighbor*, the number of neighbors to be considered, k, is set to *3*.

Learner	**Setting of the experiment**	**RMS Error** *(Normalized RMS)*
NN	*Cosine* and *sine* on the same network (2 outputs)	**1.08** *(0.34) — 62 degrees*
NN	*Cosine* and *sine* on different networks (1 output)	**1.13** *(0.36) — 64 degrees*
NN	*Longitude* and *latitude* on the same network (4 outputs)	**1.20** *(0.38) — 68 degrees*
k-NN	*Cosine* and *sine* trained separately	**1.08** *(0.34) — 60 degrees*
GP	*Cosine* and *sine* trained separately	**1.01** *(0.32) — 57 degrees*

Table IV.1: RMS error over different learners and different settings

We can notice that when the *sine* and *cosine* of the angles are learned on the same *NN*, the error is slightly smaller. This can be regarded as a form of *transfer learning*, which explains the better performance. On the other hand, when the *latitudinal* angles are added to the network and we try to learn a mapping to target vectors of size 4, the error increases. This can indicate that the amount of training data is not sufficient for learning both *latitudinal* and *longitudinal* angles or that the architecture of the network is not complex enough. We can observe that the *Gaussian process* outperforms the *NN*. The reason for this outcome is that *the distribution of functions generated by a neural network will tend to a Gaussian process in the limit* $M \rightarrow \infty$ [Bishop, 2006], where M represents the number of basis functions used by the *neural network*. Given these results, the further experiments are all using the *Gaussian process* for learning.

IV.3.3.2 Experiment 2 — Comparison of features

In the experiments presented here we compare all the features available on the best learner — the *Gaussian process*. For the comparison of different features the experiments are performed on *dataset 2* using *12* people. Since these experiments are computationally expensive and time-consuming the *ground truth* annotations are employed instead of the people-detection system. The features are extracted from the upper half of the body.

The optimal hyperparameters of the *Gaussian Process* depend on the representation (feature space) of the data and also vary with the dataset: different camera positioning, different light conditions or image quality affect the possible parameter values of the *GP* — which are integrated out — and therefore the optimal values of the hyperparameters. These hyperparameters can best be learnt on a validation set, while the actual parameters of the model are learnt from the training set. In practice, however, it has been shown that the hyperparameters could well be learnt by maximum likelihood on the same data (type II maximum likelihood, [Bishop, 2006] — section 6.4). The importance of the precise value of the hyperparameters is, however, limited. In this work, the parameters are learnt by grid search. This is a time-consuming procedure, and the hyperparameters were, therefore, trained on a subset of the 9 features and, although they provide a realistic comparison of the different features, the absolute prediction accuracies could probably be improved by more careful selection of the hyperparameters.

We can observe that complex features such as *SIFT*, *SURF* and *HoG* descriptors perform worse than the *Gabor responses*, *template matching* and *concatenated color channels*. These features have an error rate that is slightly worse than random and the tendency in the predictions is opposite to what we would expect — having a large number of points falling the first bins (predictions close to the targets) and last bins of the histogram (confusion between a person facing the camera and a person looking away from the camera).

The performance on different combinations of features is also estimated. Because there are too many combinations of features possible, we started by determining the feature that gave the best results combined with the best performing feature. For combinations of three features we started from the best combination of two features and we checked to see which one out of the 7 features gives the best performance when concatenated to these two features. For combinations of 4 features we started from the best combination of 3 features. It turns out that the combinations of features do not improve on the best feature.

IV.3.3.3 Experiment 3 — Generalization over people and positions

In these experiments we research the generalization of the system over different people and the generalization over different positions. For both situations the *Gaussian process* is used for learning and the *concatenated pixel values* are extracted over the head area. The window corresponding to the head area is resized to 10×10 pixels, giving rise to a feature of size *300* pixels. The initial size of the window depends on the dataset and the detected foreground area, and its size is between 20×20 pixels and 30×30 pixels.

For the generalization over different people, an experiment is run on *dataset 2*. In this experiment the *12*-fold cross-validation method is used. The smallest error obtained here, in one round — when comparing the results of each round separately — is of: **0.64** radians (**36** degrees) which means a normalized *RMS* of **0.20**.

