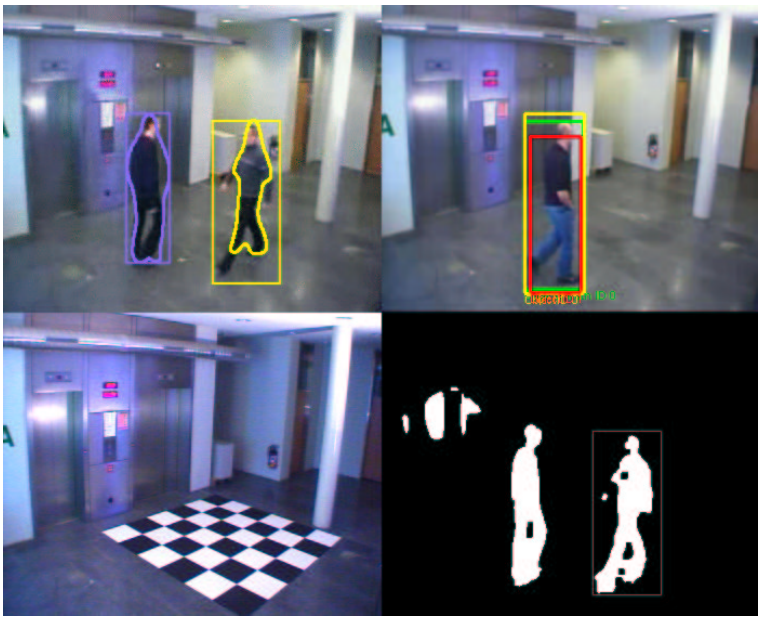
Eidgenössische Technische Hochschule
Swiss Federal Institute of Technology
Zürich

Perceptual Computing
Computer Vision

Diploma Thesis March 8, 2004

Qualitative and Quantitative Evaluation of the Reading People Tracker



Christoph Kiefer
(D-INFK)

Advisor: **Martin Spengler**

Prof. B. Schiele
Institute of Scientific Computing
ETH Zürich

Abstract

People tracking is one major part of visual surveillance systems. A number of different people trackers exist nowadays. One of those multi-people trackers is the *Reading People Tracker*. It consists of four individual co-operating tracking modules. Weaknesses of one module are thereby compensated by the strengths of another one. The *Motion Detector* performs background-foreground segmentation to extract moving regions. The *Region Tracker* tracks the regions from the Motion Detector over time. Region merging and splitting helps to overcome problems during motion detection. The *Human Feature Detector* detects possible head positions in moving regions. But the fundamental tracking module of the Reading People Tracker is the *Active Shape Tracker*. It uses a 2D appearance model of human beings, the *Active Shape Model*. A person's outlines are represented by an *Active Shape* which is actually a closed parameterized B-spline curve. An additional feature of the Reading People Tracker is its use of camera calibration to perform image measurements. This diploma thesis evaluates qualitatively and quantitatively the Reading People Tracker. In order to perform the evaluation, a large set of test video sequences is captured. A capturing and calibration environment is implemented to rapidly capture video sequences. Camera calibration is accomplished using Tsai's perspective pin hole camera model. A particular method is explained to eliminate the effects of radial lens distortion. Input image points to the calibration algorithm are computed in sub-pixel accuracy by fitting lines to a checkerboard calibration target. The focus of the evaluation part is put on the Motion Detector and the Active Shape Tracker. Evaluations are accomplished by comparing tracking output objects with a set of *ground truth* objects according to different performance metrics. These metrics are pixel-based as well as object-based. The *ViPER* package is used to annotate test video sequences and to compute the evaluation performance metrics.

Table of Contents

Chapter 1

Introduction

We live in a dangerous world. The demand for visual surveillance systems is still growing. Airports, train stations, courts and public buildings are only few examples of places where security has an extremely high priority. Security systems must provide a high degree of reliability to be credible. This means for visual surveillance systems to minimize false alarms as much as possible. This is especially true for systems that provide automatic alarming mechanisms.

One major part of visual surveillance is people tracking. A people tracker must be able to deal with many different situations. Simple situations where only one person appears in the scene, and very complex situations as multiple people in scene, large groups or people occluding each other. Tracking people in real-time is also a very important requirement of a tracking systems.

There are nowadays a lot of different tracking systems using different approaches to overcome major tracking problems. The *PCCV-Tracker* [1] is based on a Bayesian multi-people tracker. The human body is modeled as a cylinder with three segments: Legs, torso and head. In addition, a blob-based object detection system allows to detect abandoned objects in the scene. *BraMBLe* [2] is another tracker which uses a Bayesian filter for tracking multiple objects.

The goal of this diploma thesis is to evaluate qualitatively and quantitatively the *Reading People Tracker* [3] which is originally based on the *Leeds People Tracker* [4, 5]. The fundamental property of the Reading People Tracker is its use of a 2D appearance model of human beings. The *Active Shape Model* represents the shape of human beings as parameterized B-spline curves. So-called *Active Shapes* [6, 7, 8, 9, 10] are fitted to person outlines and corrected on a frame-by-frame basis. The *Active Shape Tracker* is one of four tracking modules. The *Motion Detector*, the *Region Tracker* and the *Human Feature Detector* are the other modules. The purpose of using four individual modules is to compensate the weaknesses of one module by the strengths of another one. An additional feature of the Reading People Tracker is the possibility to use camera calibration to perform image measurements.

A large data set of test video sequences needed to be captured in order to evaluate the Reading People Tracker. This data set is also considered as a benchmarking data set. Different tracking systems can be compared with the same *ground truth*. In addition, by analyzing the tracking output, algorithms can be improved to achieve better tracking performance.

A capturing and calibration environment is implemented to capture test video sequences. Tsai's perspective pin hole camera model [11, 12, 13] is used to perform camera calibration. The algorithm to perform the calibration needs a large set ($\gg 7$) of input image points. A checkerboard calibration target is used to select these image points in sub-pixel accuracy. The points are then obtained by firstly fitting lines to edges of the checkerboard and secondly, by intersecting these lines.

An important issue when evaluating tracking algorithms is to define *what* is to be evaluated and *how*. This thesis evaluates the tracking output of the Reading People Tracker. Computational performance,

i.e. real-time tracking is not examined. The tracking output for a video sequence is compared with the corresponding ground truth of that video sequence according to different performance metrics [19, 20]. In general, precision of output objects and recall of ground truth objects are analyzed. The evaluation is focused on the Motion Detector and the Active Shape Tracker. The Motion Detector is responsible for correctly identifying and extracting moving regions from video images. It therefore maintains a model of the background and uses background-foreground segmentation to identify moving regions. The Active Shape Tracker in turn tries to decide if a tracked region contains a person.

`ViPER` [21, 22] is used to calculate the performance metrics. It offers three different evaluation types: framewise, object and tracking evaluation. In this thesis only framewise evaluations are performed, i.e. tracking output and ground truth is always compared on a frame-by-frame basis.

Outline. This thesis is divided into three main parts. Starting with this introduction, chapter 2 describes the process of camera calibration. Tsai's perspective pin hole camera model is explained. To eliminate the effects of radial lens distortion, an algorithm is introduced that straightens curved lines in the distorted video image. The procedure of selecting accurate input image points to the calibration algorithm by fitting and intersecting lines is explained as well in chapter 2.

Chapter 3 introduces the Reading People Tracker. The four individual co-operating tracking modules are described. The focus of the chapter is on the Active Shape Tracker and the Active Shape Model which describes the 2D appearance of human beings.

Chapter 4 defines pixel-based, object-based and localized object-based evaluation performance metrics. Quantitative and qualitative evaluation results are presented. The last chapter of this thesis gives conclusions and remarks for future work. The appendix contains (among others) additional precision and recall curves and a short user manual of the implemented capturing and calibration environment.

Chapter 2

Camera Calibration

This chapter explains the capturing and calibration environment that was built for capturing test video sequences which will later be used to evaluate the Reading People Tracker (see chapter 4). The capturing and calibration application that was developed is `CCTool`. Refer to appendix C for a short user manual. Video sequences were captured from a Sony EVI-D100P pan-tilt-zoom (PTZ) camera [40, 41]. Video frame acquisition is implemented using the `video4linux` API [44]. The encoding of individual video frames is implemented using the `ffmpeg` library [45]. Finally, the graphical user interface (GUI) is implemented using `Qt` [43]. It is possible to supply camera calibration data to the Reading People Tracker (3) to perform image measurements, but it is not mandatory.

Basically, camera calibration is the process of finding a set of *intrinsic* and *extrinsic* parameters to specify the relation between 3D world coordinates $[X_w, Y_w, Z_w]$ and 2D image coordinates $[X_f, Y_f]$ in the camera's frame buffer. Figure 2.1 shows the reference systems for camera calibration. Tsai's pin hole camera model was used for camera calibration [11, 12, 16]. The camera model as well as intrinsic and extrinsic parameters are described in section 2.1. Section 2.2 describes implementation details and shows results of camera calibration.

2.1 Tsai Camera Model

The camera model that is used for camera calibration in this thesis is the camera model originally proposed by Tsai [11, 12, 15]. It is based on the pin hole camera model of perspective projection. Given the position of a point $P(X_w, Y_w, Z_w)$ in known 3D world coordinates $[X_w, Y_w, Z_w]$, the model projects point $P(X_w, Y_w, Z_w)$ onto its position in 2D pixel coordinates $[X_f, Y_f]$ in the camera's frame buffer. The model has five intrinsic and six extrinsic parameters. Only four of the intrinsic parameters are used for coplanar camera calibration (as is the case for this thesis). Extrinsic parameters determine the relation between world coordinates $[X_w, Y_w, Z_w]$ and camera coordinates $[X_c, Y_c, Z_c]$, i.e. they specify the orientation of the camera. Intrinsic parameters are camera specific parameters determined by the hardware of the camera. They specify the relation between camera coordinates $[X_c, Y_c, Z_c]$ in millimeters (here) and image coordinates $[X_f, Y_f]$ in pixels.

The following extrinsic parameters are approximated by the camera model:

- **$\mathbf{R_x}$, $\mathbf{R_y}$ and $\mathbf{R_z}$:** Roll, pitch and yaw rotation angles for the transformation between world and camera coordinates. In sensorics, orientations are usually described with roll, pitch and yaw angles [38]. Roll angle corresponds to a rotation around Z_w-axis, pitch angle to a rotation around Y_w-axis and yaw angle to a rotation around X_w-axis.

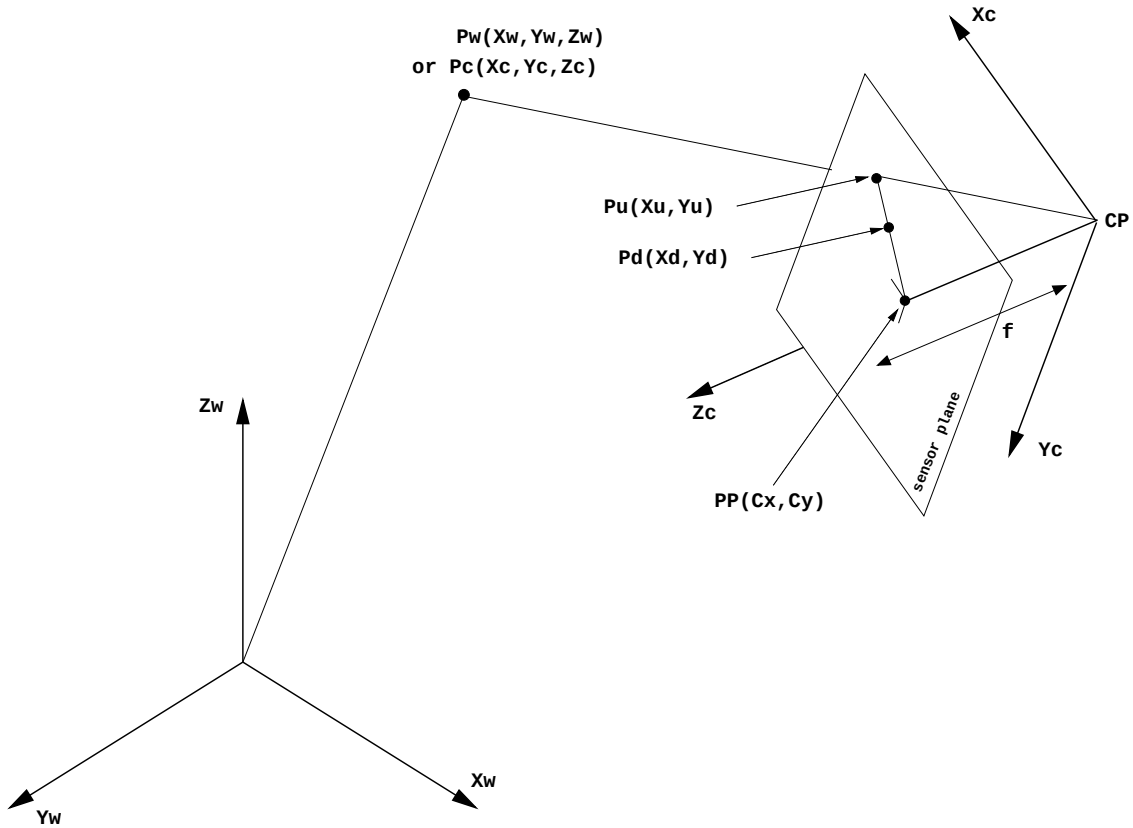


Figure 2.1: Reference systems for camera calibration. *The figure illustrates the reference systems used for camera calibration. The world coordinate system $[X_w, Y_w, Z_w]$ and camera coordinate system $[X_c, Y_c, Z_c]$ are shown. The optical axis (Z_c-axis) of the camera coordinate system is perpendicular to the camera's sensor plane. The sensor plane is assumed to be parallel to the $[X_c, Y_c]$-plane at a distance f, where f is the effective focal length of the camera. The optical axis intersects the sensor plane at the principal point $PP(C_x, C_y)$. A point P in world coordinates $[X_w, Y_w, Z_w]$ is first transformed by a rigid body transformation into point P_c in camera coordinates $[X_c, Y_c, Z_c]$. The second transformation is an ideal perspective projection of point P_c into point P_u in undistorted sensor plane coordinates $[X_u, Y_u]$. Then, point P_u is transformed into distorted sensor plane coordinates $[X_d, Y_d]$. The last transformation is between point P_d and its coordinates $[X_f, Y_f]$ in the camera's frame buffer.*

- $\mathbf{T_x}$**,** $\mathbf{T_y}$ **and** $\mathbf{T_z}$**:** The translational components of the transformation between world and camera coordinates.

Approximated intrinsic parameters are.

- **Principle Point:** Coordinates (C_x, C_y) (in pixels) of the intersection of the Z_c-axis and the camera's sensor plane.

- **Focal length f (principal distance):** Distance between the center of projection CP and the principal point PP.

- $\boldsymbol{\kappa_1}$**:** First order radial lens distortion coefficient (see section 2.1.3).

2.1.1 Scene to Camera Transformation

To transform world coordinates into camera coordinates, rotating and translating world coordinates results in corresponding camera coordinates. For a point P_w in world coordinates, this is expressed as

$$\boxed{\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = R \times \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + T} \tag{2.1}$$

where R is a 3x3 rotation matrix describing the orientation of the camera in the world coordinate system

$$R = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \tag{2.2}$$

and T a vector specifying the translation of world coordinates to camera coordinates.

$$T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} \tag{2.3}$$

Having found a solution for R_x, R_y and R_z, the orthonormal rotation matrix R is build by filling in the rotation matrix elements $r_1, \ldots, r_9$ as defined in appendix B.

2.1.2 Camera to Sensor Plane Transformation

The transformation of camera coordinates to undistorted sensor plane coordinates is an idealized 3D-to-2D projection because influences of lens distortion are not yet considered. The transformation of point P_c to an undistorted image point P_u is done using perspective projection with pin hole camera geometry. The theorem on intersecting lines is used by comparing similar triangles. The coordinates X_u and Y_u of the projected point P_c are given by the equations

$$X_u = f\frac{X_c}{Z_c} \tag{2.4a}$$

$$Y_u = f\frac{Y_c}{Z_c} \tag{2.4b}$$

2.1.3 Lens Distortion

The next step is to consider the effects of lens distortion. Figure 2.2 shows two types of distortion, barrel and pincushion distortion. Barrel distortion is associated with wide angel lenses. The image appears curved outward. Pincushion distortion is associated with telephoto lenses (maximum zoom lenses) and causes the image to appear bent inward.

Tsai's camera model only addresses radial lens distortion. Tangential lens distortion is ignored. The center of radial distortion is assumed to be at the center of the camera's sensor plane PP. The true sensor plane coordinates (X_d, Y_d) of point P_d are defined as

$$X_d = X_u - \delta_x \tag{2.5a}$$

$$Y_d = Y_u - \delta_y \tag{2.5b}$$

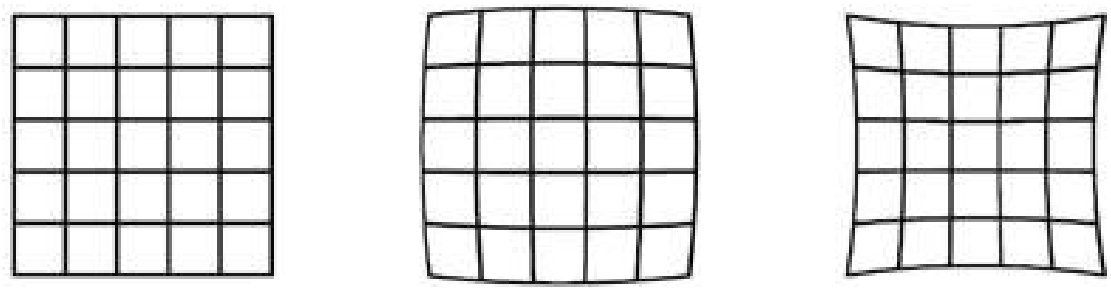


Figure 2.2: Example of barrel and pincushion radial lens distortion. *The figures shows to types of radial lens distortion, barrel and pincushion. A barrel distorted image appears curved outward whereas a pincushion distorted image appears bent inward. Barrel distortion is associated with wide angle lenses, pincushion with telephoto lenses (maximum zoom lenses).*

where δ_x and δ_y are given by the power series'

$$\delta_x = X_d\kappa_1 r^2 + X_d\kappa_2 r^4 + X_d\kappa_3 r^6 + \ldots \tag{2.6a}$$

$$\delta_y = Y_d\kappa_1 r^2 + Y_d\kappa_2 r^4 + Y_d\kappa_3 r^6 + \ldots \tag{2.6b}$$

The distance r is the radius from the principal point PP to the distorted point P_d.

$$r = \sqrt{X_d^2 + Y_d^2} \tag{2.7}$$

This results in new equations for X_u and Y_u.

$$X_u = X_d[1 + (\kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \ldots)] \tag{2.8a}$$

$$Y_u = Y_d[1 + (\kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \ldots)] \tag{2.8b}$$

It is sufficient to only use the first order term of equations 2.6. Using more terms causes numerical instability [12]. Thus, only κ_1 is approximated by the model. The final transformation is between position (X_d, Y_d) of point P_d on the camera's sensor plane and its position (X_f, Y_f) in the camera's frame buffer.

$$X_f = X_d + C_x \tag{2.9a}$$

$$Y_f = Y_d + C_y \tag{2.9b}$$

where C_x and C_y are the pixel coordinates of the principal point.

2.2 Calibration

The calibration is performed using the calibration software provided by R.G.Willson [17]. It needs as input a large number ($\gg 7$) of corresponding 3D world points and 2D image points. The reason to select many world-image point correspondences is to systematically eliminate inaccuracies in the image points selection. The approach presented in this thesis to choose accurate input image points is to intersect lines given by the grid of a checkerboard calibration target (figure 2.6). The lines to be intersected are obtained by fitting them to a set of user selected image points using least-squares. The precision of the lines is then

improved by fitting the lines to edges of the checkerboard. The checkerboard is a suitable standard pattern that makes it easy to select points. The desired input image points to the calibration software are the points of intersection of the lines. These point should coincide with the corner points of the checkerboard.

One problem arises due to distortion. Fitting lines to distorted image points would result in inaccurate intersection points. Therefore, it is necessary to first undistort the image of the checkerboard, i.e. to calculate an approximation of the distortion coefficient κ_1 (section 2.1.3) prior to approximate it by the calibration software. This method to approximate κ_1 is described in section 2.2.1. Having found κ_1, any image point can be undistorted by equations 2.8 where only the first term of κ_1 is used.

Lines can then be fitted to undistorted image points. Intersecting these lines results in undistorted intersection points. The desired input image points are found by applying distortion to the intersection points again, i.e. solving equations 2.8 for (X_d,Y_d). The process of acquiring these 2D input image points P_d is described more precisely in section 2.2.2. Finally, an overdetermined set of non-linear equations is obtained by setting equal equations 2.4 and 2.8.

$$X_d[1+(\kappa_1 r^2)] = f\frac{r_1X_w + r_2Y_w + r_3Z_w + T_x}{r_7X_w + r_8Y_w + r_9Z_w + T_z} + C_x \tag{2.10a}$$

$$Y_d[1+(\kappa_1 r^2)] = f\frac{r_4X_w + r_5Y_w + r_6Z_w + T_x}{r_7X_w + r_8Y_w + r_9Z_w + T_z} + C_y \tag{2.10b}$$

This set of equations is solved by the Levenberg-Marquardt least-squares method [42]. Having obtained a solution for $r_1,\ldots,r_9$ and T_x, T_y and T_z, a point in world coordinates $[X_w, Y_w, Z_w]$ can be projected onto a point in image coordinates $[X_f, Y_f]$ as described by the following equation using homogeneous coordinates.

$$\boxed{\begin{pmatrix} X_f \\ Y_f \\ 1 \end{pmatrix} = \begin{pmatrix} r_1 & r_2 & r_3 & T_x \\ r_4 & r_5 & r_6 & T_y \\ r_7 & r_8 & r_9 & T_z \end{pmatrix} \times \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}} \tag{2.11}$$

2.2.1 Approximation of κ_1

The method to approximate κ_1 is to identify curved lines in the distorted image that are straight in the real world [39]. These curved lines are then straightened during the undistortion process. This approach gives a first approximation of κ_1 before camera calibration is performed. A set L of lines is specified to undistorted an image. The number of lines in this set is $|L|$. One line $l_i \in L, i = 1,\ldots,|L|$ is given by manually choosing a set of arbitrary points P_i on line l_i in the distorted image of the checkerboard calibration target. The size of the ith point set P_i is m_i. For line $l_i \in L$, the line connecting its first with its last point is $l_i^{1m_i}$. Figure 2.4 shows a curved line l_i specified by nine arbitrary points $p_1,\ldots,p_9$. The orthogonal distance $d(p_k, l_i^{1m_i})$ from a point $p_k \in P_i, k = 1,\ldots,m_i$ to the line $l_i^{1m_i}$ is given by[1]

$$d_i(p_k, l_i^{1m_i}) = \frac{|\det((p_1 - p_{m_i})\ (p_{m_i} - p_k))|}{|p_1 - p_{m_i}|} \tag{2.12}$$

The algorithm outlined in figure 2.3 minimizes the total sum D of distances $d(p_k, l_i^{1m_i})$ to approximate κ_1. An image of the checkerboard which is undistorted using this approximation of κ_1 is shown in figure 2.6b.

$$D = \sum_{i=1}^{|L|}\sum_{k=1}^{|P|} d_i(p_k, l_i^{1m_i}) \tag{2.13}$$

[1] from MathWorld (`http://mathworld.wolfram.com/`)

```
while (step > β) do
   d1 = findDist(κ1 − step)
   d2 = findDist(κ1 + step)
   d3 = findDist(κ1)
   if ((d3 ≤ d1) and (d3 ≤ d2)) then {κ1 is good}
      step = step / 4
   else if (d1 ≤ d3) then {κ1 − step is better}
      κ1 = κ1 − step
   else {κ1 + step is better}
      κ1 = κ1 + step
   end if
end while
```

```
function findDist(κ1) : double
begin
D = 0.0
for all l_i ∈ L do
   for all p_k ∈ P_i do
      D = D + d(p_k, l_i^{1mi})
   end for
end for
return D
end
```

Figure 2.3: Algorithm to approximate κ_1 by minimizing the total sum of distances D. *The figure illustrates the algorithm to approximate κ_1 by minimizing the total sum D of distances $d(p_k, l_i^{1m_i})$. If d_3 is smallest, the current value for κ_1 is best. The step size is reduced until a minimum step size β is reached.*

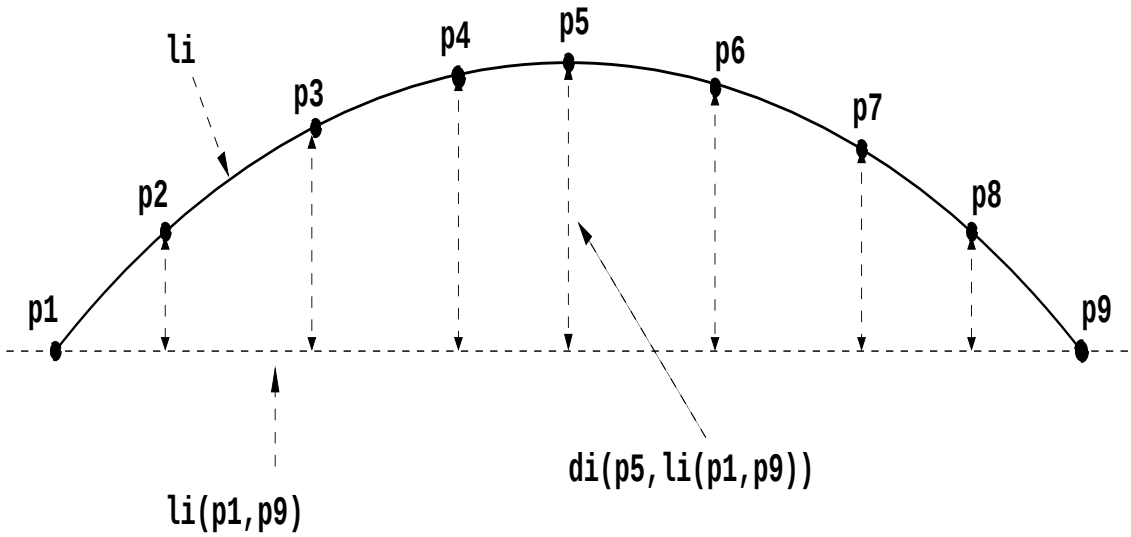


Figure 2.4: Straightening a curved line l_i in the distorted image. *The curved line l_i is specified by selecting a set P_i of points lying on l_i ($p_1, \ldots, p_9$) in the distorted image. Line l_i is straight in the real world. The algorithm shown in figure 2.3 to approximate κ_1 tries to minimize the total sum D of distances $d(p_k, l_i^{1m_i})$.*

2.2.2 Accurate Image Point Selection

The camera calibration software is very sensitive to inaccuracies in the input data. To select precise image points $P_d(X_d, Y_d)$ in sub-pixel accuracy the following approach is implemented (refer to figure 2.7). In step one, a user manually selects corner points in the image of the checkerboard calibration target (figure 2.6) as precise as possible. At the end of step one, a set L of $|L|$ lines $l_i, i = 1, \ldots, |L|$ is specified. In the second step, set L is used to approximate κ_1 as described in section 2.2.1. Then, in the third step, normal lines $n_{l_i}(p_n)$ at discrete positions $p_n, n = 1, \ldots, V$ on each line $l_i \in L$ are sampled. The position p_n is moved to the point of the largest image intensity change p_n^c along the normal line $n_{l_i}(p_n)$. A sobel filter is used to find changes in image intensity (compare figure 2.5). The fourth step is to undistort each point $p_n^c, n = 1, \ldots, V$ by using κ_1. A straight line is fitted to the set of undistorted points p_n^c by least-squares. The results of least-squares are the line coefficients c_0 and c_1.

Line $l_i \in |L|$ is then given by

$$l_i : Y = c_0 + c_1 X \tag{2.14}$$

This process of fitting lines to strong edges is repeated until the changes in the coefficients c_0 and c_1 are smaller than a threshold ε. The line coefficients will not change significantly any more, if the line lies close to an edge. The last step is to find the true image points $P_d(X_d, Y_d)$ by intersecting pairwise orthogonal lines in L. Two lines l_1 and l_2 of the form

$$l_1 : Y = c_{01} + c_{11} X_1 \tag{2.15a}$$

$$l_2 : Y = c_{02} + c_{12} X_2 \tag{2.15b}$$

are orthogonal if the angle θ between the two lines is π. Angle θ is given by[2]

$$\theta = \arccos\left(\frac{X_1 \bullet X_2}{|X_1| \cdot |X_2|}\right) \tag{2.16}$$

where $X_1 \bullet X_2$ is the dot product of X_1 and X_2. The point of intersection of l_1 and l_2 is found by bringing the lines to the form

$$l_1 : \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{11} \\ y_{11} \end{pmatrix} + s \begin{pmatrix} x_{d1} \\ y_{d1} \end{pmatrix} \tag{2.17a}$$

$$l_2 : \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{21} \\ y_{21} \end{pmatrix} + t \begin{pmatrix} x_{d2} \\ y_{d2} \end{pmatrix} \tag{2.17b}$$

and solving for s or t. The (distorted) intersection points P_d of all pairwise orthogonal lines in L are then used as input to the calibration software together with corresponding world points P_w. Steps one, three, four and five are shown in figure 2.8. The resulting homography of the ground plane is depicted in figure 2.8d. The green crosses shown in figure 2.8d are the image points P_d obtained by intersecting lines.

[2] from MathWorld (`http://mathworld.wolfram.com/`)

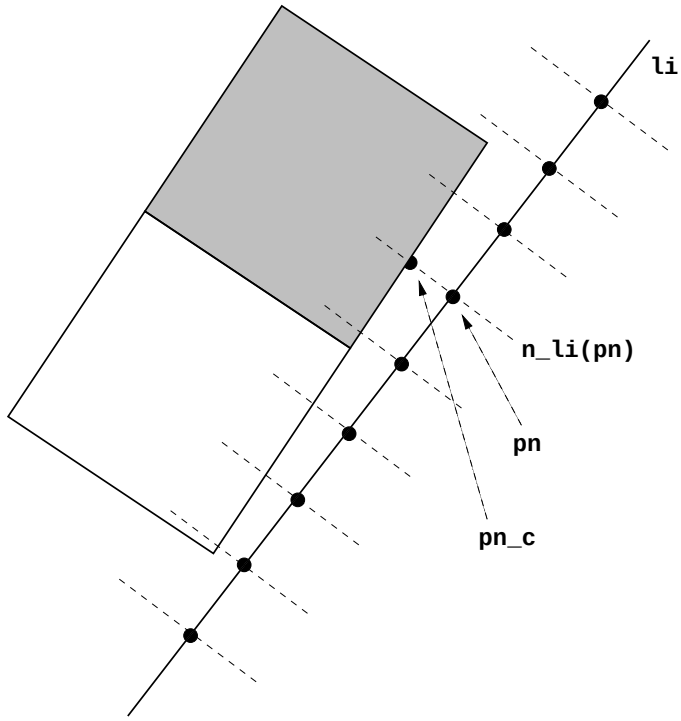


Figure 2.5: Line fitting. *Normal lines $n_{l_i}(p_n)$ at discrete positions $p_n, n = 1, \ldots, V$ on the line l_i are sampled along. The point p_n is moved to point p_n^c of the largest change in image intensity along the normal line $n_{l_i}(p_n)$. Points p_n^c are undistorted (section 2.2.1) and line l_i is corrected by fitting it to the new set of points p_n^c using least-squares.*

(a) (b)

Figure 2.6: Distorted and undistorted images of the checkerboard calibration target. *The figure shows a distorted (2.6a) and an undistorted (2.6b) image of the checkerboard calibration target. The target consists of 36 individual black and white floor tiles of size 50x50cm. Although the image on the right is undistorted, it still appears curved outward at its borders. This effect is caused due to the fact that the curved lines to undistort the image (see section 2.2.1) are not selected equally distributed across the image. Only the lines given by the grid of the checkerboard are selected. I.e. κ_1 is approximated to undistorted those lines best.*

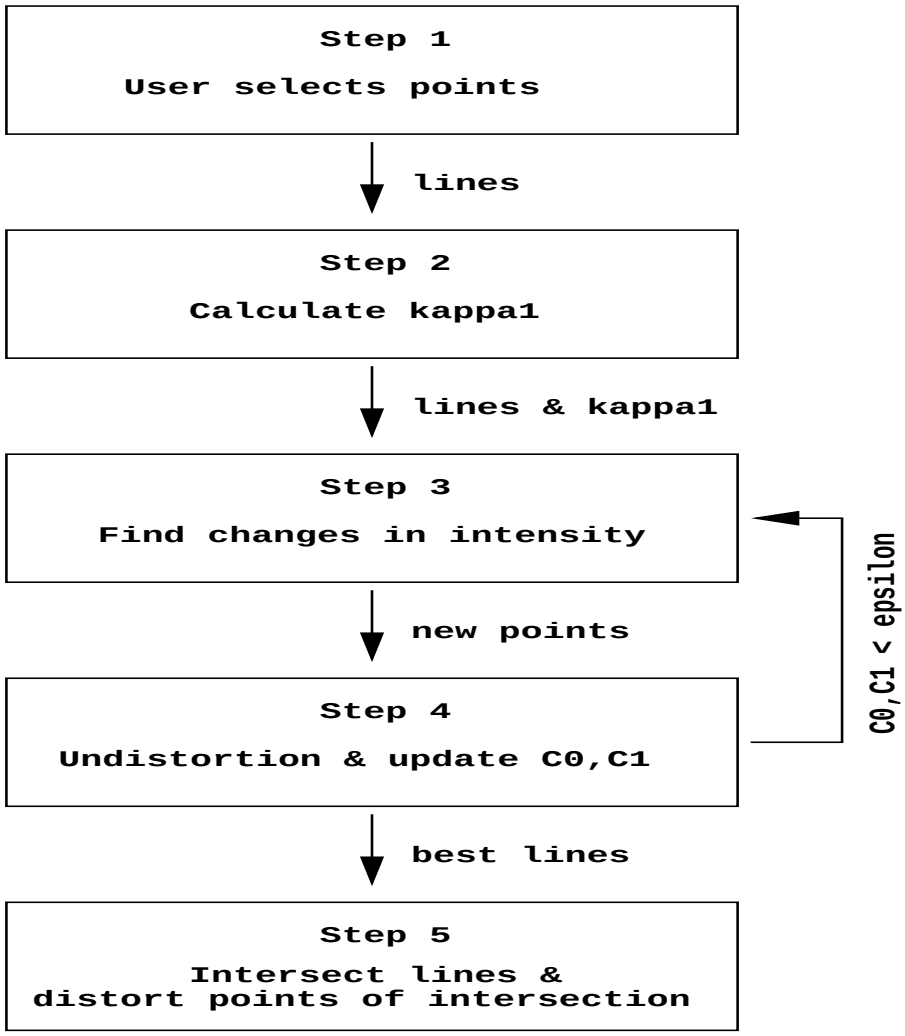
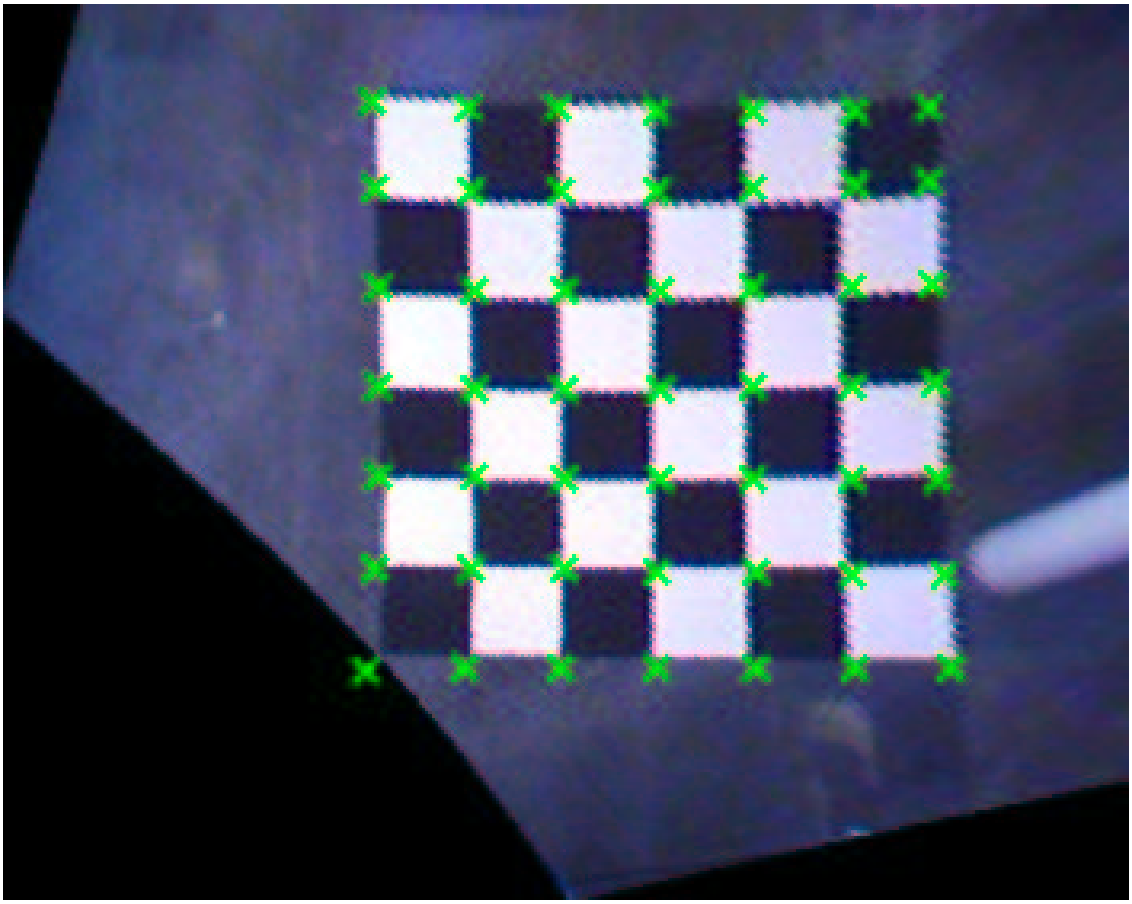


Figure 2.7: Five steps to find image points in sub-pixel accuracy. *Lines are specified by selecting points in the distorted image of the checkerboard calibration target. The distortion coefficient κ_1 is approximated (section 2.2.1). Normal lines $n_{l_i}(p_n)$ at discrete positions $p_n, n = 1, \ldots, V$ on each line $l_i \in L$ are sampled along. Point p_n is moved to the position p_n^c of the largest change of image intensity (see figure 2.5). A sobel filter is used therefore. The modified points are undistorted. A straight line is fitted to the set of undistorted data points p_n^c using least-squares. The results are the line parameters c_0 and c_1. The process of fitting and correcting lines is repeated until c_0 and c_1 are stable. Finally, all pairwise orthogonal lines in L are intersected and again distorted to find true image points in sub-pixel accuracy.*

(a)

(b)

(c)



(d)

Figure 2.8: Steps one, three, four and five to find image points and the homography of the ground plane. *Figure 2.8a corresponds to step one of figure 2.7 where a user specifies lines by selecting image points. Figure 2.8b corresponds to steps three and four and illustrates the grid lines fitted to edges of the checkerboard. The short red lines are normal lines. The fifth step is shown in figure 2.8c where the fitted grid lines are intersected. Finally, in figure 2.8d a homography of the ground plane* ($Z_w = 0$) *is depicted. The green crosses mark the projected image points* P_d *of image 2.8c.*

Chapter 3

Description of the Reading People Tracker

The *Reading People Tracker is* is designed to track people for visual surveillance systems. It is originally based on the *Leeds People Tracker* [4, 5]. The tracker consists of four co-operating modules:

- The **Motion Detector**,
- the **Region Tracker**,
- the **Human Feature Detector** (Head Detector)
- and the **Active Shape Tracker**.

Using four co-operating modules results in more stable and reliable overall tracking performance, i.e. limitations of one module are compensated by other modules. In contrast to other tracking systems like the *PCCV-Tracker* [1], the Reading People Tracker implements an *Active Shape Tracker*. This tracker uses a 2D appearance model of human beings. This *Active Shape Model* is explained in section 3.4.1. Shape variations are extracted from a large training set by using principal component analysis (PCA). Further differences exist in the implemented background model of the Reading People Tracker. The background is modeled as one image with no people in it. The image is updated by a temporal median filter. The Region Tracker tracks all extracted moving regions from the Motion Detector. Region merging and splitting (section 3.2.1) is applied to tracked regions. In addition, the Reading People Tracker implements a Human Feature Detector which is actually a head detector. The Active Shape Tracker benefits from the head detector because people are faster identified by their head position. This means shapes of multiple people are initialized preciser and tracked robuster. Figure 3.1 illustrates the four tracking modules of the Reading People Tracker.

The following terms are used throughout this chapter. Figure 3.2 shows their relations.

- *Prediction:* Once, a person is successfully identified as moving in frame t, its position is predicted for frame $t + 1$. The Region Tracker as well as the Active Shape Tracker perform predictions.
- *Measurement:* A predicted region or shape for frame t is tried to match with local image features of frame t. If successful, the prediction is moved to the measurements and a new prediction for the measurement for frame $t + 1$ is performed.
- *Hypotheses:* Hypotheses are measurements that need to be approved first. The Region Tracker generates hypotheses by region merging and splitting (section 3.2.1). The Human Feature Detector generates hypotheses by searching for possible head positions in regions.

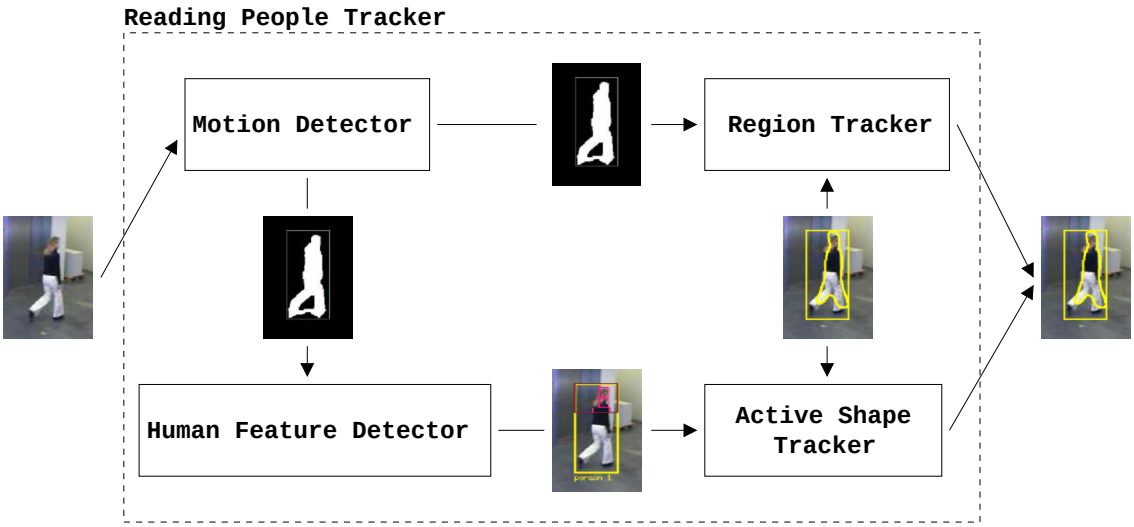


Figure 3.1: The four tracking modules of the Reading People Tracker. *The figure illustrates the four tracking modules of the Reading People Tracker. The Motion Detector computes the binary motion image. Extracted regions by the Motion Detector are passed on to the Region Tracker and the Human Feature Detector (head detector). The Region Tracker tracks regions from the Motion Detector over time. The Human Feature Detector extracts possible head positions. Tracked regions and possible head position are inputs to the Active Shape Tracker. Together, the four tracking modules provide better tracking performance than a single module would do.*

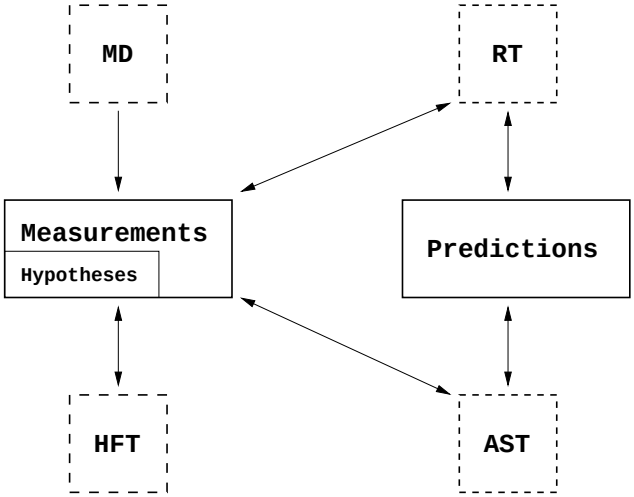


Figure 3.2: Relation between measurements and predictions and the four tracking modules. *The figure shows the relation between measurements and predictions and the four tracking modules Predictions for frame t are tried to match with image measurements of frame t. The Motion Detector (MD) generates measurements by background-foreground segmentation. The Human Feature Detector (HFT) takes a measurement (a region) to search for possible head positions in it. The Region Tracker (RT) as well as the Active Shape Tracker take both measurements and predictions and try to match them. Hypotheses are a subset of measurements which are generated by the RT and HFT.*

3.1 The Motion Detector

The Motion Detector of the Reading People Tracker maintains a model of the background. The background is modeled as one video image with no persons in it. The background image is updated by a temporal pixelwise median filter of the video images. The Motion Detector uses the background model to identify moving blobs in a video frame. The Motion Detector deals with four different kinds of images. Figure 3.3 shows an example of each image.

- **The video image** corresponds to one video frame of a video sequence. The video image is to be examined by the tracking algorithm.
- **The background** image models the background. It is a video image with initially no people in it. The background image is temporally updated using a median filter.
- **The difference image** is obtained by pixelwise differencing the video image and the background image. The difference image is used by the Active Shape Tracker to search for edges.
- **The motion image** results by thresholding the difference image.

Video Image | Background Image | Difference Image | Motion Image

Figure 3.3: Image kinds involved in motion detection. *The tracking algorithm examines the video image. The background is modeled by a temporally updated video image with initially no people in it. The difference image is the pixelwise absolute difference of the video image and the background image. The motion image is obtained by thresholding the difference image.*

The following parameters affect the behavior of the Motion Detector. Section 4.4 evaluates the Reading People Tracker with respect to this parameters.

- `DETECT_DIFF_THRESH`: The motion image result of thresholding the difference image. This parameter corresponds to this threshold. It decides for each pixel of the difference image if it is considered as foreground (moving) or as background (static).
- `MIN_REGION_SIZE`, `MAX_REGION_SIZE`: These parameters define the minimum and maximum size of regions to be extracted by the Motion Detector. The size of a region is given by its area (in pixels) normalized by the area of the video image. The parameters are computed by using a statistical model which is described in section 4.4.2.
- `MIN_HEIGHT_TO_WIDTH`, `MAX_HEIGHT_TO_WIDTH`: These parameters define the minimum and maximum ratio of height to width of regions to be extracted from the motion image. The same statistical model (section 4.4.2) is used to calculate these parameters.

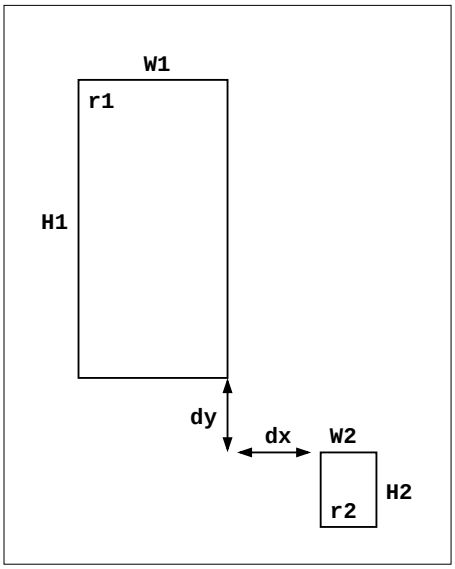


Figure 3.4: Region merge technique. *The figure illustrates two regions r_1 and r_2. Their relative distance in x and y coordinates is d_x and d_y. Two regions are merged together if the ratio of their relative distance with their sizes is less than* `REGION_MERGE_THRESH`.

- `REGION_MERGE_THRESH`: This threshold defines if two regions r_1 and r_2 are merged together (figure 3.4). The distance measure D_{r_1,r_2} relates the relative distance of the regions to their sizes. Two regions are merged if D_{r_1,r_2} is smaller than this threshold.

$$D_{r_1,r_2} = \frac{dx + dy}{max(W_{r_1}, H_{r_1}) + max(W_{r_2}, H_{r_2}) + 1} \tag{3.1}$$

- `MEDIAN_FILTER_MOTION_IMAGE`: This parameter defines if a spacial median filter is applied to the motion image.
- `BLUR_MOTION`: Specifies if the video image and the background image are blurred during motion detection.
- `MOTION_IMAGE_DILATION`: This parameter specifies if dilation is used to improve the quality of the motion image. If dilation is performed, parameter `GAP_SIZE` defines size of the dilation mask.

The output of the Motion Detector are the regions that contain foreground objects (moving blobs). Extracted regions are passed on to the Region Tracker (section 3.2) and the Human Feature Detector (section 3.3).