We, also, want to determine how well the *Gaussian process* can generalize over different

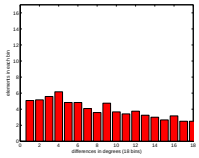
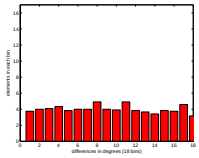
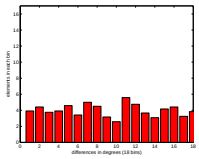
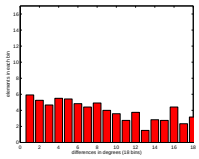
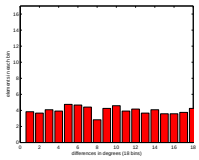
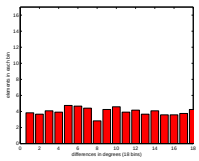
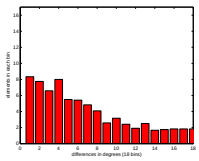
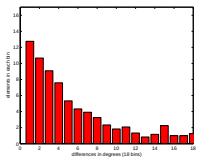
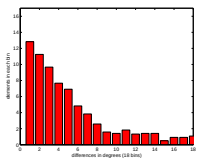
Feature	Histogram of differences	RMS Error*(Normalized RMS)*
Grid of interest points		**1.58** *(0.50) — 90 degrees*
Mask of skin pixels		**1.77** *(0.56) — 101 degrees*
Edges		**1.79** *(0.57) — 102 degrees*
SURF descriptors		**1.62** *(0.51) — 93 degrees*
SIFT descriptors		**1.79** *(0.57) — 102 degrees*
HoG descriptors		**1.79** *(0.57) — 102 degrees*
Gabor responses		**1.33** *(0.42) — 76 degrees*
Template matching		**1.14** *(0.36) — 65 degrees*
Concatenated color channels		**1.05** *(0.33) — 60 degrees*

Table IV.2: Comparative results over different features — for the histograms, the differences between the targets and the predictions, Δ_i, are binned into *18* bins.

Research question	Dataset	No. training/ evaluation people	RMS Error *(Normalized RMS)*
Generalization over people	*Dataset 2*	11/1 (12-folds)	**0.98** *(0.31) — 56 degrees*
Generalization over positions	*Dataset 1*	1/1 (4 folds)	**0.84** *(0.26) — 48 degrees*
Generalization over positions	*Dataset 2*	12/12	**0.82** *(0.26) — 47 degrees*

Table IV.3: Results showing the generalization over positions and the generalization over people

positions. For this task two experiments are performed. In the first experiment we employ the data from *dataset 1*. The training and the evaluation is done on the same person. The orientations of the person present in the training set and in the evaluation set are more or less the same and they are uniformly spread in the interval *[0, 360)*. In consequence, the only significant difference between the training frames and the evaluation frames is the position of the person with respect to the camera. Given that there are 4 different positions available in *dataset 1* — displayed in figure IV.12 — for one person, the process is repeated 4 times: at each round training on 3 locations and evaluating on the left position. For each position a number of *100* images are used. The second experiment performed on this task uses images from *dataset 2*. As mentioned in section IV.3.1, this dataset is more challenging due to the lower quality of the images and the great variety of locations. The training set contains *672* images of all *12* people at different locations. For the evaluation, *192* frames containing all *12* people at another set of distinct locations, are used.

IV.3.3.4 Experiment 4 — Learning on different body parts

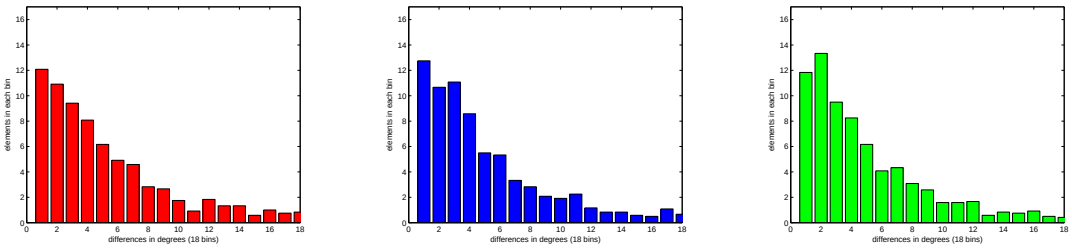
These experiments answer the research question: which part of the body contains more information about the orientation of a person. Table IV.5 displays the results obtained when

Body part used for feature extraction	RMS Error*(Normalized RMS)*
Complete body area	**1.07** *(0.34) — 61 degrees*
Upper half of the body	**1.01** *(0.32) — 57 degrees*
Predicted head area	**0.98** *(0.31) — 56 degrees*

Table IV.4: Comparative results over different body parts

training a *Gaussian process* on *dataset 2* using the *12-folds cross-validation* method for evaluation, and employing the *concatenated pixel values* as a feature.

Figure IV.19 shows the histograms of the differences between the target angles and the predicted angles obtained when learning on the *Gaussian process*. The number of degrees with which the target angles and the predicted angles differ is divided into 18 bins — *x-axis* of the histograms. The features are extracted from the *complete body area* (figure IV.19(a)), *upper half of the body* (figure IV.19(b)) or the *head area* (figure IV.19(c)). These results indicate that



(a) Complete body area (b) Upper half of the body (c) Head area

Figure IV.19: Error histograms on different body parts

both the shoulders' area and the head area encode useful information. We can notice in the histograms that, for the upper half of the body, a larger number of points fall in the first bin which indicates that it is a bit more precise, while using the head area is less often wrong. We can also notice that the complete body area acquires less good results because there is a large variance in body poses, body sizes and clothing which complicates the task of the learner.