3.2 The Region Tracker

The Region Tracker tracks regions extracted from the Motion Detector over time. Predictions of these regions are generated using a first order motion model. It is assumed that people move with constant velocity. Bounding boxes of detected foreground regions by the Motion Detector and bounding boxes of detected person shapes by the Active Shape Tracker are both inputs to the Region Tracker (figure 3.1). These inputs are both measurements because they are obtained by either background-foreground segmentation or local edge search.

The Region Tracker matches predictions with measurements. Region splitting and merging is used (section 3.2.1). If a prediction can be matched successfully to a measurement, the prediction is replaced by the measurement and a new prediction performed. If the Region Tracker fails to match predictions with measurements from the Motion Detector, it consults the output of the Active Shape Tracker. If the Active Shape Tracker successfully tracks the person, the bounding box of the Active Shape representing the person is taken by the Region Tracker as a new hypothesis for the person's size and position.

3.2.1 Region Merging and Splitting

Region splitting and merging is useful in the situation where two persons cross each other (compare figure 3.5). The two persons are tracked correctly as long as they are apart (figure 3.5a). As soon as one person occludes the other, the Motion Detector extracts them as one moving region (figure 3.5b). To overcome this problem the Region Tracker always merges predicted regions that are close to each other. This new, larger prediction of a region is tried to match with measurements from the Motion Detector. If the match is successful, two new "synthetic" measurements are created by splitting the large region into two (figure 3.5c). This is reasonable because it is assumed that the measurement from the Motion Detector now contains two persons.

(a) (b) (c)

Figure 3.5: Region merging and splitting. *The figure shows how region merging and splitting overcome problems during motion detection. Two persons are extracted separately as long as they are far apart (figure 3.5a). When they cross each other, the Motion Detector extracts them as one moving region (figure 3.5b). The Region Tracker always merges predictions that are close to each other. Possibly, this larger predictions of regions can be matched with measurements. This is the case in figure 3.5c where the combined prediction is again split into two new (synthetic) measurements.*

3.3 The Human Feature Detector (Head Detector)

The head detector examines the upper part of a region for moving heads. It looks for vertical peaks within a region in the motion image. Figure 3.6 illustrates the algorithm for the video image. Region R is examined column-by-column. Each column is scanned top-to-bottom until a moving (foreground) pixel is found. The distance d of this pixel to the bottom of R is stored in vector v. Figure 3.6 shows vertical columns that are scanned for moving pixels. Distance d is shown for one column. Vector v is then searched for intervals v_i whose values do not differ significantly. Interval v_i is valid if the values before and after v_i are significantly lower than the values inside v_i. The x-position of a possible head is defined as the center of v_i. The y-position is the average of the highest and the lowest value in a small neighborhood outside of v_i. The resulting head position corresponding to v_i is depicted in figure 3.6.

Figure 3.6: Algorithm to detect possible head positions. *The figure illustrates region R which is searched for possible head positions. The region is examined column-by-column. Each column is scanned top-to-bottom for moving (foreground) pixels. The distance d from a moving pixel to the bottom of the region is stored in vector v. Vector v is search for intervals v_i of similar values. The center of v_i is the x-position of a possible head. The y-position is the average of the lowest and highest value in a small neighborhood outside of v_i. The small square depicts the resulting head position.*

3.4 The Active Shape Tracker

The Active Shape Tracker implements an Active Shape Model (section 3.4.1) to represent the 2D appearance of human beings. A person's outlines are tried to match with a model instance. Regions tracked by the Region Tracker as well as possible head positions found by the Human Feature Detector are inputs to the Active Shape Tracker. Each region is examined for walking people, i.e. local edge detection is performed within each region. Figure 3.7 explains the process of shape fitting. In addition, figure 3.8 illustrates local edge detection for two hypotheses of the same person as accomplished by the Reading People Tracker. The first measurement is given by the region of the Region Tracker (figures 3.8a,3.8b). The second hypotheses for the person comes from a possible head position detected by the Human Feature Detector (figures 3.8c,3.8d). Normal lines to the predicted Active Shape at a number of sampling points are sampled along to search for strong local image edges. The Active Shape is corrected by fitting it to the new position of the person's outlines. Once, a new Active Shape is calculated that fits the person's outlines best for frame t, the Active Shape Tracker predicts its position for frame $t + 1$ using a Kalman Filter (see section 3.4.1).

3.4.1 Active Shape Model

B-spline representation. Active Shapes are represented as parametric B-spline curves. A shape is to be approximated with a closed spline

$$P(s) = (P_x(s), P_y(s)) \tag{3.2}$$

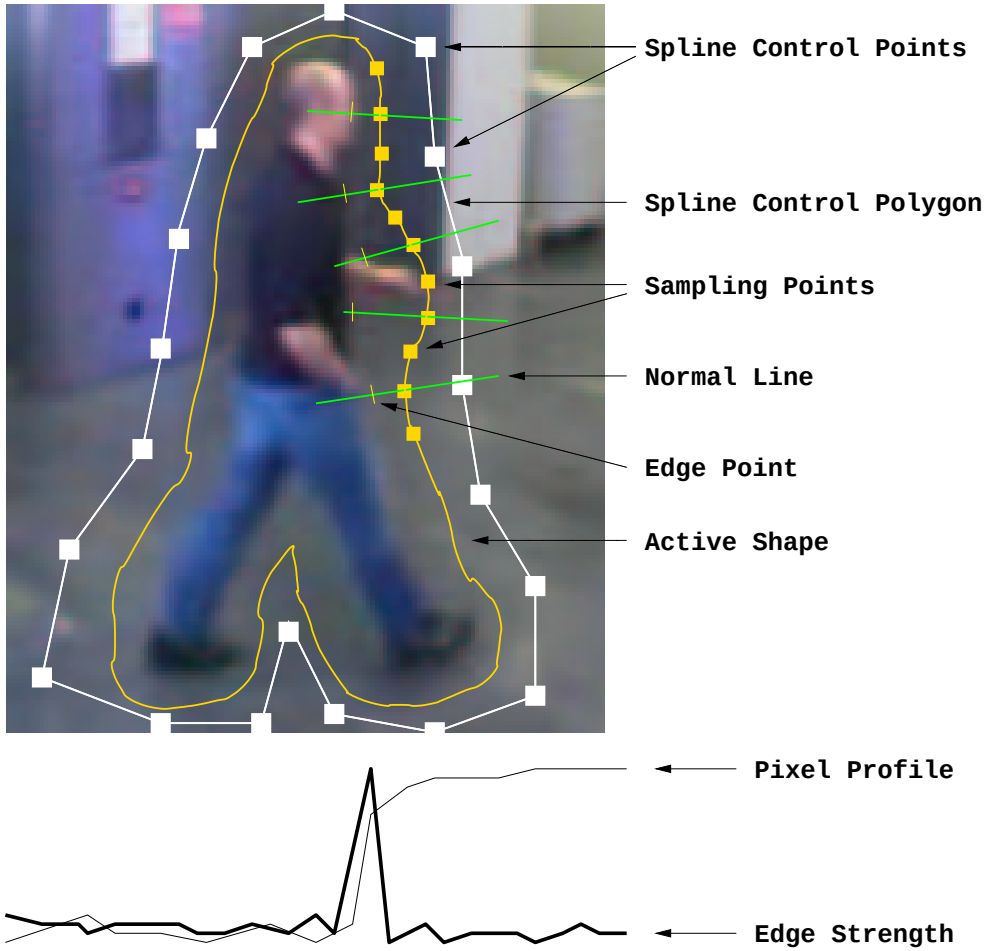


Figure 3.7: The process of shape fitting. *An Active Shape is mathematically represented as closed parameterized B-spline curve. A B-spline is given by a set of basis functions and spline control points. The appearance of an Active Shape is changed if the control points are modified. The figure shows one prediction of an Active Shape for the depicted person. The prediction is corrected by performing local edge search. First, a number of sampling points (usually larger than the number of spline control points) are defined on the Active Shape. Normal lines to the shape at these sampling points are sampled along for large changes in image intensity. One edge point is labeled in the figure and the corresponding pixel profile along the normal line depicted. By convoluting the pixel profile with a sobel mask for instance, edge strengths are detected. Having performed local edge search for each sampling point, the detected edge points define the corrected position of the Active Shape. The Active Shape is fitted to the person's outlines. By mapping the corrected sampling points (edge points) to corrected spline control points, a new Active Shape is calculated that fits best to the detected edge points. This means, the new shape fits best to the new position of the person.*

where $s \in [0, 1]$ is a parameter that increases as the shape is traversed and $P_x(s)$ and $P_y(s)$ are concrete spline functions of s. The name "B-spline" comes from the fact that the spline functions are build as a weighted sum of basis functions $B_{k,d}(s)$ whose degree is d (highest power of x). $P(s)$ is defined as

$$P(s) = \sum_{k=0}^{N-1} Q_k B_{k,d}(s) \tag{3.3}$$

(a) (b)

(c) (d)

Figure 3.8: Local edge search. *The figure shows the local edge search that is performed for each tracked region to find person outlines. The difference image on the left is used to identify edges. Results are show on both the difference image and the video image on the right. Two hypotheses are examined in this case. The first is given by the region passed from the Region Tracker (figures 3.8a,3.8b). The second hypothesis comes from a possible head position detected by the Human Feature Detector (figures 3.8c,3.8d). Normal lines to the predicted Active Shape of the person at a number of sampling points are sample along to search for strong edges. Green dots mark the starting points of the search, red dots mark detected edges.*

where

$$Q_k = (R_k, S_k) \tag{3.4}$$

are the weights also known as control points (or control polygon if represented as a vector). Basis functions are combined linearly to form the spline functions $P_x(s)$ and $P_y(s)$. Figure 3.9 shows four quadratic B-spline basis functions. In this simple case $B_{0,2}(s)$ is non-zero for $0 \leq s < 3$.

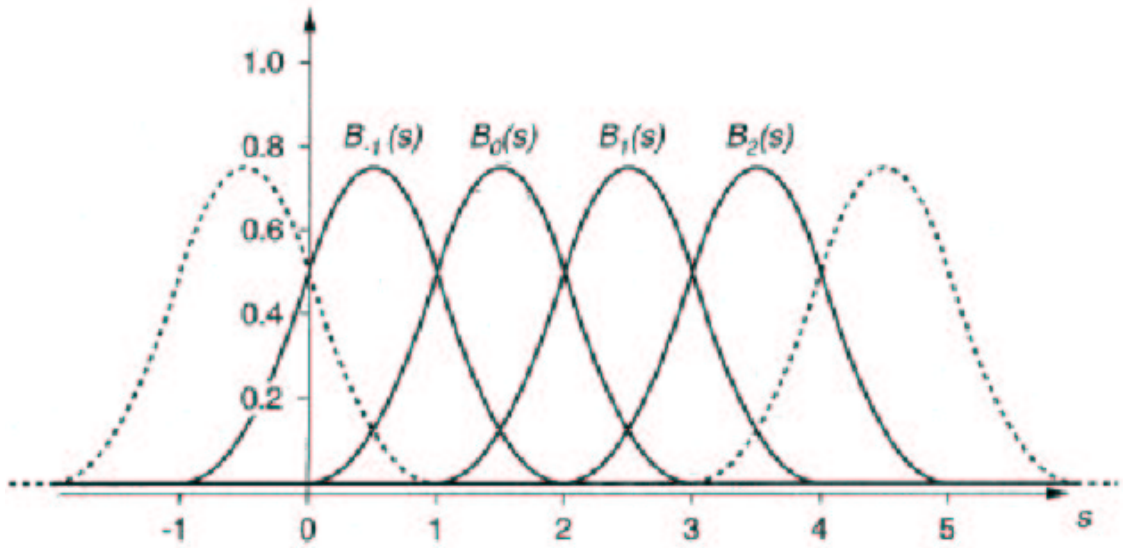


Figure 3.9: Four quadratic B-spline basis functions. *The figure illustrates four quadratic B-spline basis functions $B_{-1,2}(s), \ldots, B_{2,2}(s)$ where each basis functions is a translated copy of the previous one. These basis functions are combined linearly to form the spline functions $P_x(s)$ and $P_y(s)$.*

The other basis functions are translated copies of the previous one.

$$B_{k,2}(s) = B_{0,2}(s-k) \tag{3.5}$$

In the case of Active Shapes, the corresponding B-splines $P(s)$ are closed curves. Parameter s is defined over the interval $[0, N]$ and treated as periodic (N is the number of basis functions and control points). Figure 3.10 illustrates a quadratic, closed splines curve. The curve in figure 3.10a is a smooth approximation to its control polygon shown in figure 3.10b. In the following, an Active Shape is defined by its control polygon or *shape vector* x given as

$$x = (R_0, S_0, R_1, S_1, \ldots, R_{N-1}, S_{N-1}) \tag{3.6}$$

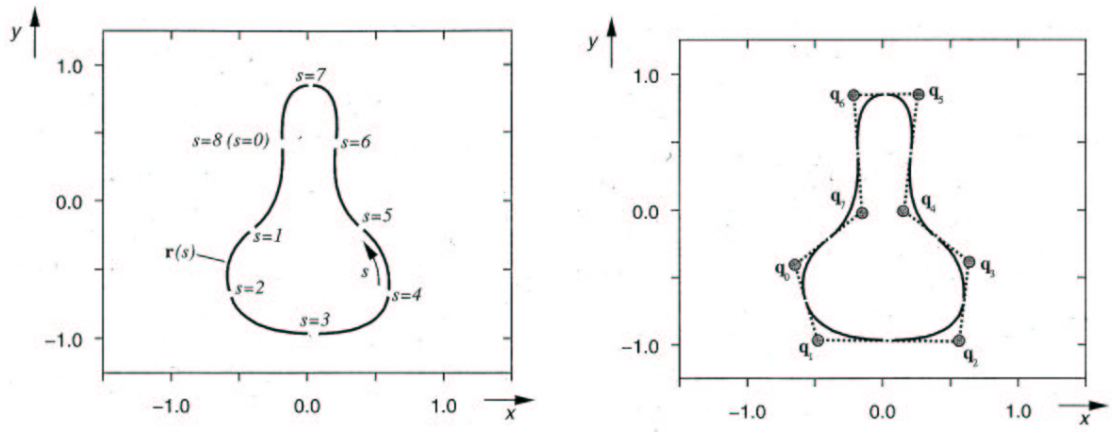
Generation of the Active Model from the training set. The Active Model is generated using a large set of size M training images that show people in different poses. A number of points are selected on each person's contour. The result is a vector w of contour (or boundary) points for each image in the training set.

$$w = (x_1, y_1, x_2, y_2, \ldots, x_n, y_n) \tag{3.7}$$

The required spline $P(s) = (P_x(s), P_y(s))$ approximating the boundary points w minimizes the least-squares error function

$$Error(Q_0, \ldots, Q_{N-1}) = \sum_{i=0}^{n-1} (P_x(s_i) - x_i)^2 + (P_y(s_i) - y_i)^2 \tag{3.8}$$

The control points $Q_0, \ldots, Q_{N-1}$ are the unknown quantities to be determined. The shape vector x is obtained by using standard least-squares methods [23, 42].



(a) (b)

Figure 3.10: Closed B-spline curve. *The figure illustrates a closed B-spline curve. The spline curve in figure 3.10a is smoothly approximated by its control points* $q_0, \ldots, q_7$ *shown in figure 3.10b. Parameter* s *is defined over the interval* $[0, 8]$ *and is treated as periodic.*

$$x = (R_0, S_0, R_1, S_1, \ldots, R_{N-1}, S_{N-1}) \tag{3.9}$$

The training set now consists of M shape vectors x. All shape vectors are aligned to a normalized frame of reference [6]. Capturing statistical information from the set of shape vectors involves calculating the mean shape

$$\overline{x} = \frac{1}{M} \sum_{i=1}^{M} x_i \tag{3.10}$$

and the standard deviation of each shape x from the mean shape $\overline{x}$.

$$dx_i = x_i - \overline{x} \tag{3.11}$$

Principal component analysis (PCA) of the covariance matrix C is used to find the principle axis of the $2N$-dimensional point cloud of all shape control points. The advantage of using PCA for this particular case is to constitute the essence of all shape vectors x. This means, to extract and compress the information of all shape vectors to explain the most common shape variations (principal components) in the training set. The covariance matrix C is calculated as

$$C = \frac{1}{M} \sum_{i=1}^{M} dx_i \, dx_i^T \tag{3.12}$$

Ordering the eigenvectors $p_1, p_2, p_3, \ldots, p_{2N}$ in the order of descending eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{2N}$ (obtained by PCA), an ordered orthonormal basis with the first t eigenvectors having the largest variance of dx_i can be constructed. This makes sense, because most of the shape variations can be explained by less than $2N$ modes. To choose only the first t modes of variation, a threshold ξ is introduced. The t modes of variation are supposed to explain $\xi \times 100\%$ of shape variation.

$$\sum_{i=1}^{t} \lambda_i = \xi \times \sum_{i=1}^{2N} \lambda_i \tag{3.13}$$

The matrix P of the first t eigenvectors is given as

$$P = [p_1\ p_2\ p_3 \ldots p_t] \tag{3.14}$$

A new shape instance can be generated by adding a weighted linear combination to the mean shape.

$$\boxed{v = \overline{x} + Pb} \tag{3.15}$$

where b is a vector of weights, one for each eigenvector.

Verifying new shape instances. Having build the Active Shape Model, the mean shape $\overline{x}$ and the matrix P of the first t modes of variation are given. A new shape instance v defined by its number of control points can be generated in the same way as from the training set or by other methods to extract boundary points of people's outlines from images (by segmentation for instance). Vector b of weights that corresponds to the new shape instance v is given by

$$\boxed{b = P^T(v - \overline{x})} \tag{3.16}$$

To restrict the shape v to take free boundaries, limitations on the weights b are imposed. The weights are restricted to lie within a hyper-ellipsoid using the Mahalanobis distance D_m [8].

$$D_m^2 = \sum_{i=1}^{t} \frac{b_k^2}{\lambda_k} \leq D_{max}^2 \tag{3.17}$$

If the Mahalanobis distance D_m is bigger than D_{max}, the weights b are rescaled to lie on the hyper-ellipsoid.

$$b = b \cdot \frac{D_{max}}{D_m} \tag{3.18}$$

Cootes et al. [8] proposes a value of 3.0 for D_{max}. Otherwise, if D_m is smaller than D_{max}, the weights can be projected back using equation 3.15 to obtain a "component filter" version v_f of spline v (figure 3.11).

Predicting Active Shapes. The position of a measured shape instance $\hat{v}(k)$ in frame k is predicted for frame $k+1$ using a Kalman filter [24, 25]. The shape $\hat{v}(k)$ is obtained by shape fitting as described in section 3.4 and illustrated in figure 3.7. The first step is to predict the coarse prediction $\tilde{v}(k)$ which involves the predictions for the last two frames $p_{v(k-1)}$ and $p_{v(k-2)}$.

$$\tilde{p}_{v(k)} = p_{v(k-1)} + (p_{v(k-1)} - p_{v(k-2)}) \tag{3.19}$$

In the second step, the coarse prediction $\tilde{p}_{v(k)}$ is corrected using the measured shape instance $\hat{v}(k)$.

$$\boxed{p_{v(k)} = \tilde{p}_{v(k)} + K(\hat{v}(k) - \tilde{p}_{v(k)})} \tag{3.20}$$

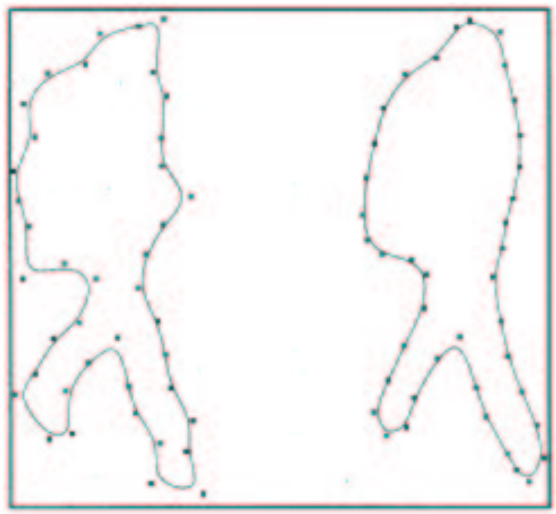


Figure 3.11: Component filtered shape. *The figure shows a shape v on the left that was obtained by $v = \overline{x} + Pb$ using the first t modes of shape variation. The weights $b = P^T(v - \overline{x})$ are rescaled (b_f) to fulfill equation 3.17. The resulting (smoothed) spline v_f on the right is obtained by $v_f = \overline{x} + Pb_f$.*

where K is the *Kalman gain*. K is defined as the ratio of the relative predictions $\triangle p_{v(k)} = p_{v(k-1)} - p_{v(k-2)}$ with the variance of the relative measurements $\triangle \hat{v}(k) = \hat{v}(k-1) - \hat{v}(k-2)$ for the last two frames [25].

$$K = \frac{\sigma^2_{\triangle p_{v(k)})}}{\sigma^2_{\triangle \hat{v}(k)}} \tag{3.21}$$

Chapter 4

Evaluation of the Reading People Tracker

When evaluating tracking algorithms and in general computer vision algorithms a few crucial questions need to be answered first. For instance, which kind of evaluation is performed. A qualitative evaluation strategy assesses the output of the tracker according to subjective criteria. Criteria, as are all subjects correctly identified by the tracker or, for instance, does it often loose track of detected subjects. A quantitative evaluation in contrast, makes use of evaluation performance metrics (section 4.1) to assess the tracking output according to objective criteria. How well, for example, does the tracking algorithm identify subjects. Precision an recall are examples of performance metrics. Both, qualitative and quantitative evaluations are performed to evaluate the Reading People Tracker. An evaluation itself can be represented by "evaluation meta data". For this thesis, this meta data is defined as

1. the *type* of evaluation.
2. the set of parameters configuring the tracking algorithm.
3. the video sequences on which the tracker is evaluated.
4. the *ground truth* - a set of objects found in a video sequence determined a priori to be correct.
5. evaluation parameter settings.

The type of accomplished evaluations is always "framewise" in this thesis. This means, tracking output and ground truth are compared on a frame-by-frame basis. The type itself can be parameterized as well and is described in section 4.3. Having specified this kind of meta data, especially tracking parameters, video sequences and ground truth, it is possible to compare different tracking algorithms with the same data using similar settings.

The video sequences which were used to evaluate the Reading People Tracker are described in section 4.2. This sequences were annotated using `ViPER`[1]. Authoring ground truth is done by drawing rectangular bounding boxes as close as possible around each person in the video sequence whose body has an estimated visibility of at least 75%. Any shadows or reflections against walls caused by illumination were ignored. Bounding boxes were drawn around person outlines solely. This reflects that any people tracker basically would have to deal with changes in illumination, indoor and outdoor, as this happens always and everywhere in the real world. This does not imply that a tracker which does not explicitly deal with shadows or reflections generally produces bad results in object detection. However, its tracking output will probably

[1] `http://lamp.cfar.umd.edu/media/research/viper/`

be less precise. Furthermore, people appearing in groups are always considered as separate persons, hence annotated separately as long as their estimated body visibility is at least 75%.

The `ViPER` package provides scripts for performing the actual evaluation. In principle, two files of which one holds the ground truth of a video sequence and the other one the tracking output are compared according to different performance metrics. Figure 4.1 depicts the process of evaluation. The performance metrics are explained in section 4.1.