IV.3.3.5 Experiment 5 — Results for different setups

The results displayed here indicate how the error changes when the data is processed in a certain manner or new, artificial data is added. For all three experiments we use the *Gaussian process* and the learning is done on *dataset 2* using the *12-fold cross-validation* method and extracting the *concatenated pixel values* corresponding to the head area.

Experimental setup	**RMS Error***(Normalized RMS)*
Baseline	**0.98** *(0.31) — 56 degrees*
Artificial data is added	**1.03** *(0.32) — 59 degrees*
Horizontally flipped version of the images are added	**0.98** *(0.31) — 56 degrees*
The images & targets are not rotated wrt. camera	**1.28** *(0.40) — 73 degrees*

Table IV.5: Comparative results over different settings

The fact that the error increases slightly when the *artificial data* is added is caused by a number of reasons such as: the artificial data is not similar enough to the real data, the quality of the images is better, the lighting conditions are different from the ones present in *dataset 2*. We can observe that adding the horizontally flipped versions of the images does not significantly improve the results. An explanation for this outcome is that, as we can see in the histogram depicted in figure IV.19(c), there is not a large degree of confusion between the orientations that differ from each other with *180* degrees. The increase in the error when the images and the target labels are not rotated anymore with respect to the camera proves that the rotation with respect to the camera represents a form of normalization and it simplifies the task of the learner.

IV.3.3.6 Experiment 6 — Generalization over orientations

Our goal in this part is to determine if the orientations can be learned. For the experiment presented here, we use the *Gaussian process* for learning and the *concatenated pixel values of all channels* extracted from the head area. *Dataset 1* is employed in this experiment. The window enclosing the head area is resized to 10×10 pixels, in consequence the feature used here has a size of *300* pixels.

No. training/evaluation people	RMS Error *(Normalized RMS)*	Standard deviation
1/1 (5-fold cross-validation)	**0.59** *(0.18) — 34 degrees*	**0.11** *radians — 6 degrees*

For this experiment, the data contains *626* images of one person situated at 4 different locations — as depicted in figure IV.12. The images are randomized and a *5*-fold cross-validation method is applied. These results show the basic capabilities of the system.

IV.3.3.7 Experiment 7 — Results on *dataset 3*

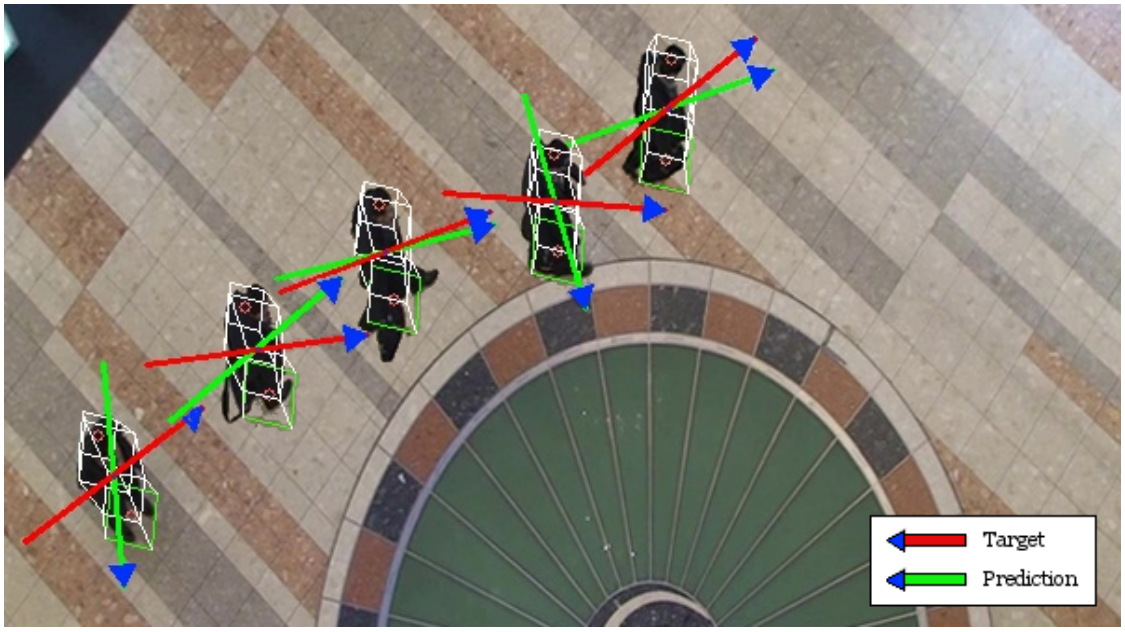


Figure IV.20: The predicted angles and the target angles for the trajectory of one person

Finally, the error is estimated on *dataset 3* which was also employed in the experiments presented in [Ozturk and Aizawa, 2009]. For this experiment the *concatenated pixel values* corresponding to the *head area* are used. Figure IV.20 shows the *predicted angles* and the *target angles* for the trajectory of one person present in this dataset. The increase in the error can be explained by the fact that in *dataset 2* the body orientations of the people correspond to the head orientations for all images. Unlike in this situation, in *dataset 3* there are a large

number of cases in which these two angles differ. As it was previously mentioned, the change in the illumination, the quality of the images and the background also influence the results.

No. training/evaluation people	**RMS Error** *(Normalized RMS)*
864/150 people	**1.07** *(0.34) — 61 degrees*

Although the authors use the *shape context matching* in combination with *Canny edges* to estimate the *body orientations*, we can not compare our results with the ones presented here. For the situation in which a side-camera is used, the body orientations are divided into 5 classes labeled as: $\frac{\pi}{4}$, $\frac{\pi}{8}$, 0, $\frac{-\pi}{8}$, $\frac{-\pi}{4}$. As it is mentioned in chapter II, these classes do not represent the directions in which people are oriented, but rather body shapes — for example a person assigned to the class $\frac{-\pi}{4}$ can be oriented *180* degrees away from the class label.

Chapter V

Conclusions & Future Work

The goal of this project is to estimate the orientation of the people in a video recorded using a single elevated sideway camera. An extensive comparison of different recognition methods and features is performed, and the impact of the two major forms of variation — position with respect to the camera and identity of the subject — are thoroughly investigates. Although the data available is not diverse enough for the problems posed by our task, our method obtains good results in more limited scenarios (constrained location or single subject). Having better data — better quality of the images, a larger number of people to train on and more orientations — would definitely improve the results. Our results show that the performance degrades gracefully as the complexity of the task increases: we still obtain informative results when testing on unseen subjects or locations. This is particularly impressive, when considering the difficulty of the task and the limited amount of training data.

An interesting thing to notice is that the simplest feature available, *concatenated pixel values*, was able to acquire better results than more complex features such as *SIFT* or *HoG*. This is possible due to the power of the *Gaussian process*.

Another surprising fact is that the simple *k-nearest neighbors* classifier was able to outperform the *multi layer perceptron*, which indicates that the orientation estimation problem can be solved by employing less complex techniques. Despite this fact, the accuracy attained by the *k-nearest neighbors* is outperformed by the *Gaussian process*, indicating that a regression method is the correct approach for this problem.

In the future, improvements, apart from a better and a more diverse dataset, can be gained by including a method of combining multiple targets for the same *Gaussian process*. The ability of training both the *sine* and the *cosine* on the same *Gaussian process* could greatly improve the results. Having a method available for learning targets of multiple dimensions would also allow to add the *sine* and the *cosine* of the *latitudinal angles* in the same *Gaussian process*.

Finally, as it was previously mentioned, not every element of the feature vector is as important as the rest. The head area should be weighted more than the pixels around it, corresponding to the background. Thus, determining specialized *lengthscales* for different parts of the features in an automatic way — gradient descent — would represent another extension to this work.

Bibliography

[Ando et al., 2007] Ando, S., Wu, X., Suzuki, A., Wakabayashi, K., and Koike, H. (2007). *Human Pose Estimation for Image Monitoring.* NTT Technical Review, 5(11).

[Barrett et al., 1994] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and der Vorst, H. V. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* SIAM, Philadelphia, PA.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Gool, L. V. (2006). *Surf: Speeded up robust features.* In In ECCV, pages 404–417.

[Belongie et al., 2002] Belongie, S., Malik, J., and Puzicha, J. (2002). *Shape Matching and Object Recognition Using Shape Contexts.* IEEE Trans. Pattern Anal. Mach. Intell., 24(4):509–522.

[Bishop, 1995] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press, Inc.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Brekhof, 2005] Brekhof, M. J. (2005). *Looking at People – Detecting People & Estimating their Facing Direction.* Master's thesis, University of Amsterdam.

[Canny, 1986] Canny, J. (1986). *A computational approach to edge detection.* IEEE Trans. Pattern Anal. Mach. Intell., 8:679–698.

[Cappé et al., 2005] Cappé, O., Moulines, E., and Rydén, T. (2005). *Inference in hidden Markov models.* Springer series in statistics. Springer.