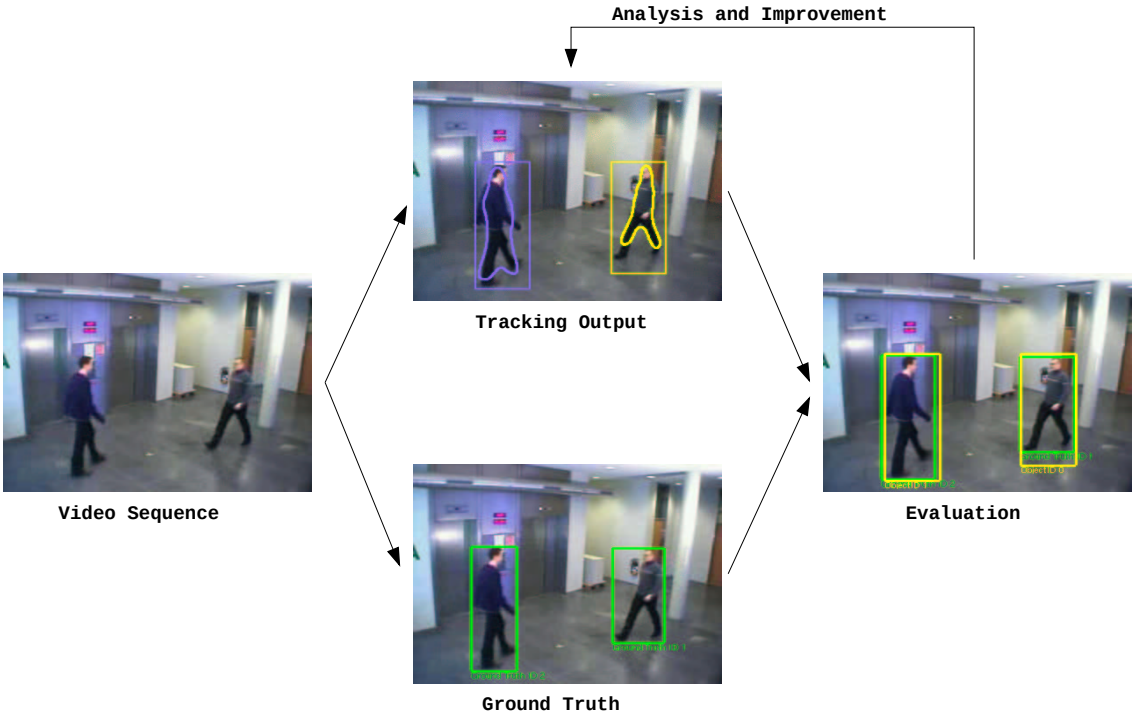


Figure 4.1: Evaluation and benchmarking scheme. *The evaluation of the Reading People Tracker is performed by comparing ground truth and tracking output of a video sequence on a frame-by-frame basis.* `ViPER` *is used to calculate the performance metrics. By analyzing evaluation results the tracking algorithm can be improved to achieve better tracking performance.*

4.1 Evaluation Performance Metrics

This section explains the performance metrics used to quantitatively evaluate the Reading People Tracker. Let $G^{(t)}$ be the set of ground truth objects in a single frame t. Then $N_{G^{(t)}} = |N_{G^{(t)}}|$ is the number of elements in this set of ground truth objects. $D^{(t)}$ is the set of detected tracking output objects and $N_{D^{(t)}} = |N_{D^{(t)}}|$ the number of objects in this set. $N_f \leq N$ is the number of frames that contain ground truth objects. N is the total number of frames of the evaluated video sequence. $G_i^{(t)}, i = 1, \ldots, |N_{G^{(t)}}|$ is one object of $G^{(t)}$, whereas $|G_i^{(t)}|$ is the spatial union of pixels that are covered by $G_i^{(t)}$. The same holds for tracking output objects. $D_i^{(t)}, i = 1, \ldots, |N_{D^{(t)}}|$ is an object of set $D^{(t)}$, $|D_i^{(t)}|$ its spatial union of pixels. The spatial union of all pixels of all objects $G_i^{(t)} \in G^{(t)}$ is defined as

$$U_{G^{(t)}} = \bigcup_{i=1}^{N_{G^{(t)}}} |G_i^{(t)}| \tag{4.1}$$

The value $|U_{G^{(t)}}|$ is the total number of pixels in $U_{G^{(t)}}$, i.e the total number of pixels in the ground

truth of frame t. The spatial union of tracking output objects $D_i^{(t)} \in D^{(t)}$ is

$$U_{D^{(t)}} = \bigcup_{i=1}^{N_{D^{(t)}}} |D_i^{(t)}| \tag{4.2}$$

The value $|U_{D^{(t)}}|$ is the total number of pixels in $U_{D^{(t)}}$ or the total number of pixels in the tracking output. In addition, $U_{I^{(t)}}$ is the spatial union of the total number of pixels in t. The total number of pixels S of a video sequence of N frames is then defined as

$$S = \sum_{t=1}^{N} |U_{I^{(t)}}| \tag{4.3}$$

4.1.1 Pixel-based Metrics

This metrics do not treat a video frame as a collection of individual objects but as a set of individual pixels. The following pixel-based metrics are defined:

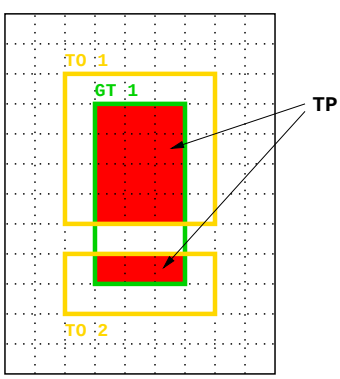
1. **True Positives**

   The number of matched pixels of video frame t is given by the intersection

   $$TP = |U_{G^{(t)}} \cap U_{D^{(t)}}| \tag{4.4}$$

   True Positives are those pixels the tracking algorithm successfully identified as ground truth pixels.

   Example:

   The number of True Positives is 15.

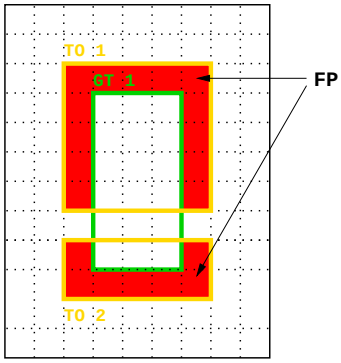
2. **False Positives**

   False Positives (or false pixels) are those pixels the tracking algorithm mistakenly classified as ground truth pixels.

   $$FP = |\overline{U_{G^{(t)}}} \cap U_{D^{(t)}}| \tag{4.5}$$

   where $\overline{U_{G^{(t)}}}$ is the difference $U_{I^{(t)}} - U_{G^{(t)}}$.

Example:



The number of False Positives is 20.

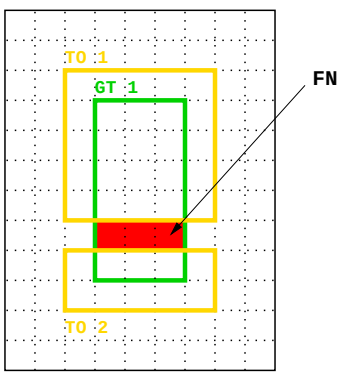
3. **False Negatives**

   False Negatives (or missed pixels), are all pixels which are not detected, although labeled in ground truth. Thus, its number is given by

$$FN = |U_{G^{(t)}} \cap \overline{U_{D^{(t)}}}| \tag{4.6}$$

   where $\overline{U_{D^{(t)}}}$ is the difference $U_{I^{(t)}} - U_{D^{(t)}}$.

   Example:



The number of False Negatives is 3.

4. **Pixel-based Precision**

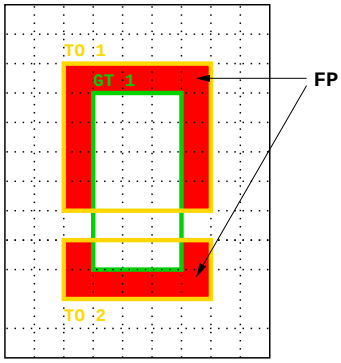
   Pixel-based precision of video frame t is defined as 1 minus the ratio of False Positives with the total number of pixels in $U_{D^{(t)}}$.

$$\boxed{Precision_{pixel-based}(t) = \begin{cases} undefined & if\, U_{D^{(t)}} = \emptyset \\ 1 - \frac{FP}{|U_{D^{(t)}}|} & otherwise \end{cases}} \tag{4.7}$$

   Then weighted average pixel-based precision of all tracking output objects of all video frames N_f that contain ground truth objects is given by

$$OverallPrecision_{pixel-based}(t) = \begin{cases} undefined & if\ \sum_{t=1}^{N_f} |U_{D^{(t)}}| = 0 \\ \frac{\sum_{t=1}^{N_f} |U_{D^{(t)}}| \times Precision_{pixel-based}(t)}{\sum_{t=1}^{N_f} |U_{D^{(t)}}|} & otherwise \end{cases} \tag{4.8}$$

Example:



Pixel-based precision is defined as 1 minus the ratio of False Positives with total number of tracking output pixels.

$$Precision_{pixel-based}(t) = 1 - \frac{20}{35} = 1 - \frac{4}{7} = 0.429$$

Pixel-based precision measures how well the tracking algorithm minimizes false alarms.

5. **Pixel-based Recall**

Pixel-based recall is defined in a similar way. For a single frame t, it is the ratio of True Positives with the total number of pixels in $U_{G^{(t)}}$.

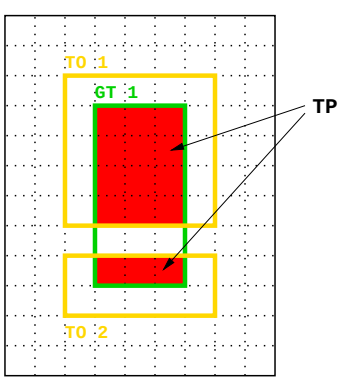
$$\boxed{Recall_{pixel-based}(t) = \begin{cases} undefined & if\, U_{G^{(t)}} = \emptyset \\ \frac{TP}{|U_{G^{(t)}}|} & otherwise \end{cases}} \tag{4.9}$$

Overall pixel-based recall of a video sequence is then defined as the weighted average pixel-based recall of all frames N_f that contain ground truth objects.

$$OverallRecall_{pixel-based}(t) = \begin{cases} undefined & if\, \sum_{t=1}^{N_f} |U_{G^{(t)}}| = 0 \\ \frac{\sum_{t=1}^{N_f} |U_{G^{(t)}}| \times Recall_{pixel-based}(t)}{\sum_{t=1}^{N_f} |U_{G^{(t)}}|} & otherwise \end{cases} \tag{4.10}$$

Pixel-based recall measures how well the tracking algorithm output covers ground truth.

Example:



Pixel-based recall is the ratio of True Positives with the total number of ground truth pixels.

$$Recall_{pixel-based}(t) = \frac{15}{18} = \frac{5}{6} = 0.833$$

4.1.2 Object-based Metrics

1. **Average Object Fragmentation**

   The intention of this metric is to penalize a tracking algorithm for multiple tracking output objects covering the same ground truth object. Given a ground truth object $G_i^{(t)} \in G^{(t)}$, the fragmentation of $G_i^{(t)}$ is defined as

$$Frag(G_i^{(t)}) = \begin{cases} undefined & if\, N_{D^{(t)} \cap G_i^{(t)}} = 0 \\ \frac{1}{1+\log_{10}(N_{D^{(t)} \cap G_i^{(t)}})} & otherwise \end{cases} \tag{4.11}$$

   where $N_{D^{(t)} \cap G_i^{(t)}}$ is the number of tracking output objects in $D^{(t)}$ that overlap with ground truth object $G_i^{(t)}$. The average ground truth object fragmentation for frame t is given by

$$\boxed{Frag(t) = \frac{\sum_{i=1}^{N_{G^{(t)}}} Frag(G_i^{(t)})}{N_{G^{(t)}}}} \tag{4.12}$$

   The overall ground truth object fragmentation of a video sequence is defined as

$$OverallFragmentation = \frac{\sum_{t=1}^{N_f} Frag(t)}{\sum_{t=1}^{N_f} N_{G^{(t)}}} \tag{4.13}$$

   Figure 4.2 illustrates the function $Frag(n) = \frac{1}{1+\log_{10} n}$. It can be seen, that for a small number of overlapping tracking output objects in $D^{(t)}$ with a ground truth object $G_i^{(t)}$ the value of the function is higher than for a bigger number of overlappings. This means the bigger the value for $OverallFragmentation$ the less tracking output objects are produced to explain one ground truth object.

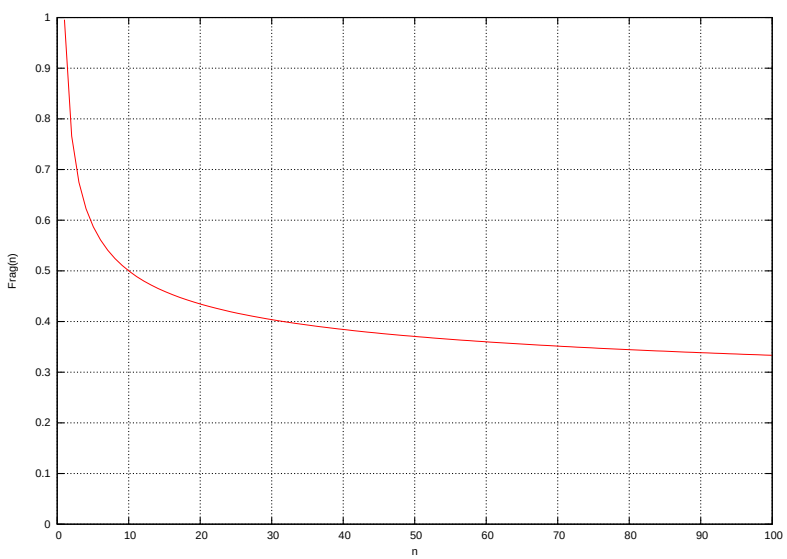
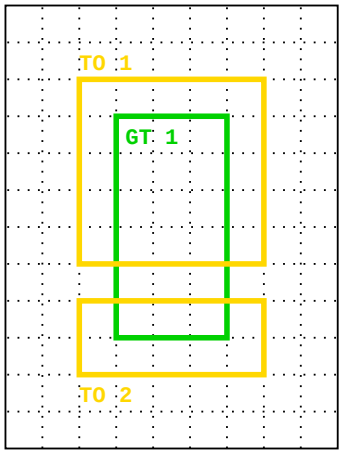


Figure 4.2: Ground truth object fragmentation metric. *The figure illustrates the function $Frag(n) = \frac{1}{1+\log_{10} n}$ where n represents the number of overlapping tracking output objects $D^{(t)}$ with a ground truth object $G_i^{(t)}$. The less overlappings, the bigger the value of the function and the less tracking output objects are produced to explain a ground truth object.*

Example:



The ground truth object is explained by two tracking output objects. Framewise fragmentation is

$$Frag(t) = \frac{1}{1 + \log_{10} 2} = 0.769$$

2. **Object-Based Precision**

Object-based precision of an object $D_i^{(t)}$ is the ratio of the proportion of its area that covers ground truth objects $G_{(t)}$ with its total area size in pixels. Thus, object-based precision is defined as

$$\boxed{ObjectPrecision(D_i^{(t)}) = \frac{|D_i^{(t)} \cap U_{G(t)}|}{|D_i^{(t)}|}} \tag{4.14}$$

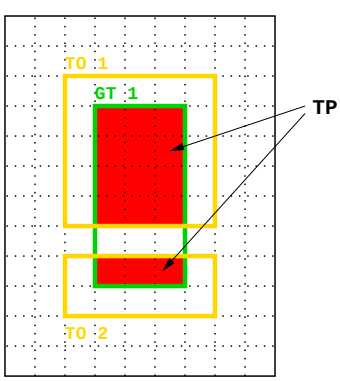
Total object-based precision of frame t is given by the ratio

$$Precision_{object-based}(t) = \frac{\sum_{\forall D_i^{(t)}} ObjectPrecision(D_i^{(t)})}{N_{D^{(t)}}} \tag{4.15}$$

Overall object-based precision of all tracking output objects of a video sequence is the weighted average object precision of all frames in the ground truth data set.

$$OverallPrecision_{object-based} = \frac{\sum_{t=1}^{N_f} N_{D^{(t)}} \times Precision_{object-based}(t)}{\sum_{t=1}^{N_f} N_{D^{(t)}}} \tag{4.16}$$

Example:



Object-based precision is defined as the ratio of True Positives per tracking output object (normalized by the tracking output object's size) with the total number of tracking output objects (two).

$$Precision_{object-based}(t) = \frac{\frac{12}{25} + \frac{3}{10}}{2} = 0.39$$

3. **Object-based Recall**

Object-based recall of an object $G_i^{(t)}$ is the ratio of the proportion of its area that is covered by the set of tracking output objects $D^{(t)}$ with its total area size. It is defined as

$$\boxed{ObjectRecall(G_i^{(t)}) = \frac{|G_i^{(t)} \cap U_{D^{(t)}}|}{|G_i^{(t)}|}} \tag{4.17}$$

Total object-based recall of frame t is then defined as the ratio of the sum of object recalls with the total number of ground truth objects of frame t.

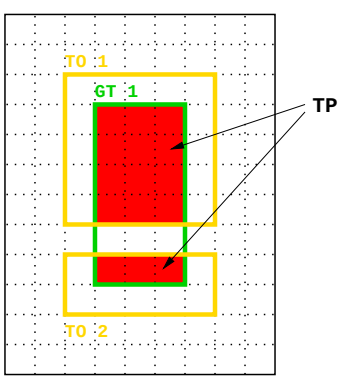
$$Recall_{object-based}(t) = \frac{\sum_{\forall G_i^{(t)}} ObjectRecall(G_i^{(t)})}{N_{G^{(t)}}} \tag{4.18}$$

Overall object-based recall of a video sequence of N_f frames containing ground truth objects is given by

$$OverallRecall_{object-based} = \frac{\sum_{t=1}^{N_f} N_{G^{(t)}} \times Recall_{object-based}(t)}{\sum_{t=1}^{N_f} N_{G^{(t)}}} \tag{4.19}$$

This metric gives credit to any portion of a ground truth object that is covered by the tracking output no matter how small that portion is. This can be thought of as the "degree" of object detection of the tracking algorithm.

Example:



Object-based recall is the ratio of True Positives per ground truth object (normalized by the ground truth object's size) with the total number of ground truth objects (one).

$$Recall_{object-based}(t) = \frac{\frac{12+3}{18}}{1} = 0.833$$

4.1.3 Localized Object-based Metrics

1. **Localized Object-Based Precision**

Localized object-based precision counts the number of tracking output objects $D_i^{(t)}$ that significantly cover ground truth objects.

Localized object-based precision is defined as

$$Loc\_Obj\_Precision(t) = \sum_{\forall D_i^{(t)}} ObjDetect_{TO}(D_i^{(t)}) \tag{4.20}$$

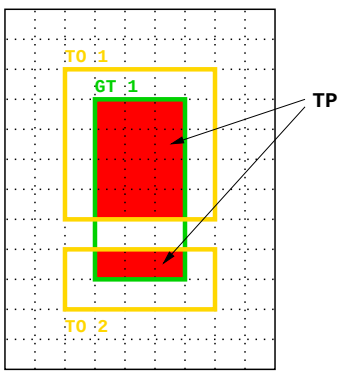
where

$$\boxed{ObjDetect_{TO}(D_i^{(t)}) = \begin{cases} 1 & if \ \frac{|D_i^{(t)} \cap U_{G^{(t)}}|}{|D_i^{(t)}|} > \xi \\ 0 & otherwise \end{cases}} \tag{4.21}$$

states for each tracking object $D_i^{(t)}$ if it sufficiently covers ground truth. Threshold ξ defines how large the minimum proportion of the overlapping area $|D_i^{(t)} \cap U_{G^{(t)}}|$ should be. The ratio of tracking output boxes that are precise enough with respect to ξ with the total number of tracking output objects produced by the tracking algorithm is given by

$$Overall\_Loc\_Obj\_Precision = \frac{\sum_{t=1}^{N_f} Loc\_Obj\_Precision(t)}{\sum_{t=1}^{N_f} N_{D^{(t)}}} \tag{4.22}$$

Example:



Threshold ξ is set to 0.75. Both tracking output objects are not precise enough:

$$\frac{12}{25} = 0.48 \not> \xi \Rightarrow ObjDetect_{TO}(TO_1) = 0$$

$$\frac{3}{10} = 0.3 \not> \xi \Rightarrow ObjDetect_{TO}(TO_2) = 0$$

$$\Rightarrow Overall\_Loc\_Obj\_Precision = \frac{0+0}{2} = 0$$

2. **Localized Object-Based Recall**

For each ground truth object $G_i^{(t)}$, localized object-based recall makes a hard decision whether it is considered detected or not. In this metric, a ground truth object $G_i^{(t)}$ is detected if a minimum proportion of its area is covered by $D_{(t)}$. The metric counts the number of ground truth objects that are detected, thus significantly covered by tracking output objects. In contrast to object-based recall, in this metric a ground truth object is either detected or missed depending on threshold ξ. Localized object-based recall is defined as

$$Loc\_Obj\_Recall(t) = \sum_{\forall G_i^{(t)}} ObjDetect_{GT}(G_i^{(t)}) \tag{4.23}$$

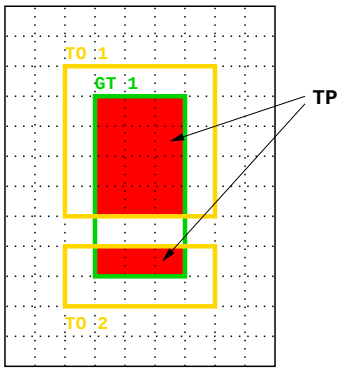
where

$$\boxed{ObjDetect_{GT}(G_i^{(t)}) = \begin{cases} 1 & if \ \frac{|G_i^{(t)} \cap U_{D^{(t)}}|}{|G_i^{(t)}|} > \xi \\ 0 & otherwise \end{cases}} \tag{4.24}$$

states for each ground truth object $G_i^{(t)}$ if it is sufficiently covered by the tracking output. The ratio of the total number of detected ground truth objects with the total number of ground truth objects of video sequence is

$$Overall\_Loc\_Obj\_Recall = \frac{\sum_{t=1}^{N_f} Loc\_Obj\_Recall(t)}{\sum_{t=1}^{N_f} N_{G^{(t)}}} \tag{4.25}$$

Example:



Threshold ξ is set to 0.75. The ratio of True Positives with the total number of pixels of ground truth object one is large enough.

$$\frac{15}{18} = \frac{5}{6} = 0.833 > \xi \Rightarrow ObjDetect_{GT}(GT_1) = 1$$

$$\Rightarrow Overall\_Loc\_Obj\_Recall = \frac{1}{1} = 1$$

4.2 Selection of Suitable Video Sequences

In order to evaluate the Reading People Tracker, video sequences needed to be captured. In total, 92 sequences were recorded of which 23 are used for evaluation purposes. All sequences were captured indoor at ETH Zurich. Each sequence shows a typical situation which may or may not cause problems for a people tracker: People crossing (occluding) each other, occlusion with background and groups of people are examples for such situations. The goal was to obtain a large data set of video sequences for the evaluation of the Reading People Tracker but also for benchmarking different tracking algorithms later.

There are 12 different subjects in the sequences in total. Two women and 10 men of different height and weight. Each type of sequence was captured three to twelve times with different persons. This allows it to compute average values for precision and recall for a type of video sequence. In this way, trends can be identified which hold for a whole type of video sequence. All sequences are captured at a constant frame rate of 25Hz and have a size of 352x288 pixels (approximately half PAL resolution). Those sequences which were captured many times but with different people are grouped together and associated with a “type” of video sequence. The type of sequence says something about the complexity of the sequence. In general, the higher the type the more difficult the situation for today’s tracking algorithms. This grouping of video sequences does not follow general guidelines or standards. It rather follows personal than formal guidelines for the notion of complexity. Table 4.1 and 4.2 gives information about the sequences that were used for the evaluation. Figure 4.3 shows the ground plane of the video capturing scene at ETH Zurich.

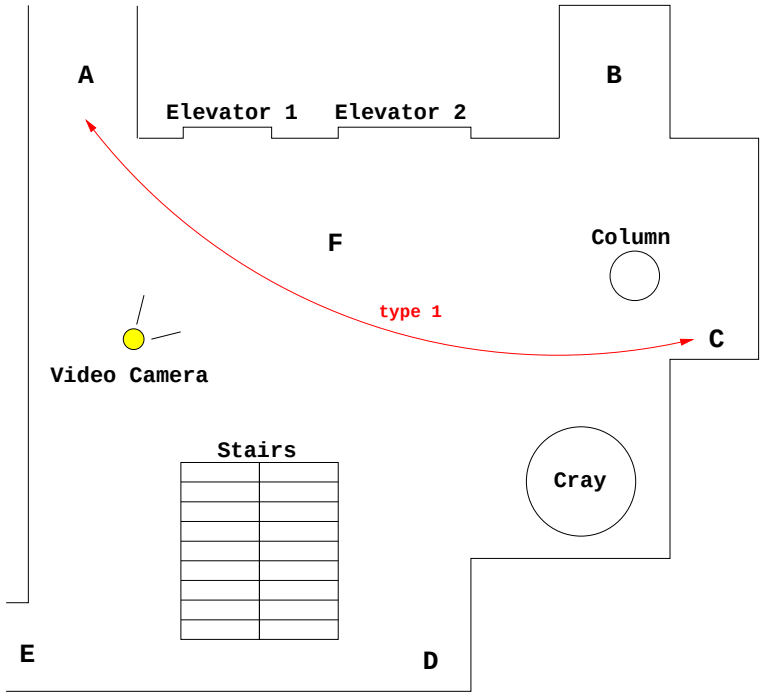


Figure 4.3: Video capturing scene at ETH Zurich. *The figure shows the ground plane of the video capturing scene at ETH Zurich. The letters A to F denote start or end points of people walking through the scene. A calibrated pan/tilt camera is used. In addition, video sequence type 1 is shown where people either walk from A to C or C to A.*

4.3 Framewise Evaluation Type

The current version of `ViPER` (v4 Alpha 3) offers three types of evaluations: framewise, object and tracking evaluation. Only framewise evaluations are performed in this thesis. Given a video sequence of N frames, a framewise evaluation compares the set of tracking output objects $D^{(t)}$ for each frame t, $t = 1, \ldots, N$ with the set of ground truth objects $G^{(t)}$ for this frame t. A typical framewise evaluation is shown in the following code snippet:

```
#BEGIN_FRAMEWISE_EVALUATION
OBJECT PERSON
        BODY : matchedpixels missedpixels falsepixels fragmentation \
        areaprecision arearecall [areaprecision .75] [arearecall .75]
#END_FRAMEWISE_EVALUATION
```

In this example, the object or *descriptor* [19] of type `PERSON` is evaluated. Although it is possible to specify different types of objects to evaluate using `ViPER`, like `GROUP_OF_PEOPLE`, `CAR` etc, only objects of type `PERSON` are considered in this work. `ViPER` searches for corresponding objects of type person in the tracking output and in the ground truth. Both are specified in XML. The objects' *BODY* attributes are compared. True Positives (matchedpixels), False Negatives (missedpixels), False Positives (falsepixels) as well as object-based fragmentation is calculated (compare section 4.1). Object-based precision and recall (areaprecision, arearecall) as well as localized object-based precision and recall ([areaprecision .75],[arearecall .75]) are also computed in this example. Threshold ξ which specifies the minimum portion of an object that must be covered is set to 0.75 (see section 4.1).