[Chen and Wang, 2009] Chen, Y. and Wang, G. (2009). *Real-time Tracking and Pose-Estimation of Walking Human from Silhouette Images.* In Computational Intelligence for Visual Intelligence, pages 1 – 7. IEEE.

[Cover and Hart, 1967] Cover, T. and Hart, P. (1967). *Nearest neighbor pattern classification.* IEEE Transactions on Information Theory, 13(1):21–27.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An introduction to support vector machines : and other kernel-based learning methods.* Cambridge University Press.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). *Histograms of Oriented Gradients for Human Detection*. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 1:886–893.

[Daugman, 1985] Daugman, J. G. (1985). *Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters*. Journal of the Optical Society of America A: Optics, Image Science, and Vision, 2(7):1160–1169.

[Dimitrijevic et al., 2006] Dimitrijevic, M., Lepetit, V., and Fua, P. (2006). *Human body pose detection using Bayesian spatio-temporal templates*. Computer Vision and Image Understanding, 104(2).

[Englebienne and Kröse, 2010] Englebienne, G. and Kröse, B. J. (2010). *Fast Bayesian People Detection*. Template-based method for object/people detection.

[Enzweiler and Gavrila, 2010] Enzweiler, M. and Gavrila, D. M. (2010). *Integrated pedestrian classification and orientation estimation*. In Computer Vision and Pattern Recognition (CVPR), pages 982 – 989. IEEE.

[Haar, 1910] Haar, A. (1910). *Zur Theorie der orthogonalen Funktionensysteme*. Mathematische Annalen, 69(3):331–371.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). *A Combined Corner and Edge Detector*. In Proceedings of the 4^{th} Alvey Vision Conference, pages 147–151.

[Jiang, 2009] Jiang, H. (2009). *Human pose estimation using consistent max-covering*. In Computer Vision, pages 1357 – 1364. IEEE.

[Kleinman and van den Berg, 1992] Kleinman, R. E. and van den Berg, P. M. (1992). *A modified gradient method for two — dimensional problems in tomography*. Journal of Computational and Applied Mathematics, 42(1):17 – 35.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324.

[Lowe, 1999] Lowe, D. (1999). *Object Recognition from Local Scale-Invariant Features*. pages 1150–1157.

[Lu et al., 2007] Lu, Y.-x., Liu, Y., Guo, Y., Kong, L.-t., Chang, X.-q., and Shan, X.-y. (2007). *Features of human skin in HSV color space and new recognition parameter*. Optoelectronics letters, 3(4):312–314.

[Matas et al., 2002] Matas, J., Chum, O., Martin, U., and Pajdla, T. (2002). *Robust wide baseline stereo from maximally stable extremal regions*. In Proceedings of the British Machine Vision Conference, volume 1, pages 384–393.

[Mikolajczyk and Schmid, 2002] Mikolajczyk, K. and Schmid, C. (2002). *An Affine Invariant Interest Point Detector*. In ECCV (1), pages 128–142.

[Navaratnam et al., 2005] Navaratnam, R., Thayananthan, A., Torr, P., and Cipolla, R. (2005). *Hierarchical part-based human body pose estimation*. In BMVC.

[OpenCV,] OpenCV. *OpenCV camera calibration*. http://opencv.willowgarage.com/documentation/cpp/.

[Ozturk and Aizawa, 2009] Ozturk, O. Yamasaki, T. and Aizawa, K. (2009). *Tracking of Humans and Estimation of Body/Head Orientation from Top-view Single Camera for Visual Focus of Attention Analysis*. In Computer Vision Workshops (ICCV Workshops), pages 1020 – 1027. IEEE.

[Pearson, 1901] Pearson, K. (1901). *On lines and planes of closest fit to systems of points in space*. Philosophical Magazine, 2(6):559–572.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*, pages 3 – 31. The MIT Press.

[Rogez and Orrite, 2007] Rogez, G. Guerrero, J. and Orrite, C. (2007). *View-invariant Human Feature Extraction for Video-surveillance Applications*. In Advanced Video and Signal Based Surveillance, pages 324 – 329. IEEE.

[Shimizu and Poggio, 2004] Shimizu, H. and Poggio, T. (2004). *Direction estimation of pedestrian from multiple still images*. In Intelligent Vehicles Symposium, pages 596 – 600. IEEE.

[Smola and Schölkopf, 2003] Smola, A. J. and Schölkopf, B. (2003). A tutorial on support vector regression. Technical report, Statistics and computing.

[Stenger et al., 2004] Stenger, B., Thayananthan, A., Torr, P. H. S., and Cipolla, R. (2004). *Hand Pose Estimation Using Hierarchical Detection*. In European Conference on Computer Vision, pages 105–116.

[Szeliski, 2010] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*, pages 48 – 54. Springer.

[Teh et al., 2006] Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). *Hierarchical Dirichlet Processes*. Journal of the American Statistical Association, 101(476):1566–1581.