Type	Sequence	Duration in frames	Number of persons
1	V1	179	1
1	V2	193	1
1	V3	203	1
1	V4	190	1
1	V5	205	1
1	V6	216	1
1	V7	211	1
1	V8	215	1
1	V9	221	1
1	V10	201	1
1	V11	232	1
1	V12	221	1
2	V13	122	1
2	V14	134	1
2	V15	120	1
3	V16	199	1
3	V17	229	1
3	V21	199	1
3	V22	207	1
3	V23	233	1
4	V18	242	2
4	V19	297	2
4	V20	211	2

Table 4.1: Video Sequences used for evaluating the Reading People Tracker. *Each video sequence is associated with a type (column 1). In general, the higher the type the more difficult the situation for a people tracker. In addition, the table shows the duration of a sequence in frames and the number of persons appearing in it. The video sequences all captured at a constant frame rate of 25Hz and at a size of 352x288 pixels (approximately half PAL resolution).*

Type	Description
1	One person is walking either from C to A (passing in front of column), A to C (passing in front of column), B to E or E to B. These are the most simplest video sequences (besides the empty sequence with no persons appearing). They were captured to gain basic knowledge about the tracking algorithm and to fix basic tracking parameters like motion detection thresholds, region sizes etc.
2	One person is running from C to A, passing in front of the column. This type of sequence may be used to test and evaluate the implemented linear first-order motion model of the Region Tracker that predicts new positions of a track regions.
3	One person is walking from C to A or A to C, passing behind the column each way. This type of sequence shows one person that is temporally occluded. The type of sequence can be used to evaluate occlusion handling of the Reading People Tracker.
4	These type of sequence shows two persons, one walking from C to A and the other walking from A to C, crossing each other at F. It is very important to evaluate the tracker on multi-person sequences since the reliability of the tracker strongly depends on the performance on this type of video sequences. For most of today's tracking algorithms, single person tracking does not present a big challenge. Thus, multi-people tracking is crucial.

Table 4.2: Video Sequences Description. *The table gives a description of each type of video sequence that is used for the evaluation. Figure 4.3 shows the video capturing scene at ETH Zurich.*

4.4 Results focusing on Motion Detection

This section presents evaluation results focusing on motion detection. The Motion Detector of the Reading People Tracker is described in section 3.1.

4.4.1 Background-Foreground Segmentation Threshold

In the Reading People Tracker, background-foreground segmentation is the process of separating walking people from a static background. The motion image is generated by first pixelwise differencing the video and the background image and then thresholding the resulting difference image. This threshold is represented by the parameter `DETECT_DIFF_THRESH` of the Motion Detector and must be within the range of 0 and 1. The parameter is internally projected to the range $[0 : 255]$. For two corresponding pixels P_V and P_B of the video and background image, pixel P_V is considered as foreground if

$$|V_{P_V} - V_{P_B}| > \texttt{DETECT\_DIFF\_THRESH} \tag{4.26}$$

where V_{P_V}, V_{P_B} are (normalized) pixel values.

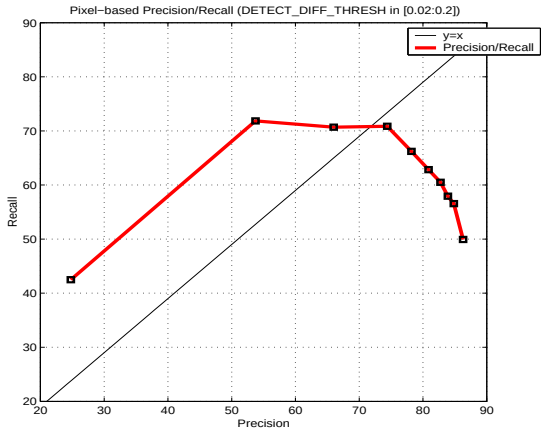
The influence of `DETECT_DIFF_THRESH` is evaluated on video sequence type 1. A framewise evaluation is performed and the range of `DETECT_DIFF_THRESH` is set to $[0.02 : 0.2]$. Figure 4.4 shows the results for average pixel-based, object-based and localized object-based precision and recall. The curves that are shown are precision-recall curves, i.e. precision on the x-axis versus recall on the y-axis. The point of interception of the precision-recall curve and the function $y = x$ is the point where precision and recall are equal. Figure A.1 of appendix A shows the results for each video sequence of video sequence type 1.

Analysis

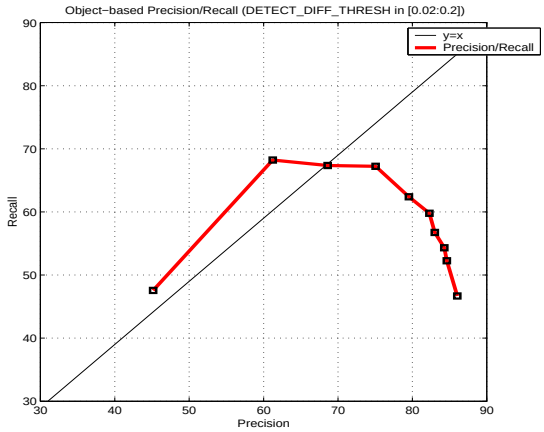
Consider figure 4.4c. The point on the precision-recall curve where `DETECT_DIFF_THRESH` is 0.08 is labeled with its function value. Increasing `DETECT_DIFF_THRESH` would result in higher precision but lower recall. On the other hand, decreasing `DETECT_DIFF_THRESH` does not affect recall as much as precision. Hence, a value of about 0.8 for `DETECT_DIFF_THRESH` is optimal.

Figures 4.5a to 4.5f show frame 77 of video sequence V2. Figures 4.5a to 4.5c show ground truth (green bounding boxes) as well as tracking output (yellow boxes) for parameter `DETECT_DIFF_THRESH` 0.02, 0.05 and 0.10. Figures 4.5d to 4.5f illustrate the corresponding motion images produced by the tracking algorithm. It can be seen that precision of tracking output objects increases as the parameter increases (tracking output boxes get smaller). Figure 4.5a shows one tracking output object caused by illumination changes on the wall on the left. It fulfills size constraints of tracked region posed by the Motion Detector. This means, the region is tracked for a short time. In figure 4.5b this region is no more recognized as foreground. The detected region is not as precise as the same region in figure 4.5c. This is because shadows on the floor caused by the walking person are also identified as foreground.

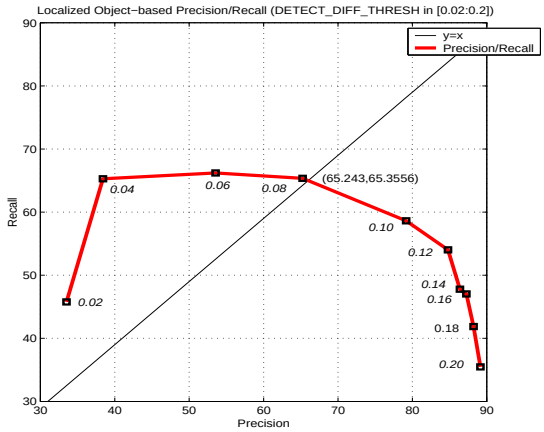
Figures 4.5g to 4.5l show ground truth and tracking output as well as motion images for frame 145. From figures 4.5g and 4.5j it can be seen that for a value of 0.02 for `DETECT_DIFF_THRESH`, no walking people are detected. This is because potential foreground regions do not fulfill size constraints. Precision increases but recall decreases as `DETECT_DIFF_THRESH` is changed. Comparing figures 4.5k and 4.5l, one can see that region tracking is successful for `DETECT_DIFF_THRESH` set to 0.05. For a value of 0.1, the Region Tracker can not detect a person. No region is extracted by the Motion Detector. But the Active Shape Tracker is still able to identify and track the walking person. This is shown in figure 4.5i where the (smaller) yellow box depicts the bounding box of the fitted shape. Figure 4.6a and 4.6b show the output pictures corresponding to figures 4.5h and 4.5i produced by the Reading People Tracker. As a result of this evaluation, `DETECT_DIFF_THRESH` is set to 0.85 in further evaluations.



(a)



(b)



(c)

Figure 4.4: Precision-recall curves for detection difference threshold evaluation. *Figure 4.4a illustrates average pixel-based precision and recall for video sequence type 1. Figure 4.4b average object-based, and figure 4.4c average localized object-based precision and recall. The function value of the precision-recall curve for a value of 0.8 for* `DETECT_DIFF_THRESH` *is shown in figure 4.4c. Choosing a higher value than 0.8 would increase precision but decrease recall. Choosing a smaller value would not affect recall very much but decreases precision. The optimum is therefore close to 0.8 for* `DETECT_DIFF_THRESH`*. In further evaluations, the parameter is set to 0.85.*

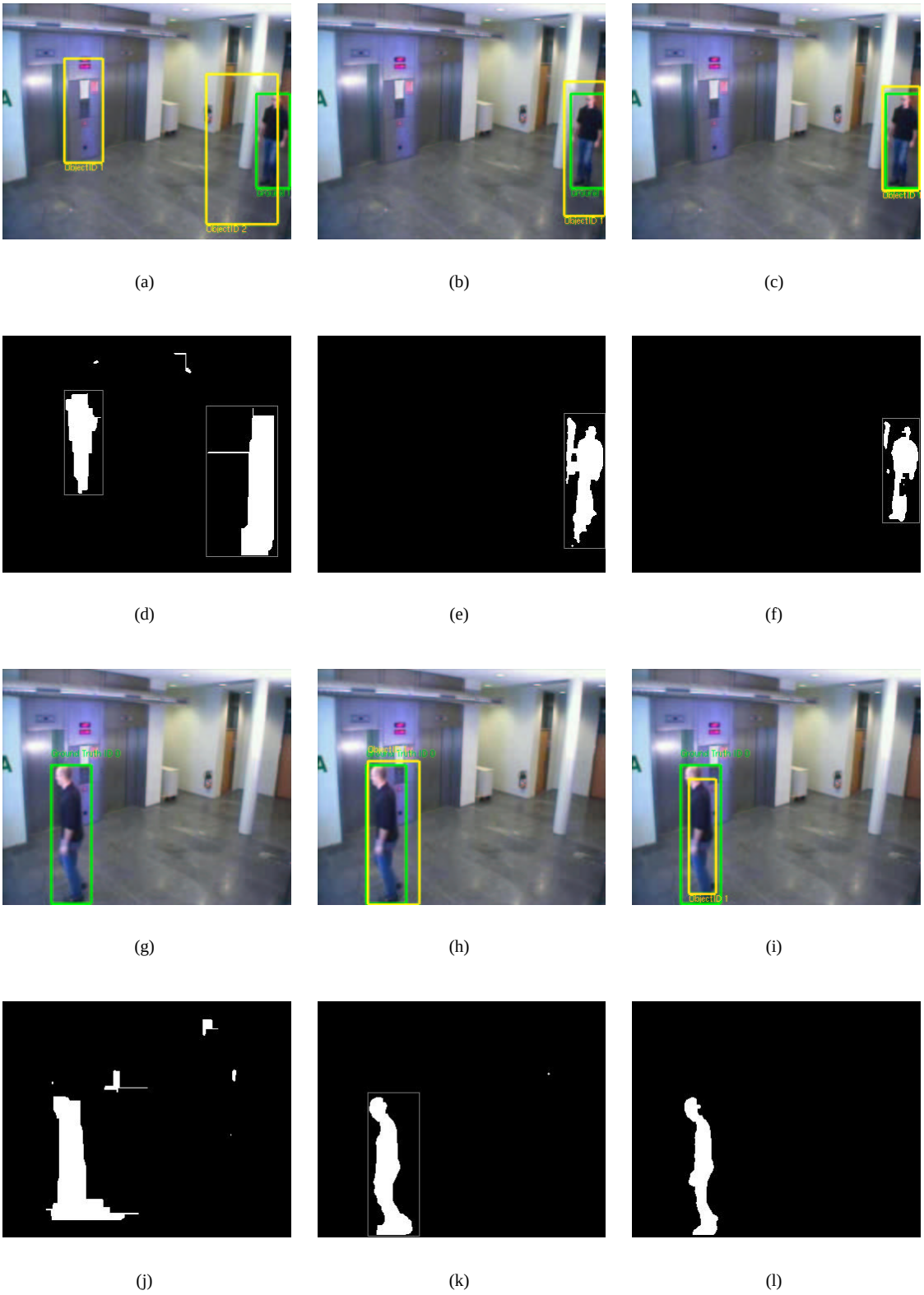
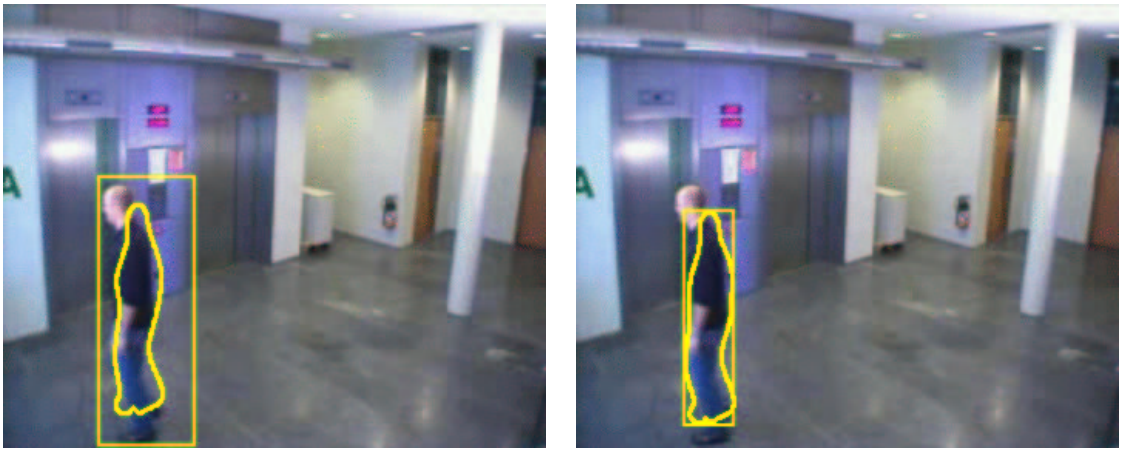


Figure 4.5: Ground truth and tracking output as well as motion image for frames 77 and 145. *The figure shows ground truth (green bounding boxes) and tracking output (yellow bounding boxes) for frames 77 (rows one and two) and 145 (rows three and four). Motion images are shown for each video image. Ground truth object recall decreases as* `DETECT_DIFF_THRESH` *increases. The reverse holds for precision of tracking output objects.*



(a) detect_diff_thresh = 0.05

(b) detect_diff_thresh = 0.10

Figure 4.6: Tracking output of Region Tracker and Active Shape Tracker. *Figure 4.6a shows tracking output where both, the Region and the Active Shape Tracker recognize the walking person. The bounding box of the tracked region and the fitted shape are depicted. For a value of 0.10 for* `DETECT_DIFF_THRESH` *the Region Tracker fails to detect the person. But the Active Shape tracker still tracks the person. Thus, the Region Tracker uses the bounding box of the fitted shape as a new hypothesis for a region (figure 4.6b).*

4.4.2 Minimum Size of Tracked Regions

The minimum size of a tracked region is given by its area A_{min}^{Region}. The value of parameter `MIN_REGION_SIZE` is calculated by the formula

$$MIN\_REGION\_SIZE = \frac{A_{min}^{Region}}{A^{Image}} \tag{4.27}$$

where A^{Image} is the area of the whole video image, i.e. the product of its width and height.

`MIN_REGION_SIZE` should be set to a suitable value. This because first, it does not make sense to track regions which are too small to represent walking people. Second, because otherwise video image noise, which is normally only a few pixels in size is identified as valid foreground. Per default the value of the parameter is set to 0.001. For video images of size 352x288 (approximately half PAL resolution), 0.001 corresponds to a region of about 100 pixels (or 10x10 pixels). Compare figure 4.7. To estimate a suitable value for `MIN_REGION_SIZE`, a statistical model is used. All ground truth objects of all video sequences which were used for evaluating the Reading People Tracker are therefore considered. The set of ground truth objects of all sequences is N_{GT}. $|N_{GT}|$ is the number of objects in this set. Ground truth objects are represented as rectangular bounding boxes with known position (top left corner), width W_{G_i} and height H_{G_i}. It is assumed that the widths and heights are normally distributed. `MIN_REGION_SIZE` is given by the formula

$$\boxed{MIN\_REGION\_SIZE = \mu_{A_{min}^{Region}} - \sigma_{A_{min}^{Region}}} \tag{4.28}$$

where $\mu_{A_{min}^{Region}}$ is the average

$$\mu_{A_{min}^{Region}} = \frac{\sum_{i=1}^{|N_{GT}|} W_{G_i} \times H_{G_i}}{|N_{GT}|} \tag{4.29}$$

and $\sigma_{A_{min}^{Region}}$ the standard deviation

$$\sigma_{A_{min}^{Region}} = \sqrt{\sum_{i=1}^{|N_{GT}|} (\mu_{A_{min}^{Region}} - (W_{G_i} \times H_{G_i}))^2} \tag{4.30}$$

The result of equation 4.28 gives a value of 0.0359 for `MIN_REGION_SIZE`. This value corresponds to about 3640 pixels. Figure 4.7 shows the relations. The red colored square depicts a region of 10x10 pixels whereas the green rectangle has size 40x91 pixels. It can be seen that the green rectangle covers much preciser the area of a walking person. The same statistical model is applied to parameters `MAX_REGION_SIZE`, `MIN_HEIGHT_TO_WIDTH`, and `MAX_HEIGHT_TO_WIDTH` (see section 3.1). A framewise evaluation is performed and the range of `MIN_REGION_SIZE` is set to $[0.01 : 0.1]$.

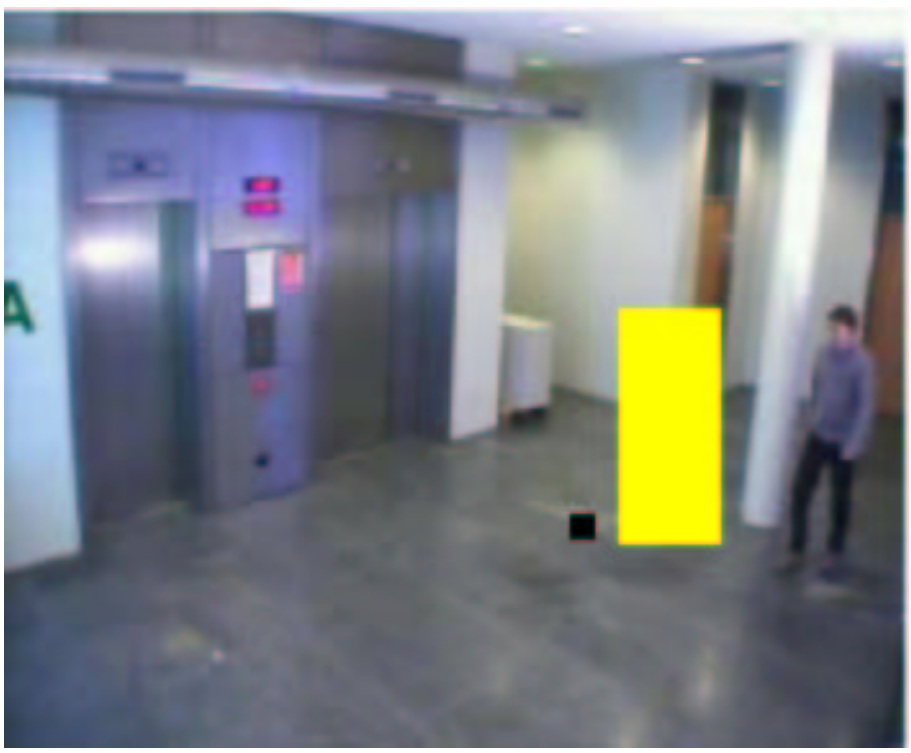


Figure 4.7: Minimum region sizes. *The area of the yellow rectangle is of size 40x91 pixels. This size is approximated by using a statistical model which takes into consideration all ground truth objects of all video sequences used for evaluation. The yellow rectangle covers the region of a walking person much preciser than the black square. The square has size 10x10 pixels (default value set by the Reading People Tracker).*

Analysis

Comparing figures 4.8a, 4.8b and 4.8c shows a linear trend of precision. Different values for `MIN_REGION_SIZE` do only affect recall, not precision. As parameter `MIN_REGION_SIZE` increases, fewer regions get extracted from the motion image. Equation 4.8 explain this trend. Detected regions are weighted by their size in pixels. Larger regions are therefore stronger weighted. For a value of 0.03 for

MIN_REGION_SIZE for example, people appearing in the rear part of the video image get extracted correctly. But this regions do not contribute much to overall precision. They are weighted less strong. This means if regions are not detected because a too large value for MIN_REGION_SIZE is used, this does not significantly affect overall precision.

Recall is decreasing for increasing values of MIN_REGION_SIZE. If no regions are detected in one frame t, pixel-based recall is 0 (equation 4.9). This affects overall recall because no tracking output objects are matched with ground truth.

Figures A.2b, A.2d and A.2f of appendix A show a sharp bend of recall curves for video sequence V11 between 0.04 to 0.06. Video sequence V12 does not show this bend. It is quite flat between 0.04 and 0.06. Figure 4.9 shows video frame 152 of sequence V11, figure 4.10 frame 163 of sequence V12. For both figures, if a too large value is used for MIN_REGION_SIZE, no regions are extracted from the motion image (4.9f and 4.10f). In the case of figure 4.10d, the Active Shape Tracker keeps tracking the walking person, whereas in figure 4.9d it already lost the track (no tracking output at all). This explains the sharp bend of recall curves in figure A.2 between 0.04 and 0.06 for video sequence V11. Both, the Region Tracker and the Active Shape Tracker lose the track of the walking person. The result is that fewer tracking output objects are compared with ground truth when calculating localized object-based recall for instance. This in turn decreases ground truth object recall.

For further evaluation steps, a value of 0.0359 for parameter MIN_REGION_SIZE is set. The statistical model which is used to compute this parameter turns out to be correct.

4.4.3 Region Merging Threshold vs. Detection Difference Threshold

REGION_MERGE_THRESH is a parameter of the Motion Detector that decides if two extracted regions are merged into one. Two regions are merged if their relative distance to their sizes is smaller than this threshold. The measure D_{r_1,r_2} tis defined as

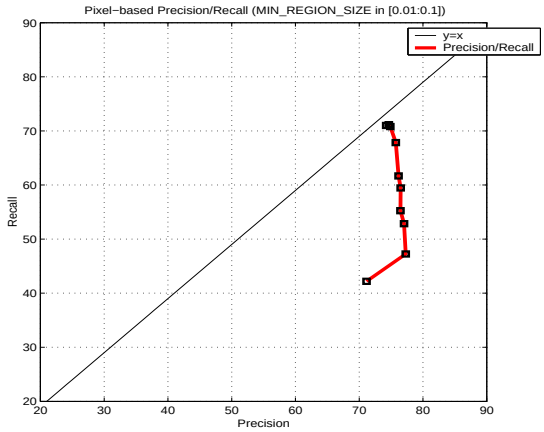
$$D_{r_1,r_2} = \frac{dx + dy}{max(W_{r_1}, H_{r_1}) + max(W_{r_2}, H_{r_2}) + 1} \tag{4.31}$$

where dx and dy is the absolute difference in x and y of the two regions. $W_{r_{1,2}}$ and $H_{r_{1,2}}$ are their widths and heights. Region merging is not an expensive operation and makes sense in two cases. First, a cloud of many small regions caused by image disturbance like noise is probably merged into one region. Second, small regions in the neighborhood of a larger one are also merged into one region. If merged regions do not fulfill other constraints as minimum or maximum region size, they are discarded at an other point of the tracking algorithm.

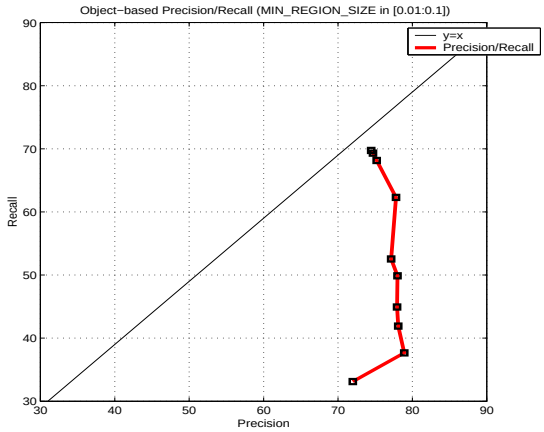
The parameter REGION_MERGE_THRESH is initially set to 0.5 in the Reading People Tracker package. To find a suitable value for this parameter a framewise evaluation on video sequence type 1 is performed. Parameter REGION_MERGE_THRESH is analyzed depending on parameter DETECT_DIFF_THRESH (see section 4.4.1). The results for average localized object-based precision and recall are shown in figure 4.11. Figure A.3 of appendix A illustrates average pixel-based and object-based precision and recall.

Analysis

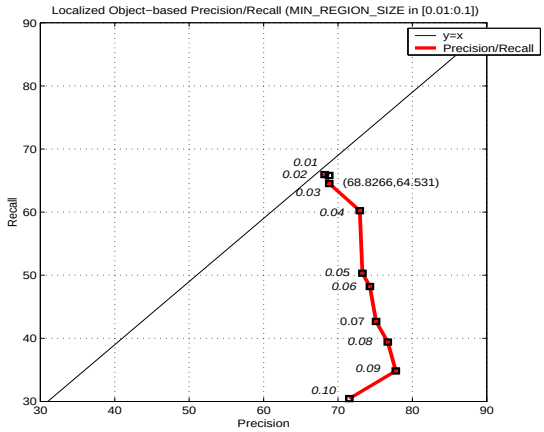
From figure 4.11 a linear trend of precision and recall can be identified as REGION_MERGE_THRESH increases. Figure 4.12 shows the precision-recall curve for only REGION_MERGE_THRESH in the range $[0.1 : 1.0]$. Recall is more affected than precision as REGION_MERGE_THRESH increases. Best recall is reached for a value of 0.5. The influences on overall precision and recall are small if this value is coupled with a value of 0.85 for DETECT_DIFF_THRESH. The dependency on this two parameters is marginal. The parameters are set to 0.5 and 0.85 for REGION_MERGE_THRESH and DETECT_DIFF_THRESH in further evaluations.



(a)



(b)



(c)

Figure 4.8: Precision-recall curves for minimum region size evaluation. *The figures illustrates average pixel-based, object-based and localized object-based precision and recall for video sequence type 1. A linear trend of precision can be identified. Figure 4.8c shows the function value where* `MIN_REGION_SIZE` *is equal 0.03. Increasing* `MIN_REGION_SIZE` *does not affect precision very much but recall rapidly decreases. A value of 0.0359 is set for further evaluations (as approximated by the statistical model).*

(a) (b)

(c) (d)

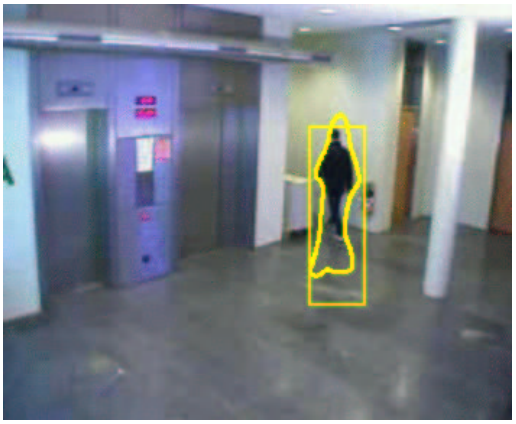
(e) (f)

min_region_size = 0.04 min_region_size = 0.05

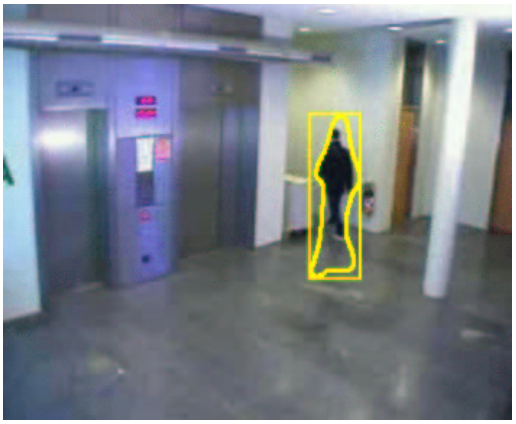
Figure 4.9: Tracking output, ground truth and motion image for frame 152 of video sequence V11 for `MIN_REGION_SIZE` of 0.04 and 0.05. *The figure illustrates tracking output (yellow bounding boxes) as well as ground truth (green bounding boxes) in the top row. The middle row only shows tracking output from Region and Active Shape Tracker as produced by the Reading People Tracker. The bottom row depicts the motion images. For too large values of `MIN_REGION_SIZE`, regions are not extracted any more by the Motion Detector. This heavily influences other tracking modules. Both, Region and Active Shape Tracker lose the track of the walking person. No tracking output is generated (figure 4.9d) and ground truth object recall rapidly decreases.*

(a)

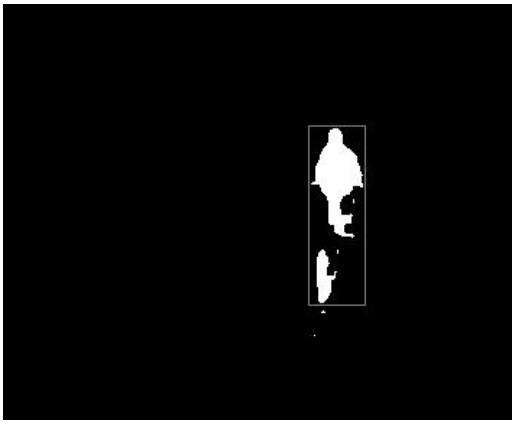
(b)



(c)



(d)



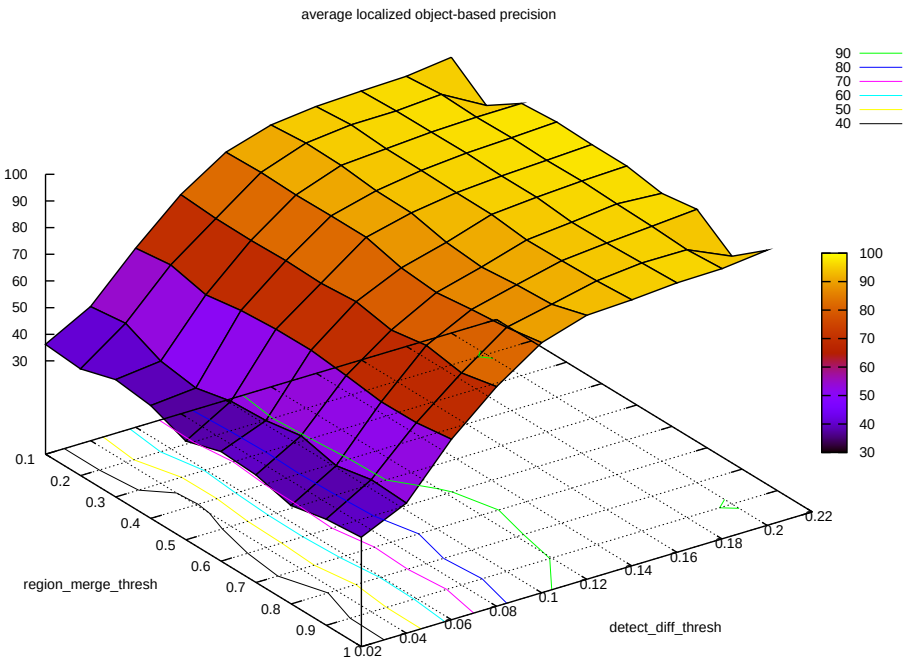
(e)

min_region_size = 0.04

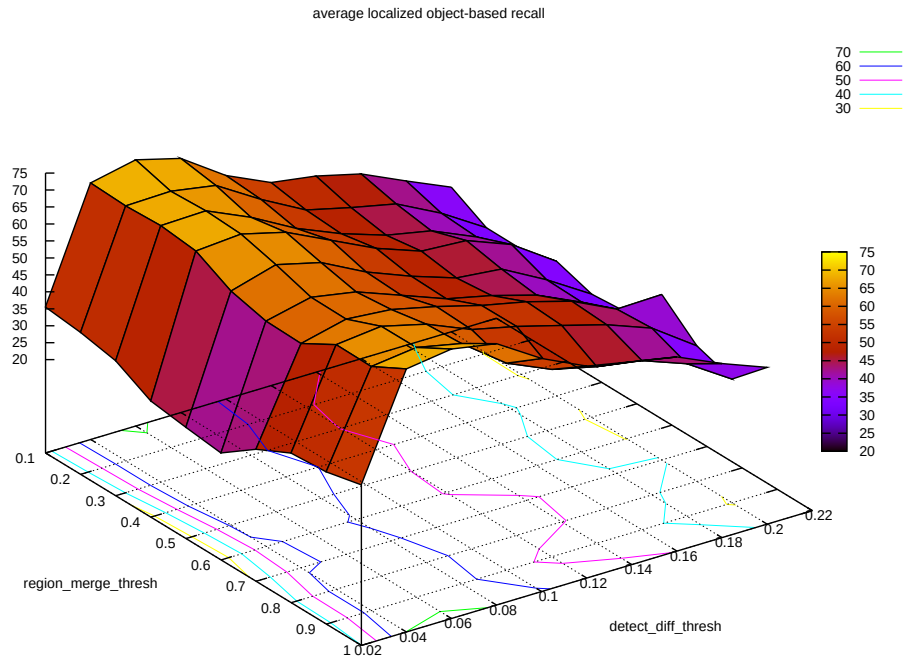
(f)

min_region_size = 0.05

Figure 4.10: Tracking output, ground truth and motion image for frame 163 of video sequence V12 for `MIN_REGION_SIZE` of 0.04 and 0.05 *The figure illustrates tracking output (yellow bounding boxes) as well as ground truth (green bounding boxes) in the top row. The middle row shows only tracking output from Region and Active Shape Tracker as produced by the Reading People Tracker. The bottom row depicts the motion images. For too large values of `MIN_REGION_SIZE`, regions are not extracted any more by the Motion Detector. In this case, the Active Shape Tracker keeps tracking the person (figure 4.10d). The track is not completely lost as in figure 4.9. This is why ground truth object recall is higher for this video sequence than for sequence V11. Ground truth is more often detected.*



(a)



(b)

Figure 4.11: Precision and recall for region merge threshold vs. detection difference threshold evaluation. *The surfaces show average localized object-based precision and recall for each pair of values. A linear trend can be identified for both, precision and recall as* `REGION_MERGE_THRESH` *increases. The dependency on this two parameters is only marginal.*

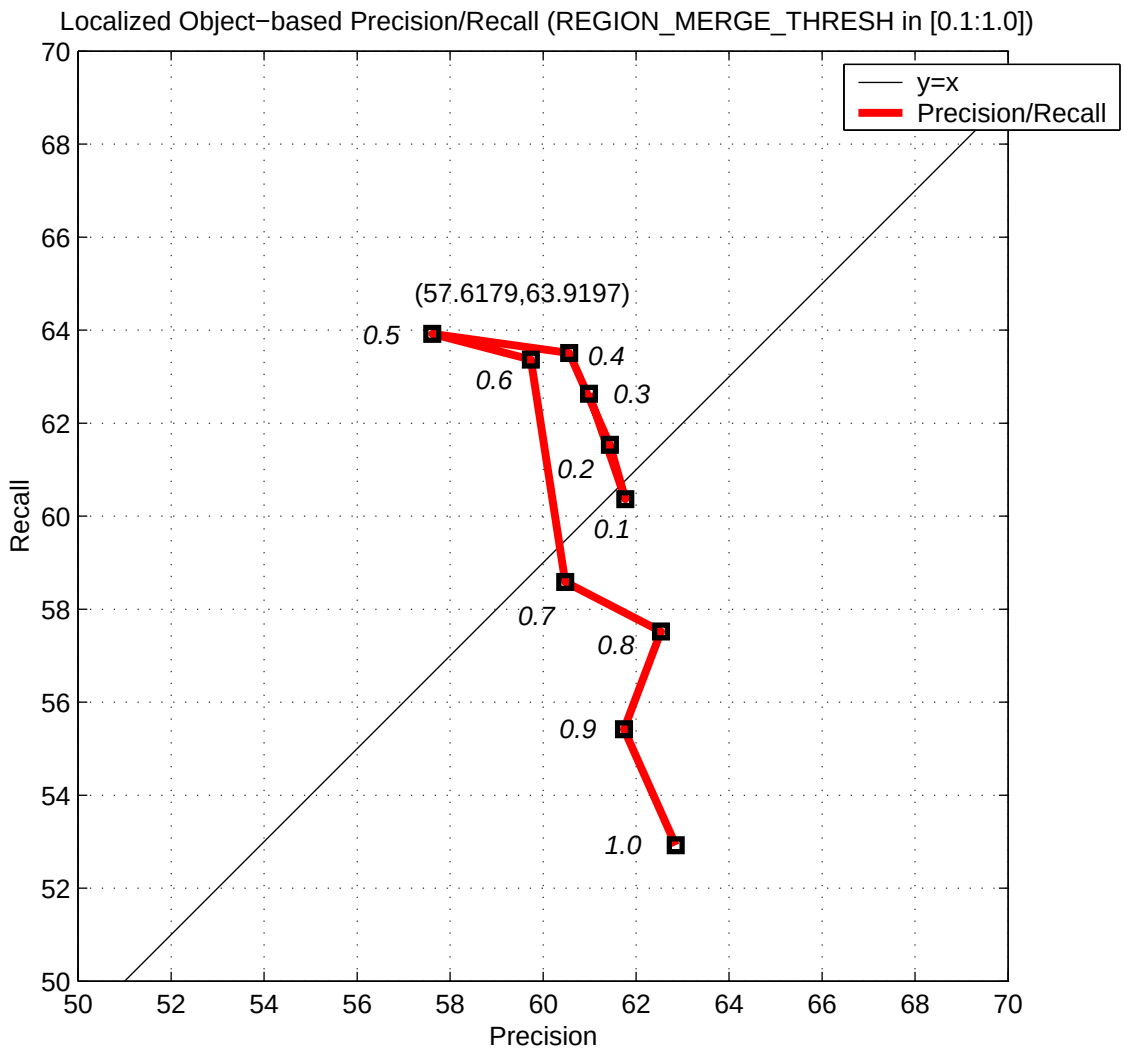


Figure 4.12: Precision-recall curve for region merge threshold evaluation. *A linear trend can be identified for precision. Recall is more affected than precision as* `REGION_MERGE_THRESH` *increases. Best recall is achieved for a value of 0.5. This value is therefore used in further evaluations. Using a higher (or lower) value would affect recall much stronger than precision.*

4.5 Influences of Image Quality on Tracking Performance

Two different techniques can be chosen to filter motion images. One technique to filter video and background images during motion detection. The influences of these filtering techniques on tracking performance are examined in this section. Available filtering techniques for motion images are:

- ***Median filter:*** This applies a spacial median filter on the motion image. For each pixel $P(x, y)$ (pivot pixel) of the motion image, its neighborhood $N_\epsilon(P(x, y))$ is examined. This neighborhood is given by a 3x3 pixel window which has $P(x, y)$ at its center. All pixel values within this window are sorted. The center of the sorted pixels is returned as new value for $P(x, y)$ in the motion image. The median filter will remove noise spikes from the motion image without significantly blurring its edges.

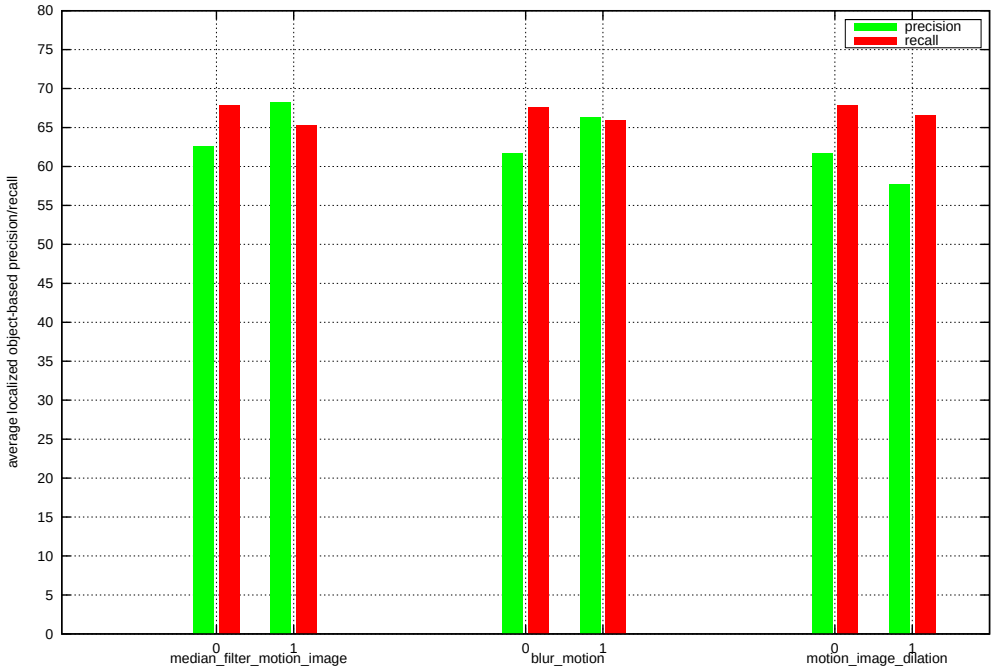


Figure 4.13: Localized object-based precision and recall for different image filtering techniques. *The figures illustrates precision and recall results for disabling (0) and enabling (1) an image filtering technique. The differences of precision and recall between median filtering motion images and blurring video images are small. Thus, only the median filter is used in further evaluations. Enabling Motion image dilation results in lower values for precision and recall (see figure 4.14).*

- ***Dilation:*** In this operation, any black pixel of the motion image (background) that has at least one white neighbor (foreground) is changed to white. This operation has the effect of changing black edge pixels to white, thereby increasing the size of foreground objects. If two or more foreground objects are separated by a gap, dilation has the effect of fusing these objects together. Dilation is the counterpart of *erosion*, i.e. dilating foreground pixels is equivalent to eroding background pixels.

Available filtering technique for video and background images is:

- ***Blurring:*** Blurring (aka *smoothing* or *Gaussian smoothing*) is achieved by convolution. A Gaussian convolution mask (Gaussian kernel) of size 3x3 pixels is used (in the Reading People Tracker). One property of a Gaussian kernel is that it forms a weighted average over all pixels within the kernel that weights pixels at its center much stronger than pixels at its boundaries [31].

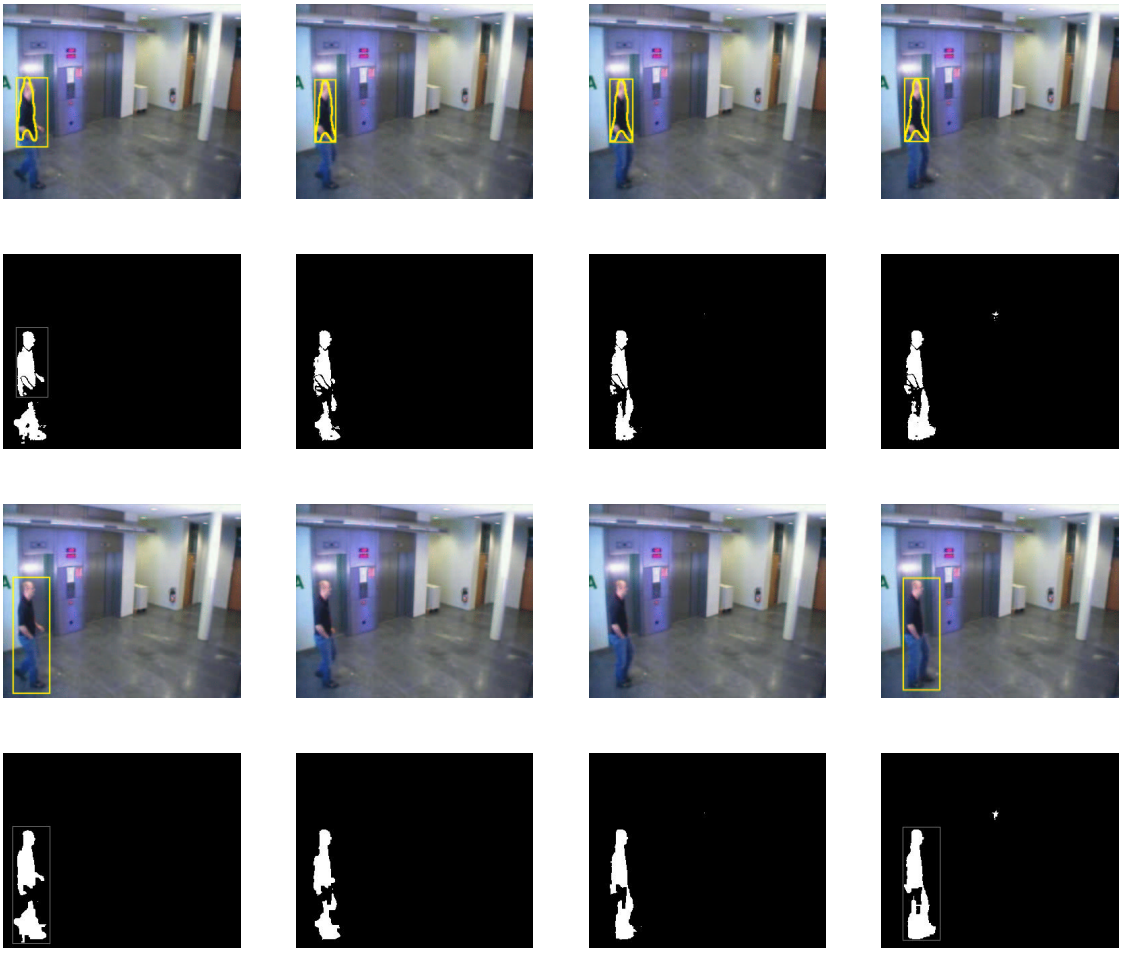
Figure 4.13 shows evaluation results for video sequence type 1. The histograms show average localized object-based precision and recall when a filtering technique is enabled (1) and disabled (0).

Analysis

Median filtering motion images and blurring video images results in similar values for precision and recall. In general, blurring, i.e. convoluting an image, is an expensive operation. Using a median filter is less expensive. Thus, blurring is disabled in further evaluations.

Figure 4.13 also illustrates that using dilation results in lower precision and recall. Consider figure 4.14. It shows frames 50 to 53 of video sequence V5. The first two rows show effects of disabling dilation.