[TSAI,] TSAI. *Tsai calibration package*. http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiDesc.html.

[Tsai, 1986] Tsai, R. Y. (1986). *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision*. In Proc. Computer Vision and Pattern Recognition (CVPR), pages 364–374.

[Van den Bergh and Van Gool, 2009] Van den Bergh, M. Koller-Meier, E. and Van Gool, L. (2009). *Real-Time Body Pose Recognition Using 2D or 3D Haarlets*. International Journal of Computer Vision, 83(1):72–84.

[Wang and Zhang, 2007] Wang, F. and Zhang, C. (2007). *Feature Extraction by Maximizing the Average Neighborhood Margin*. In Computer Vision and Pattern Recognition, pages 1 – 8. IEEE.

[Wu and Toyama, 2000] Wu, Y. and Toyama, K. (2000). *Wide-Range, Person- and Illumination-Insensitive Head Orientation Estimation*. In Automatic Face and Gesture Recognition, pages 183 – 188. IEEE.

[Yang et al., 2005] Yang, C., Duraiswami, R., and Davis, L. (2005). *Fast Multiple Object Tracking via a Hierarchical Particle Filter*. In ICCV 05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV05) Volume 1, pages 212–219. IEEE Computer Society.

[Zhao et al., 2008] Zhao, X., Ning, H., Liu, Y., and Huang, T. (2008). *Discriminative estimation of 3D human pose using Gaussian processes*. In Pattern Recognition, pages 1 – 4. IEEE.

Appendix A

Camera Calibration

The performance of the orientation-estimation system is dependent on the quality of the extracted features; the features depend on how accurately the area corresponding to a person was retrieved during pre-processing. For the task of orientation estimation the first step is represented by the people detection. This is done by employing the system presented in [Englebienne and Kröse, 2010], which is described in more details in section III.1. Due to the fact that the *people-detection system* uses templates ($3\mathcal{D}$ bounding boxes), there is a need for a method to backproject image points to real-world points. The image point at which we want to build a template needs to be backprojected into its corresponding $3\mathcal{D}$ point. A $3\mathcal{D}$ template is constructed in the world coordinate system by considering the average human height *1.75m* and a given width. Finally, the whole template (the corner points of the template) is projected back in the image plane.

In order to be able to precisely extract the pixels corresponding to a person, the template needs to be able to correctly fit that person. The calibration initially used was assuming that the projection of a $3\mathcal{D}$ point on the $2\mathcal{D}$ image plane was a special case of *perspective projection* in which the 3 *Euler angles* of the camera were all equal to 0, which assumes that the image coordinate system and the world coordinate system are parallel. Therefore, for this case the backprojection from $3\mathcal{D}$ to $2\mathcal{D}$ coordinates was easily determined.

The *perspective projection* conserves straight lines but not the parallelism between the lines (it generates *points at infinity* — points where the parallel lines from the real-world get intersected in the image plane). Because the calibration model initially used did not allow to specify the camera angles, good calibration could not be retrieved for the data used in this project. A number of tools and techniques had to be investigated and implemented to overcome this problem. Throughout this chapter we present these tools and techniques which allow the templates to fit more accurately the people present in the images.

Given that our goal is to achieve a reliable calibration, the literature regarding camera calibration and, specifically, the *perspective projection* had to be consulted. In the case of *perspective projection*, the projection is determined by applying the transformations that give the conversion from the real-world coordinate system to the image coordinate system (figure A.1) and, then, dividing the $3\mathcal{D}$ point by its z coordinate to retrieve the *homogeneous coordinates* of the $2\mathcal{D}$ point in the image plane.

The $3\mathcal{D}$ point is projected on the image plane by multiplying it with the *camera matrix* which contains both the *intrinsic parameters*, K, representing the parameters that are specific to each camera and the *extrinsic parameters*: R and t representing the rotation matrix, respectively the