Rows three and four reflect effects of enabling dilation. In the case of disabling motion image dilation, the region that is extracted from frame 50 is too small. The person's legs are missing. A shape is fitted into this region by the Active Shape Tracker. In the following frames, the Active Shape Tracker keeps tracking the person. No new region is extracted by the Motion Detector, thus the bounding box of the (wrong) fitted shape is propagated. Although the results of disabling dilation are poor, ground truth recall is increased due to equation 4.24. Ground truth objects (actually portions of ground truth objects) are more often detected when disabling dilation. In the case of enabling dilation, no tracking output is generated for frames 51 and 52, i.e. recall is lower. But the tracking output for frames 50 and 53 does fit the walking person much preciser. As a result of this evaluation, motion image dilation will be enabled in further evaluations and a small loss of recall accepted.



frame 50 frame 51 frame 52 frame 53

Figure 4.14: Effects of disabling and enabling motion image dilation. *The figure shows effects of disabling (first two rows) and enabling (second two rows) motion image dilation. Disabling motion image dilation results in poor tracking results. The size of the detected moving region is too small to contain the whole person. The legs are missing. The Active Shape Tracker fits a shape into this too small region. Recall is higher because more tracking output is generated for the whole frame span. Enabling dilation results in extracting a region of correct size (frame 50). The Active Shape Tracker fails to fit a person into this region. The Region Tracker in turn fails to detect regions for frames 51 and 52. Thus, no tracking output is generated. Ground truth recall is lower but the tracking output for frames 50 and 53 fits the walking person much preciser. Motion image dilation is therefore enabled in further evaluations and a small loss of recall accepted.*

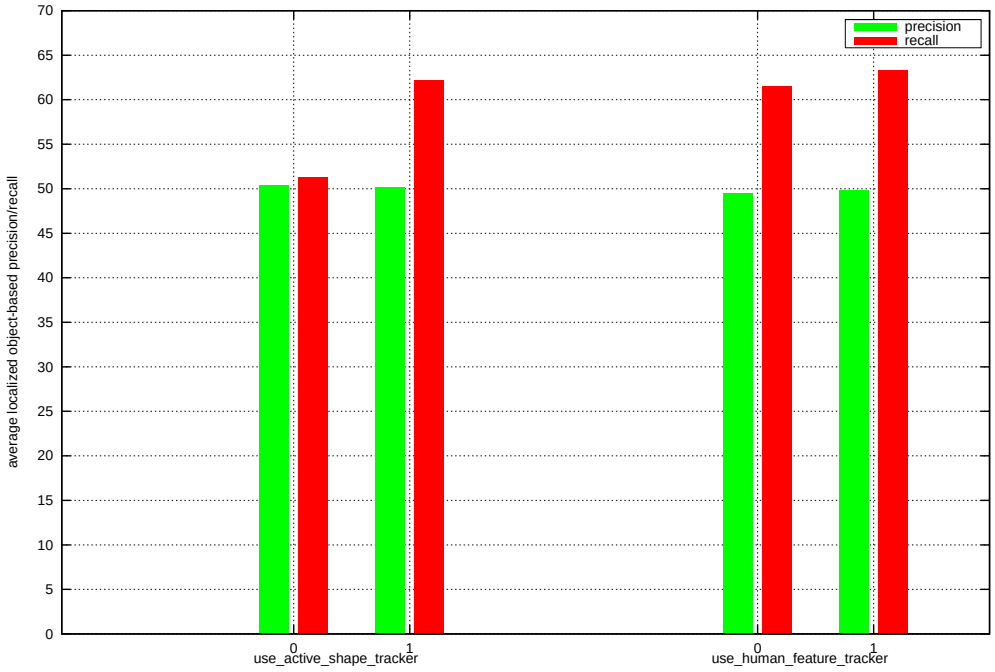


Figure 4.15: Benefits of Active Shape Tracker and Human Feature Detector on tracking performance for video sequence type 3. *The figure shows results for average localized object-based precision and recall when disabling (0) and enabling (1) the Active Shape Tracker and the Human Feature Detector. Using the Active Shape Tracker clearly increases recall. Precision is not much affected. The gain of enabling the Human Feature Detector in addition to the Active Shape Tracker is only marginal for this type of video sequence.*

4.6 Benefits of Active Shape Tracker on Tracking Performance

The Active Shape Tracker uses the output of the Region Tracker and the Human Feature Detector to initialize a track. Once a track is initialized for frame t, the Active Shape Tracker predicts its position for frame $t + 1$ using a Kalman filter (see section 3.4.1). If the Region Tracker fails to detect a region, i.e cannot match a prediction with a measurement from the Motion Detector, it consults the Active Shape Tracker. If the Active Shape Tracker successfully tracks the person, the bounding box of the fitted shape is taken as the new position of the region. This case is shown in figure 4.17. A framewise evaluation is performed on video sequence type 3.

Analysis

Figure 4.15 shows results for average localized precision and recall when the Active Shape Tracker and the Human Feature Detector are disabled (0) and enabled (1). Enabling the Active Shape Tracker clearly increases recall. The gain of enabling the Human Feature Detector to improve shape initialization is only marginal. The increase in recall can be explained considering figure 4.17. The top row shows the tracking output for frames 56 to 60 when the Active Shape Tracker is disabled. Only the Region Tracker is enabled. The middle row shows the results when the Active Shape Tracker is enabled. The bottom row shows the tracking output as produced by the Reading People Tracker (Active Shape Tracker enabled). The Region Tracker cannot detect a region for frames 57 to 59 because the Motion Detector does not extract a moving region. No tracking output is generated in this case. When enabling the Active Shape Tracker, it succeeds to fit a shape into the region detected by the Region Tracker. Frame 56 shows the fitted shape as well as

the surrounding bounding box. For subsequent frames, the bounding box of the fitted shape is used as new position for the lost region of frame 56.

The precision of the tracking output does not change when enabling/disabling the Active Shape Tracker. In either case, the precision of the output bounding boxes is the same. More valid tracking output is produced when using the Active Shape Tracker. The ground truth is well approximated for the depicted frame span. Due to equation 4.24 a higher recall is reached. Ground truth is more often detected.

Another benefit of using an Active Shape Tracker is when people are temporally occluded. Figure 4.18 shows one person passing behind a column. The Active Shape Tracker is disabled for figure 4.18a. The part of the person's outline in figure 4.18b which is again visible is sufficient to fit a shape. Hence, the track that was lost when the person was almost completely occluded is re-gained faster. This in turn increases overall ground truth recall.

The same framewise evaluation is accomplished for video sequence type 4. This type of sequence shows two persons who cross each other. The results for average localized precision and recall are shown in figure 4.16. Basically, the same arguments apply to the results in figure 4.16. Ground truth recall is again higher when using the Active Shape Tracker because people are tracked more stable. Precision increases as well. This is, because the more narrow bounding boxes of fitted shapes are taken as tracking output when the Region Tracker fails to detect regions. Another situation occurs as shown in figure 4.19. At the beginning, the two persons are successfully tracked, but as they cross, they are no more associated with the same identity.

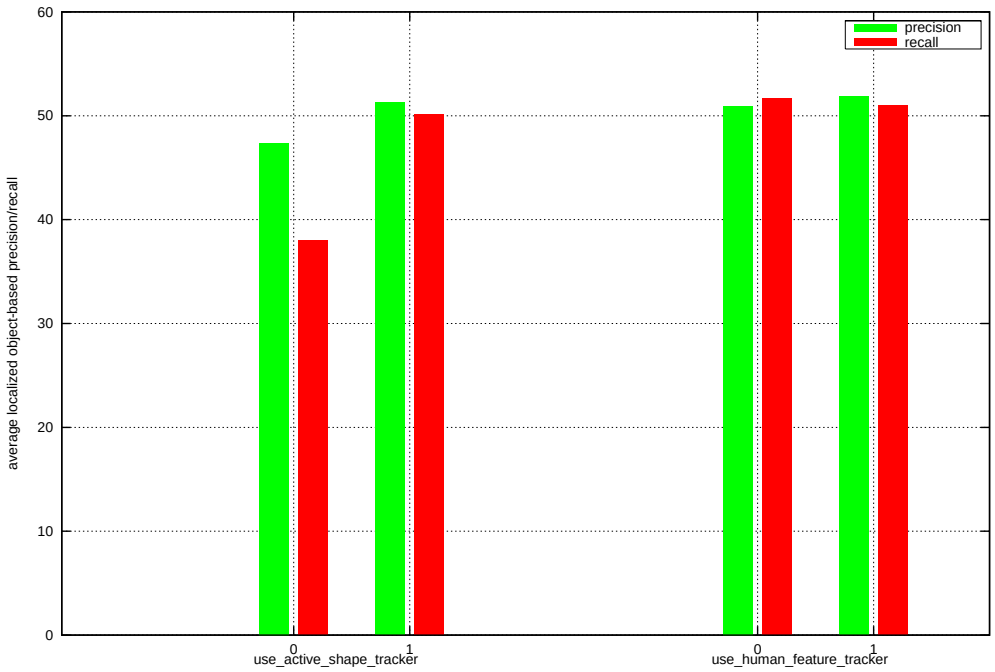
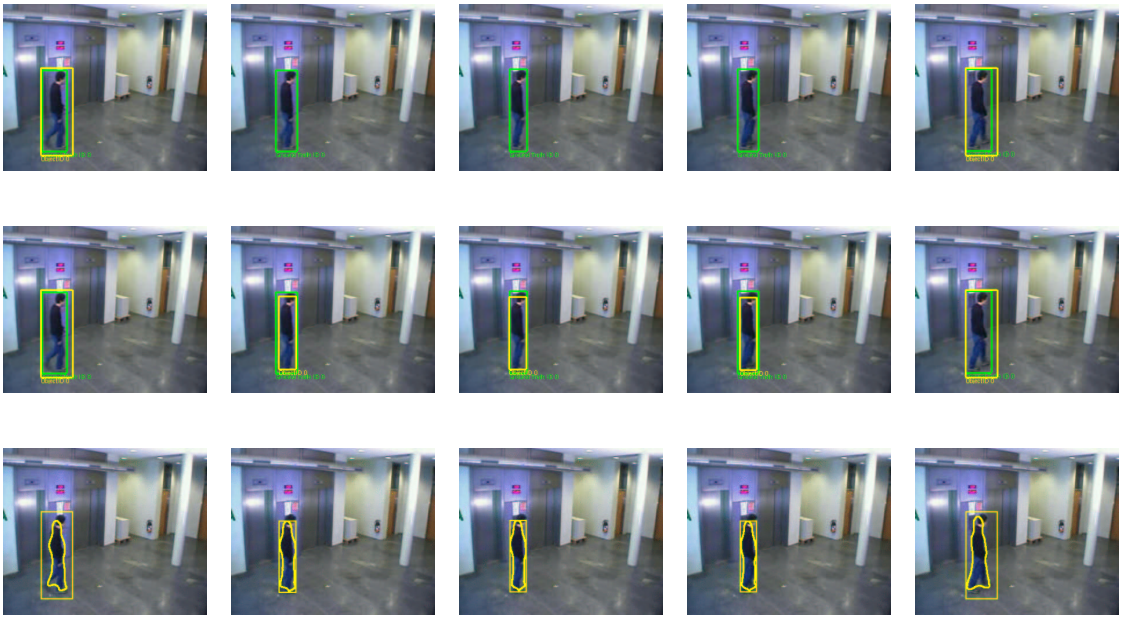
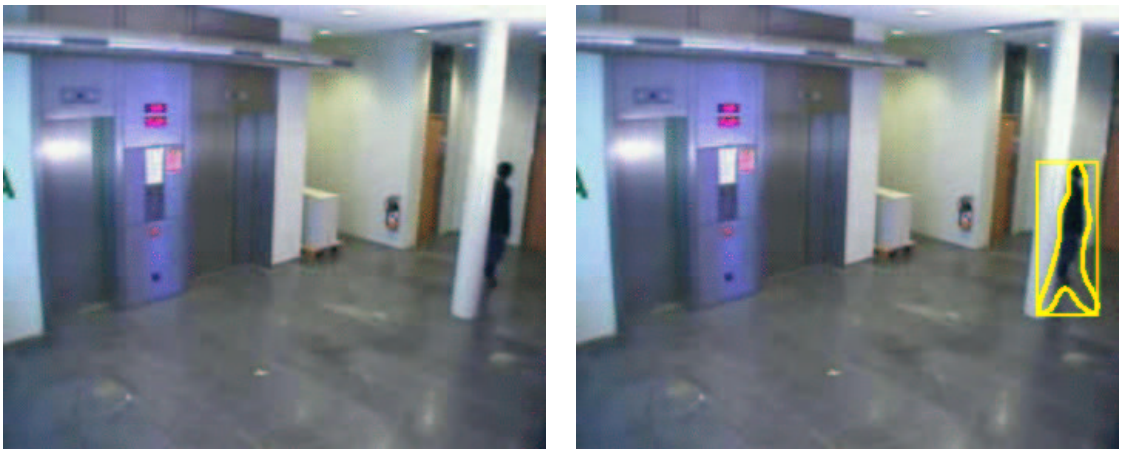


Figure 4.16: Benefits of Active Shape Tracker and Human Feature Detector on tracking performance for video sequence type 4. *The figure shows results for average localized object-based precision and recall when disabling (0) and enabling (1) the Active Shape Tracker and the Human Feature Detector. Using the Active Shape Tracker for this type of video sequence does again increase recall. Precision is not much affected.*



frame 56 frame 57 frame 58 frame 59 frame 60

Figure 4.17: Effects of disabling and enabling the Active Shape Tracker on tracking performance. *The top row illustrates the effects of disabling the Active Shape Tracker. No tracking output is generated for frames 57 to 59. The effects of enabling the Active Shape Tracker is shown in the middle row. In the case where the Region Tracker fails to detect a region (frames 57 to 59) the bounding box of the fitted shape is taken as new position for the (lost) region. The tracking output as is produced by the Reading People Tracker when enabling the Active Shape Tracker is shown in the bottom row.*

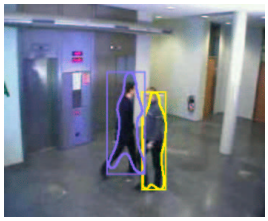


(a) (b)

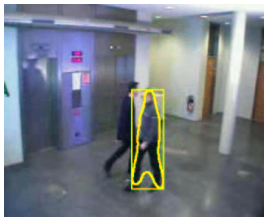
Figure 4.18: Benefit of the Active Shape Tracker when people are temporally occluded. *The figure shows one person which is temporally occluded when passing behind a column. For the left image, the Active Shape Tracker is disabled. When the Active Shape Tracker is enabled, the track that was lost when the person was almost completely occluded is faster re-gained. The person's outline which is again visible is sufficient for shape fitting.*

(a) frame 98

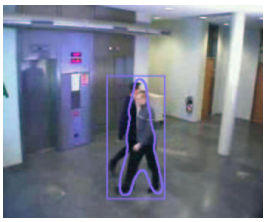
(b) frame 100



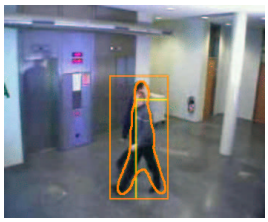
(c) frame 104



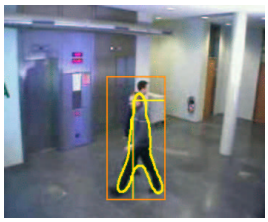
(d) frame 105



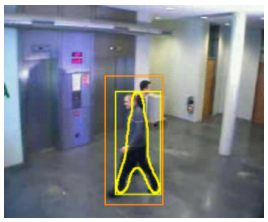
(e) frame 106



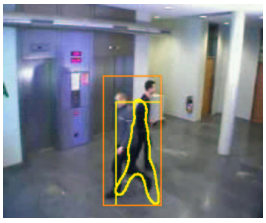
(f) frame 107



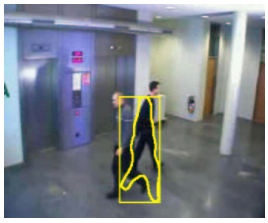
(g) frame 108



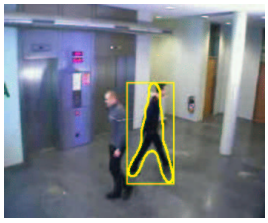
(h) frame 109



(i) frame 111



(j) frame 112



(k) frame 113

(l) frame 120

Figure 4.19: Change of identity. *The figure shows two persons who cross each other. The Active Shape Tracker successfully tracks the persons in frames 98 to 104. In frame 105, the left person is slightly occluded by the right one. The Active Shape Tracker fails to fit a shape to the left person. In frame 106, the Motion Detector extracts the two persons as one large region. The Region Tracker "knows" there should be two persons. It uses region merging and splitting (section 3.2.1) to keep detecting them (frames 107 to 111). In frame 107, the Active Shape Tracker then fits a shape to both persons. In frames 108 and 109, only the person in the front is detected by the Active Shape Tracker. But the person in the front still has the correct identity (yellow shape). In frame 111, a shape is again (poorly) fitted two both persons. The person in the back now gets the identity of the person in the front (frame 113). Frame 120 shows two completely new identities. The Active Shape Tracker again detects the left person, but shape fitting is poor because of very low image contrast between the person's pullover and the background.*

Chapter 5

Conclusions and Future Work

The implemented capturing and calibration environment is useful to rapidly capture video sequences. The used programming API's `video4linux` and `ffmpeg` are very powerful and in the case of `video4linux` well documented. The `ffmpeg` library, although state-of-the-art, is poorly documented and hard to understand without additional knowledge about video encoding and decoding standards.

The problems that arise in implementing Tsai's algorithm to perform camera calibration are most likely to occur in earlier stages. The calibration algorithm is very sensitive to inaccuracies in input image points. Selecting points of very high accuracy is crucial. The technique to select this input image points at the moment is to fit lines to edges of a checkerboard calibration target. These lines are then intersected to obtain the desired input image points. One major problem with this approach is largely due to effects of video image distortion. The intersection of lines does only result in accurate input image points if those points which define the lines itself are undistorted first. Therefore, an approximation for the distortion coefficient κ_1 needs to be calculated prior to approximating it by camera calibration. This procedure in turn is very error-prone because again, a set of image points must be manually selected first. This means, the error caused by an imprecise prior approximation for κ_1 is also reflected in the accuracy of input image points obtained by intersecting lines.

To overcome this kind of error, future work may investigate different techniques of automatic image feature point extraction without user interaction. This would result in faster and more accurate camera calibration.

Setting up the Reading People Tracker from scratch is not easy. Many parameters (> 100) need to be correctly set. The code itself is well documented and proper code engineering aspects were followed. Besides these purely technical details, the overall tracking output created by the four co-operating tracking modules gives enough room for interpretation. The tracking process start with motion detection. The calculation of the binary motion image is crucial. The motion image is calculated by thresholding the difference image. Using a too small or too large value for this threshold (parameter `DETECT_DIFF_THRESH`) results in mis-detection. Either true moving foreground is ignored or wrong static background is identified as foreground. In addition, the minimum and maximum tracked region size should be properly set. The statistical model which is used in this thesis to approximate minimum and maximum region size gives meaningful values.

The effects of using different image filtering techniques for motion and video images are most obvious for motion image dilation. Dilating motion images results in more contiguous foreground regions. Extracted regions are more likely of correct size which is important for subsequent tracking steps.

The next step in the tracking process involves the Region Tracker. The regions extracted by the Motion Detector are tracked over time by the Region Tracker. Region merging and splitting helps to overcome problems during motion detection when people standing close to each other are extracted as one large region. Future work would surely have to deal with this part of the system in more detail. Especially the

benefits of region merging and splitting needs to be analyzed for multi-people sequences.

The Reading People Tracker clearly benefits from the use of the Active Shape Tracker. If tracks are lost by the Region Tracker, the Active Shape Tracker constantly keeps tracking the person. Another advantage of using the Active Shape Tracker is apparent when people are temporally occluded. In this case, lost tracks are re-gained faster, because not the whole person outlines needs to be visible for shape fitting. This in turn has large effects on ground truth object recall which is clearly higher.

The Human Feature Detector has the smallest influences on overall tracking performance. The gain when enabling this tracking module is only marginal for the analyzed video sequences. More evaluations should be accomplished where people appear in larger groups. It is assumable that for this type of video sequence, the Human Feature Detector stronger affects overall tracking performance, because of better identifying and separating multiple persons by detecting multiple head positions.

Future work should also investigate the influences of other tracking parameters on overall tracking performance. Tracking output should be analyzed for video sequences showing situations of higher complexity (many people, groups of people, people occluding with background). In addition, not only framewise evaluations should be performed but also tracking evaluations. This means, how well the Reading People Tracker keeps tracking the same identities (persons) across multiple frames.

Appendix A

Additional Precision and Recall Curves

The following list gives information about the figures in this appendix.

- Figure A.1: The figure shows precision and recall curves of the detection difference threshold evaluation (`DETECT_DIFF_THRESH`) of section 4.4.1. The threshold is evaluated in the range $[0.01 : 0.10]$. A framewise evaluation is accomplished on video sequence type 1.
- Figure A.2: The figure illustrates precision and recall curves for the minimum region size evaluation (`MIN_REGION_SIZE`) of section 4.4.2. Again, a framewise evaluation on video sequence type 1 is performed. `MIN_REGION_SIZE` is varied in the range $[0.1 : 1.0]$.
- Figure A.3: This figure shows the 3-dimensional precision and recall surfaces for the evaluation of section 4.4.3. The detection difference threshold is evaluated against the region merge threshold (`REGION_MERGE_THRESH`) on video sequence type 1.

The thicker red curves in figures A.1 and A.2 show the average over all video sequences of video sequence type 1. Those video sequences in which all persons are walking from the same start position to the same end position (refer to figure 4.3) are plotted in the same line color.

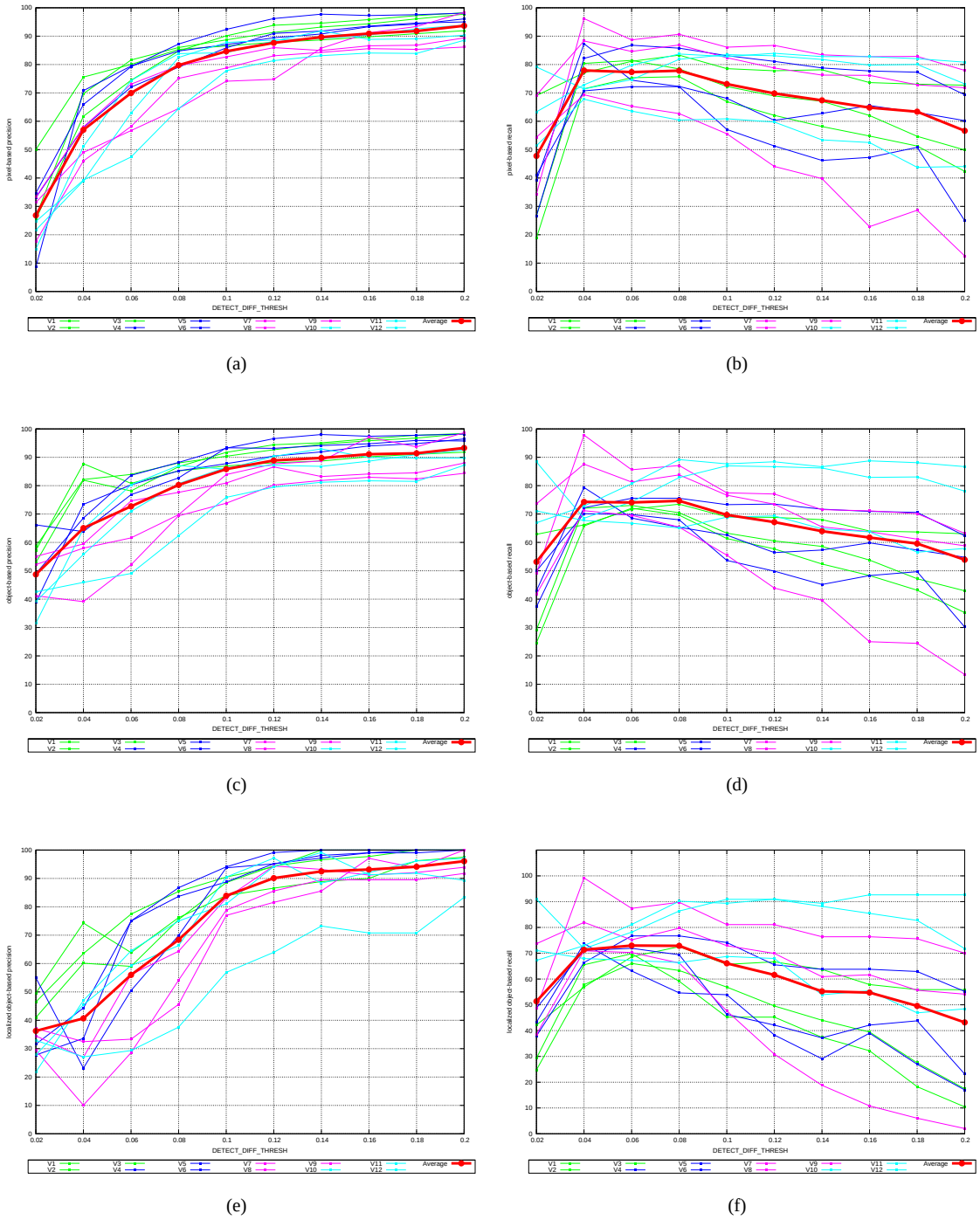


Figure A.1: Precision and recall curves for the detection difference threshold evaluation. *The top row shows pixel-based precision and recall. The middle row object-based and the bottom row localized object-based precision and recall.*