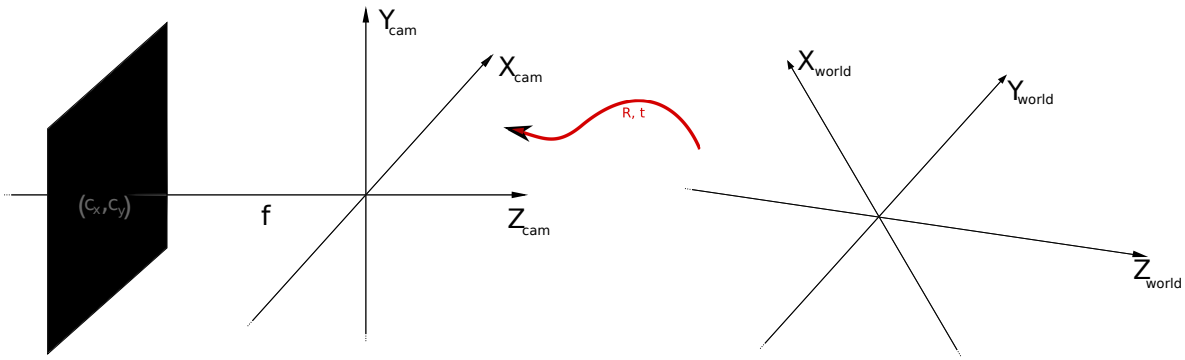


Figure A.1: Camera Calibration: (c_x, c_y) — optical center, f — focal length

translation matrix.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \boldsymbol{K}[\boldsymbol{R} \mid \boldsymbol{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{A.1}$$

The actual image coordinates are determined as: $x = \frac{x'}{z'}$ and $y = \frac{y'}{z'}$. The matrix $\boldsymbol{K}$ is called the *calibration matrix* and it contains the *camera intrinsic* parameters:

$$\boldsymbol{K} = \begin{bmatrix} f & s & c_x \\ 0 & af & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

where f is the *focal length* of the camera, *(c_x,c_y)* is the *optical center*, s — represents the *possible skew between the sensor axis due to the sensor not being mounted perpendicular to the optical axis"* [Szeliski, 2010] which in practice is set to *0*, and a is the *aspect ratio* which is usually set to *1.0* unless there is available information regarding how the height of the pixel is different from its width. The *optical axis* can be defined as the imaginary line on which the light travels through the camera lens. The *extrinsic parameters* of the camera give the rotation and translation of the camera coordinate system with respect to the world coordinate system:

$$[\boldsymbol{R} \mid \boldsymbol{t}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{A.3}$$

A.1 Retrieving Camera Calibration

There are two commonly used methods for retrieving the camera calibration: *TSAI algorithm* (TSAI calibration package) [Tsai, 1986] and the *OpenCV library* [OpenCV,]. In order to retrieve the camera calibration, both methods rely on the availability of a set of $3\mathcal{D}$ — $2\mathcal{D}$ point correspondences. These points need to be measured at relatively uniform intervals in the image plane. If the measured points are not uniformly spread over the space, these methods will, most likely, give poor results. The camera model used in both situations is a *pinhole camera* — it does not include lenses, but an ideal aperture whose diameter would be equal to the one of a light beam.

The *OpenCV library* computes out of the points correspondences the *intrinsic matrix*, $\mathbf{K}$, the *rotation vector* out of which the rotation matrix can be obtained, the *translation vector*, 6 radial distortion coefficients and 2 tangential distortion coefficients. Unlike the *OpenCV library*, *TSAI calibration package* estimates a set of 11 parameters from the point correspondences. Instead of returning the complete *intrinsic matrix*, it returns the following *intrinsic parameters*: f, $kappa_1$ — 1^{st} *order radial lens distortion* coefficient, c_x, c_y and s. And it also computes the *extrinsic parameters*: *the rotation vector* and *the translation vector*.

Unfortunately, for our experiments the measurements where either not available or not good enough (most likely the points were not spread uniformly over the ground plane). In consequence, it was not possible to retrieve a good camera calibration model with either method: *TSAI package* or the *OpenCV* library.

A.2 Projection and Backprojection

In the case in which the *OpenCV* library is used for estimating the *camera matrix*, the projection method offered here can be applied. This projection method has the advantage that it also considers the *distortion coefficients*. The same projection formula as in equation (A.11) is used, but before the multiplication by the *intrinsic matrix* the formulas that should account for the radial and tangential distortion are added:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = [\boldsymbol{R} \mid \boldsymbol{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{A.4}$$

$$x' = u\frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + 2p_1uv + p_2(r^2 + 2u^2) \tag{A.5}$$

$$y' = v\frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + p_1(r^2 + 2v^2) + 2p_2uv \tag{A.6}$$

$$\text{where} \quad r^2 = u^2 + v^2 \quad \text{and} \quad u = \frac{u'}{w'}, v = \frac{v'}{w'} \tag{A.7}$$

$$\text{finally, the image point is defined as:} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{A.8}$$

where k_1, k_2, k_3, k_4, k_5, k_6 are radial distortion coefficients and p_1, p_2 are the tangential distortion coefficients [OpenCV,].

For all the experiments presented in this paper no calibration is available, so the parameters had to be experimentally approximated by varying them such that the projection of the $3\mathcal{D}$ bounding boxes, used by the people-detection system, in the image plane would fit more or less the majority of the people present in the dataset. In the case in which it is not possible to retrieve the *camera matrix*, a simplified projection model is created in which the camera angle, $\boldsymbol{\theta} = (\theta_x, \theta_y, \theta_z)$, and the camera position in the world coordinate system, (c_X, c_Y, c_Z) and the focal length, f, need to be approximated such that a satisfactory calibration would be found:

$$z \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_X \\ 0 & f & c_Y \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{R} \begin{bmatrix} X - c_X \\ Y - c_Y \\ Z - c_Z \end{bmatrix} \tag{A.9}$$

As it can be noticed in this formula, the homogeneous coordinates are not used for the $3\mathcal{D}$ point. Also, the parameter s (the skew between the sensor axis) is set to 0, the parameter a (the aspect ratio) is set to 1 and instead of using c_x and c_y the translation parameters are used. The rotation matrix, $\mathbf{R}$, is defined in terms of the camera angle $\boldsymbol{\theta}$:

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin(\theta_x) & cos(\theta_x) \end{bmatrix} \begin{bmatrix} cos(\theta_y) & 0 & sin(\theta_y) \\ 0 & 1 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix} \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.10}$$

Unfortunately, if this allows us to project a $3\mathcal{D}$ point on a $2\mathcal{D}$ image, it does not tell us how to backproject a $2\mathcal{D}$ image point (with one known parameter e.g. its height) into the world coordinate system. This is a less common problem since for the majority of the situations only a projection method is required. A considerable amount of time was spent in solving the backprojection problem.

To backproject a $2\mathcal{D}$ point into the corresponding $3\mathcal{D}$ point, it is not sufficient to just apply the inverses of the previous transformations. In our case we are interested in points positioned in the ground plane, so we also need to enforce the constraint that the z coordinate of the backprojected point is 0. This is done by observing that the $2\mathcal{D}$ point is obtained from the $3\mathcal{D}$ point by dividing the result of the multiplications (equation (A.11)) by a scalar z'. Thus, we need to recover the value of this scalar, z', such that the backprojected point will have a 0 z-coordinate.

Because we have tried using the projection method offered by the *OpenCV* library for the situation in which the camera calibration is obtained from $2\mathcal{D}$ — $3\mathcal{D}$ point correspondences and due to the fact that this library does not offer a way to *backproject* a point, here are presented the equations deduced from the projection formulas for this case. For the case in which the *OpenCV* projection method is used, in the backprojection step the distortions are ignored and formula (A.11) is employed instead. It can be noticed that this equation can be rewritten as:

$$z \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix} = \boldsymbol{K} \left(\boldsymbol{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \boldsymbol{t} \right) \tag{A.11}$$

We need to determine the z-coordinate in the projection formula such that the backprojected point has its z-component equal to 0:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \boldsymbol{R}^{-1} \left(\boldsymbol{K}^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \boldsymbol{t} \right) \tag{A.12}$$

If we write the elements of $\boldsymbol{K}^{-1}$ and $\boldsymbol{R}^{-1}$ in an explicit form:

$$\boldsymbol{R}^{-1} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \text{and} \quad \boldsymbol{K}^{-1} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \tag{A.13}$$

then, it can be observed that the z-component of the $2\mathcal{D}$ point needs to have the following form:

$$z = \frac{t_1 r_{31} + t_2 r_{32} + t_3 r_{33}}{r_{31}(k_{11}x + k_{12}y + k_{13}) + r_{32}(k_{21}x + k_{22}y + k_{23}) + r_{33}(k_{31}x + k_{32}y + k_{33})} \tag{A.14}$$

In the case in which no camera matrix is available and the parameters need to be approximated experimentally, the same strategy as above is used for determining the $3\mathcal{D}$ coordinates. For this situation, the formula depicted in equation A.9 is implemented for the projection, thus the coordinates of the backprojected point are deduced in the following manner:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \boldsymbol{R}^{-1} \begin{bmatrix} f & 0 & c_X \\ 0 & f & c_Y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} c_X \\ c_Y \\ c_Z \end{bmatrix} \tag{A.15}$$

If we write the inverses of the rotation matrix and the calibration matrix in an explicit form:

$$\boldsymbol{R}^{-1} \begin{bmatrix} f & 0 & c_X \\ 0 & f & c_Y \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{A.16}$$

then, the value for the z-coordinate of the $2\mathcal{D}$ point in this case will be:

$$z = \frac{-c_Z}{m_{31}x + m_{32}y + m_{33}} \tag{A.17}$$

In the end we are interested in recovering the coordinates of the $3\mathcal{D}$ point: *(X, Y, Z)* as depicted in formula A.12 or in equation A.15. The problem is that when projecting the $3\mathcal{D}$ point in the $2\mathcal{D}$ image, as depicted in equation A.11 or equation A.9, we divide by the scalar z and this scalar needs to be recovered. This is done, as it was previously mentioned, by enforcing the constraint that the backprojected point should be positioned in the ground plane. The determined solutions for the two described situations are depicted in formula A.14, respectively equation A.17.