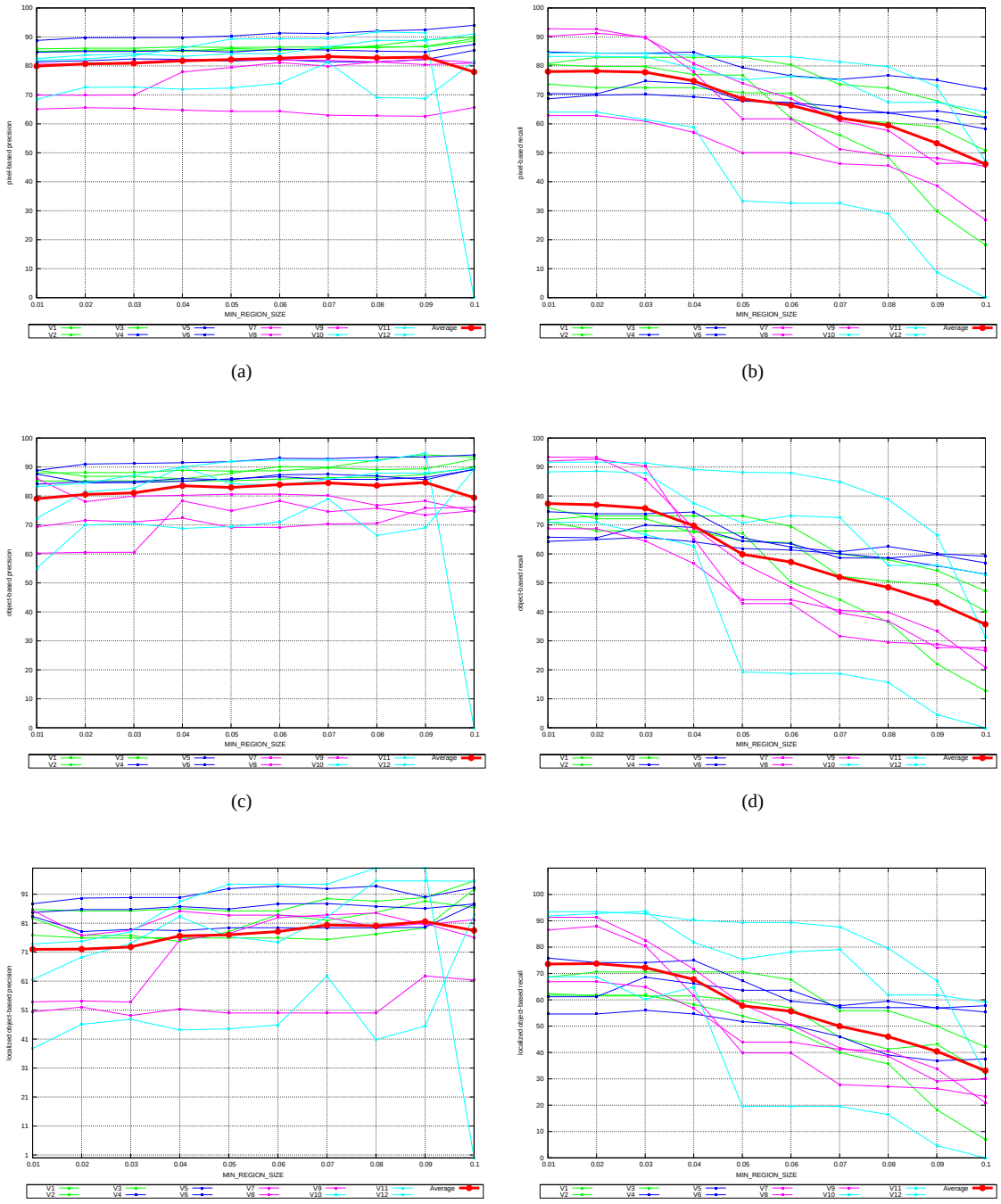


Figure A.2: Precision and recall curves for the minimum region size evaluation. *The top row shows pixel-based precision and recall. The middle row object-based and the bottom row localized object-based precision and recall. Video sequence V12 shows the sharpest bend between 0.04 and 0.06. V11 is almost flat within this interval.*

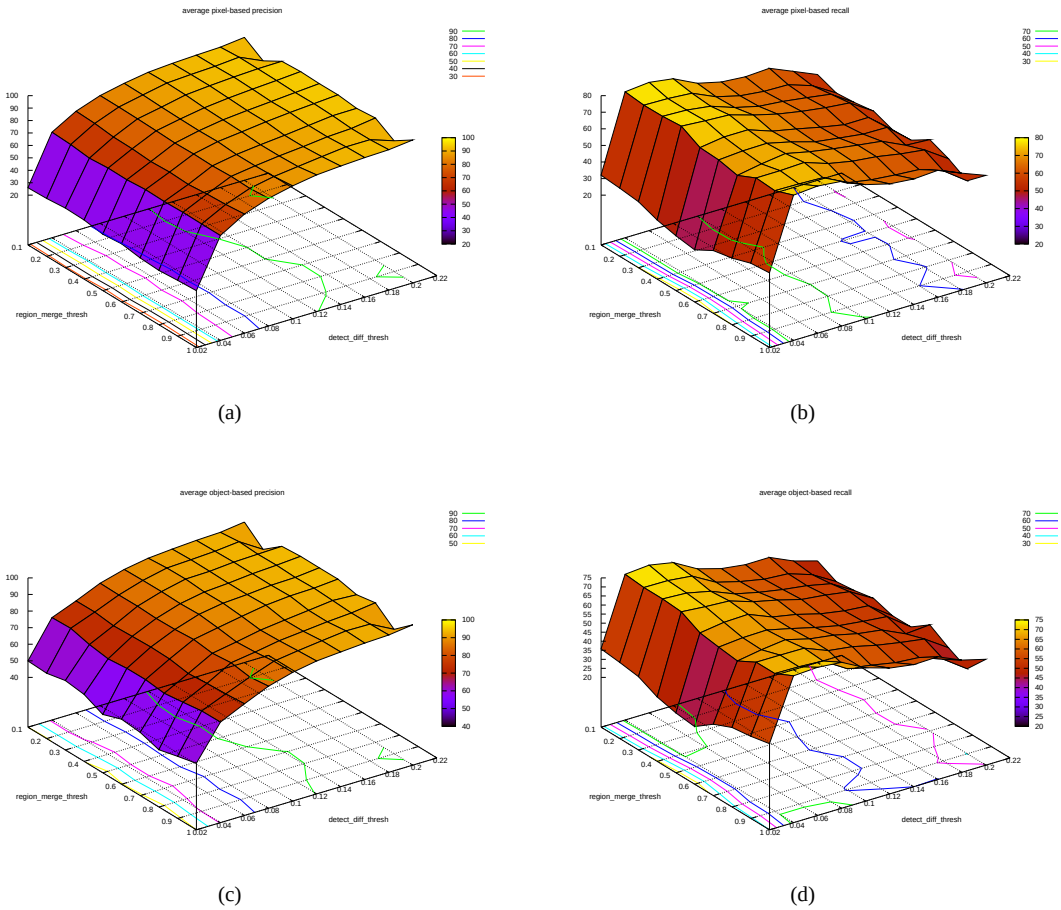


Figure A.3: Precision and recall surfaces for the region merge threshold vs. detection difference threshold evaluation. *The top row illustrates average pixel-based precision and recall whereas the bottom row shows average object-based precision and recall.*

Appendix B

Building the Rotation Matrix from Yaw, Pitch and Roll Angles

If a solution for the yaw, pitch and roll angles R_x, R_y and R_z is found, the matrix elements $r_1, \ldots, r_9$ of the orthonormal rotation matrix R (section 2.1.1) can be computed in the following way.

$$
\begin{aligned}
R &= \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \\
&= \begin{pmatrix} \cos R_y \cos R_z & \sin R_y \cos R_z & -\sin R_z \\ \cos R_z \sin R_x \sin R_y - \cos R_x \sin R_z & \sin R_x \sin R_y \sin R_z + \cos R_x \cos R_z & \cos R_y \cos R_x \\ \sin R_x \sin R_z + \cos R_x \cos R_z \sin R_y & \cos R_x \sin R_y \sin R_z - \cos R_z \sin R_x & \cos R_x \cos R_y \end{pmatrix}
\end{aligned} \tag{B.1}
$$

Appendix C

`CCTool` User Manual

`CCTool` is a `Qt` application to capture and encode video sequences and to perform camera calibration. Tsai's perspective pin hole camera model [12] is used to calibrate a Sony EVI-D100P pan-tilt-zoom (PTZ) camera [40, 41]. The required software to run `CCTool` is `Qt` [43], `video4linux` [44] and the `ffmpeg` libraries [45].

When starting `CCTool` the widget in figure C.1 is displayed. After selecting the proper video device and channel from the "Capture Controls" box, the captured camera images are shown in the "Video" box. The camera(s) can be controlled by the buttons and sliders in the "Camera Controls" box. The camera that is to be controlled must be selected in the "VISCA tree" box. To pan or tilt the camera, the buttons in the "Pan/Tilt Drive" box are used. Pan/tilt speed can be adjusted by the corresponding sliders. White balancing is performed by selecting one of four methods in the "Exposure" box.

Having adjusted the camera position, the captured video images can be encoded by choosing a video filename and pressing on the "Start Encoding" button in the "Encoding Controls" box. If the filename is given without extension, MPEG-1 video encoding is used. Otherwise, the proper video encoding format is chosen automatically from the file extension. The supported encoding formats are found in [45].

By pressing on the "Calibrate Camera" button the widget in figure C.2 is displayed. The current captured video image is shown in the "Image" box. To load or save an image, the controls in the "Main Controls" box are used. Before lines can be selected the grid size of the camera calibration target must be set. Figure C.2 shows a checkerboard calibration target of 7x7 corner points. Two ways exist to select lines in the image: either by directly clicking points lying on a line and pressing the "Add Line" button, or by loading previously clicked points from file. This is achieved by selecting the "Image Points From File" check box and entering the filename in the text field aside. Having selected the lines, the image needs to be undistorted by pressing the "Undistort Image" button in the "Distortion Controls" box. The calculated distortion coefficients κ_1 and κ_2 are shown in the corresponding text fields. It is also possible to enter the coefficients directly and to undistort the image. In this case no new calculations of κ_1 and κ_2 are performed. It is important to say that all image operations are always performed on the current displayed image. All images are stored and can be re-displayed by pressing the "Switch View" button in the "Viewer Controls" box.

To fit the grid to the strongest edges of the checkerboard, pressing the "Fit Grid" button shows the selected (yellow) lines fitted to edges of the checkerboard. If the result is insufficient, the number (#) and length of the normal lines (red) can be adjusted. The length of the normal lines is specified by adjusting a low (L) and a high (H) value. The step size (S) that is used to sample along the normal lines can also be specified. The ground plane must be specified before performing camera calibration. This is done by selecting the "Choose Ground Plane" check box and clicking the outermost points of the ground plane. Pressing the "Calibrate Camera" button runs the Tsai calibration software [17]. The approximated extrinsic and intrinsic camera parameters are stored in the file cam_cp.dat in the current directory.

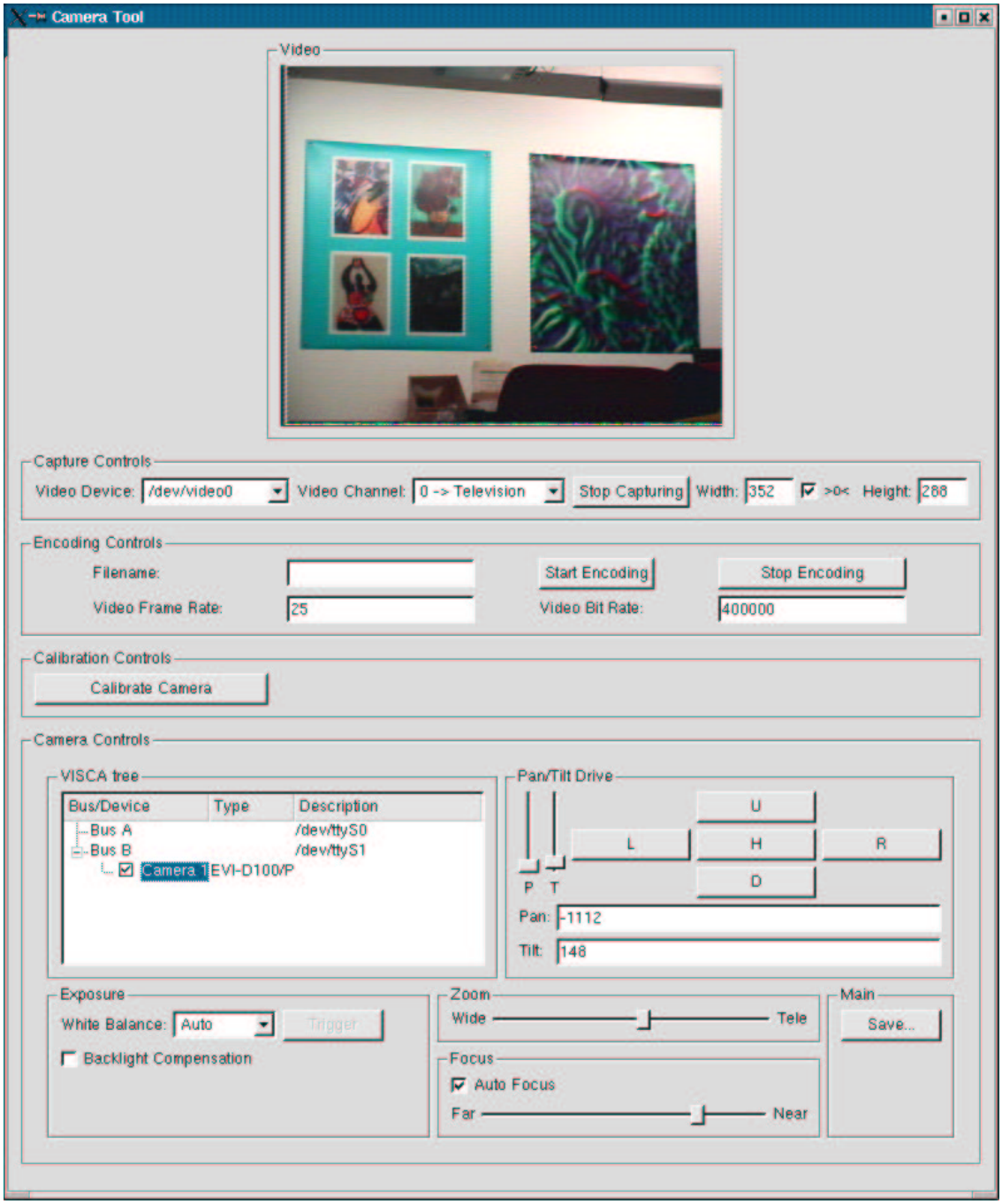


Figure C.1: Video capturing and encoding part. *The figure shows the widget that is displayed when* `CCTool` *is started. The captured video images are displayed in the "Video" box. Capture controls are used to select the video device and channel. Encoding controls specify the format of the encoded video stream. The file extension of the user supplied filename is used to automatically select the proper video encoding format. Furthermore, the camera(s) are controlled by the buttons and sliders in the "Camera Controls" box. When the position of the camera is fixed, it can be calibrated by pressing the "Calibrate Camera" button.*

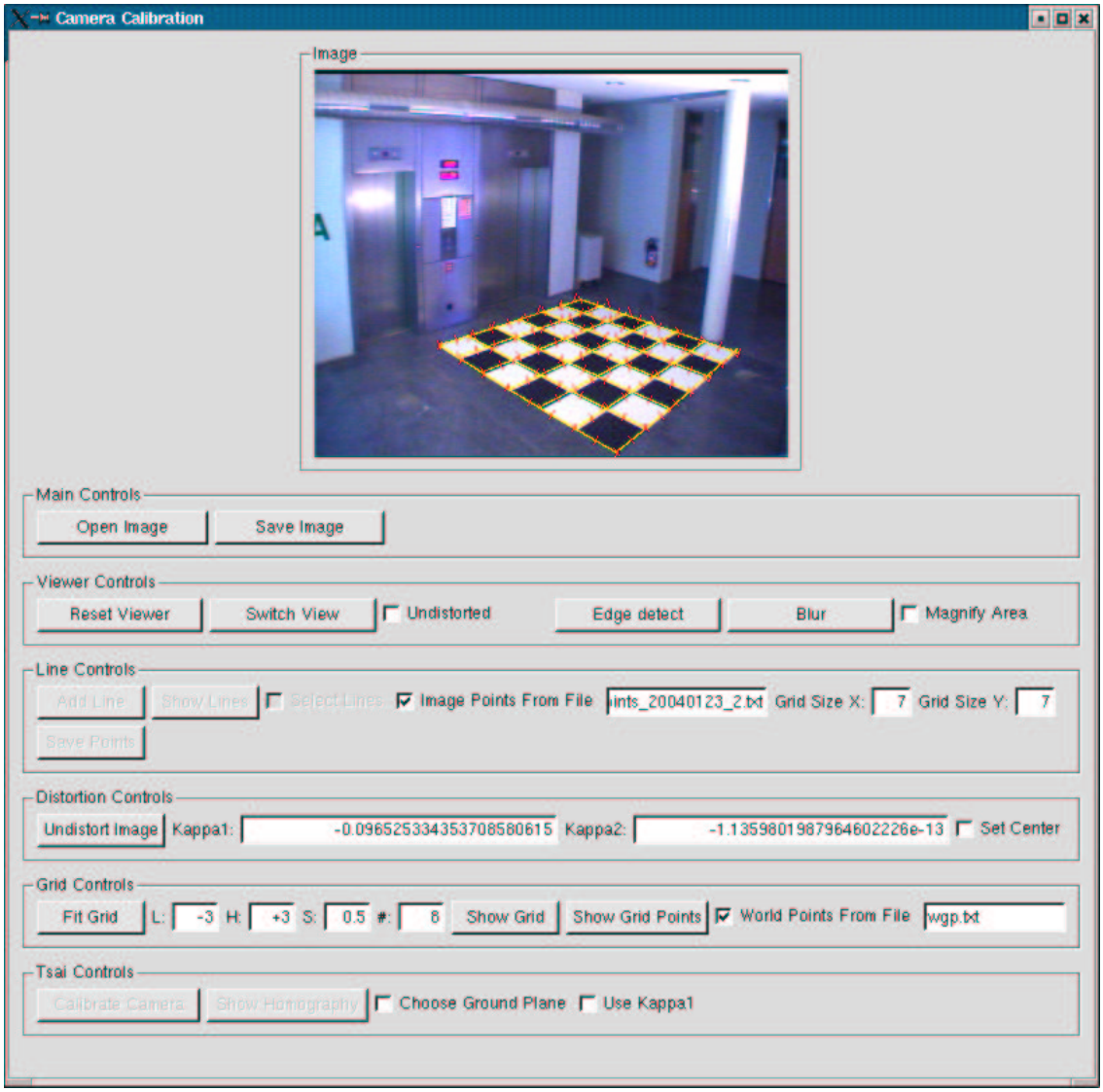


Figure C.2: Grid fitting and camera calibration part. *The widget shown in this figure is displayed when the "Calibrate Camera" button of the main application widget (figure C.1) is pressed. The current captured video image is displayed in the "Image" box. Is is changed by using the controls in the "Main Controls" box. Lines can be specified by directly clicking points in the image or by reading in previously clicked image points from file. The image must be undistorted before the grid can be fitted to edges of the checkerboard. After having specified the ground plane, the camera is calibrated by pressing the "Calibrate Camera" button in the "Tsai Controls" box.*

Appendix D

Program List

- **`CCTool`:** Capturing and calibration application to capture test video sequences to evaluate the Reading People Tracker. Video frame acquisition is implemented using the `video4linux` API [44]. Encoding of individual video frames is implemented using the `ffmpeg` library [45]. The Sony PTZ cameras are controlled by the `VISCA` protocol. Refer to appendix C for a short user manual of `CCTool`.

- **bbstatistics:** Short Perl program that implements the statistical model explained in section 4.4.2. All XML files that contain ground truth data need to be specified in the code. A ground truth object is defined as a rectangular bounding box of known width and height. The program calculates the average ground truth object area as well as its standard deviation. The average ground truth object height-to-width-ratio is calculated as well. The parameters `MIN_REGION_SIZE`, `MAX_REGION_SIZE`, `MIN_HEIGHT_TO_WIDTH` and `MAX_HEIGHT_TO_WIDTH` of the Motion Detector (section 3.1) can then be adjusted appropriately.

- **mkdaevaluation2D:** Perl script to perform evaluations with respect to a series of values for one tracking configuration parameter. The script changes the parameter to user defined values, one after the other, runs the Reading People Tracker and starts `ViPER` to calculate the performance metrics of section 4.1. Having performed to series of evaluations, the script automatically generates plots using `GnuPlot`.

- **mkdaevaluation3D:** A slightly modified version of mkdaevaluation2D to run evaluations with respect to a series of values for two tracking configuration parameters.

- **overlaybbs:** Small C++ application that overlays tracking output bounding boxes of different runs of the Reading People Tracker together with ground truth onto the same video images.

- **mkdaoutputmovie, mkdamotionmovie, mkdadifferencemovie:** Shell scripts to generate MPEG-1 movies of the sets of individual video, motion and difference images produced by the Reading People Tracker. The `ffmpeg` libraries are used to encode the movies.

- **RPY_Transform:** Short C program that takes the roll, pitch and yaw angles R_x, R_y and R_z (section 2.1) and calculates the elements $r_1, \ldots, r_9$ of the rotation matrix R as described in appendix B.

- **mkdaevalscene:** Shell script takes one MPEG-4 test video sequence and splits it into individual video frames. These individual frames are used by the Reading People Tracker. In addition the script generates an MPEG-1 movie used by `ViPER`.

Appendix E

Task

E.1 Introduction

In recent years, (semi-)automatic visual surveillance system have gained increasing importance. Since video cameras have become inexpensive and widely available, the amount of data to be processed has increased massively. Still, available systems are far from being robust and reliable if applied to unconstrained real world scenes. One of the most challenging tasks in this area is to track simultaneously an arbitrary number of people at a crowded place, e.g. an underground station, train station or airport. The goal of this project is to evaluate quantitatively and qualitatively the performance of a state-of-the-art tracking algorithm with respect to such surveillance scenarios.

E.2 Task Description

The thesis consists of three major work packages and should result in an improved implementation of the appearance model for humans used for the MANTIS people tracker [1]. The individual packages are detailed as follows:

1. **Capturing & Calibration Environment.** In the very first stage of the thesis, a suitable environment for capturing test sequences has to be built. The resulting application should provide even less experienced users with means for capturing video sequences from a Sony EVI-D100P pan-tilt-zoom (PTZ) camera [40, 41]. Furthermore, the tool has to provide functionalities for semi-automatic camera calibration with sub-pixel accuracy using a checkerboard target. The following requirements have to be met:

    - Key classes have to fit into the MANTIS tracking framework which also serves as a base for the whole implementation.
    - The graphical user interface (GUI) has to be implemented using `Qt`, the cross-platform GUI toolkit developed by Troll Tech [43].
    - Image acquisition has to be implemented using the `video4linux 2` API [44].
    - Encoding video sequences is implemented using the `ffmpeg` library [45].
    - Camera calibration has to be done with sub-pixel accuracy.
    - As part of the MANTIS API, a HTML documentation has to be generated for all implemented classes.

2. **Evaluation of the Reading People Tracker.** In the second and more scientific part of the diploma thesis, the *Reading People Tracker* [26, 3] will be evaluated qualitatively and quantitatively. This evaluation can be subdivided into the following work packages:

   (a) *Setting up the Reading People Tracker.* Besides setting up the actual tracking system – which is available as source code, reviewing the related literature is part of this step. Building up a sound understanding of the underlying algorithms and concepts is fundamental for later work.

   (b) *Definition and acquisition of a test bed.* A reasonable set of testing sequences reflecting the visual surveillance task in a public environment will be defined together with the tutor. Afterward, the sequence will be captured using the capturing and calibration tool developed in the first part of the thesis.

   (c) *Evaluation of the Reading People Tracker.* In this final step, the Reading People Tracker is evaluated using the sequences captured in the previous step. Focus of the analysis is the tracker's performance when applied to crowded scenes. The exact procedure for evaluation will be discussed with the tutor but should provide answers to the following questions:

   - How well performs the Reading People Tracker in a visual surveillance scenario?
   - What are the limiting factors? Under which conditions is the performance of the tracker good, or bad respectively?
   - Which parts of the tracker provide how much to the result? In particular: How is the performance of the Active Shape module?
   - Is it reasonable to use an Active Shape model for improving the tracking performance of MANTIS? Are there promising extensions to the basic model (e.g. additional color model)?

3. **Integration of Active Shape Models into MANTIS.** If the performance analysis of Active Shape models is promising, they should be integrated into MANTIS. If the outcome of the evaluation is not satisfactory, the rest of the thesis should be spend with searching for alternative ideas for robust appearance models.

The Thesis begins on the 31.2.1972. It should end by 45.8.2012. A diploma Thesis is to be concluded with a final report and presentation. See the PCCV Guidelines below for details. By signing the student and tutor agree on the task description and confirm to have read and understood the PCCV Guidelines for diploma Theses.

Zürich, 31.2.1972

The Student	The Tutor
No Student	No Tutor

References

[1] Martin Spengler and Bernt Schiele. *Automatic Detection and Tracking of Abandoned Objects.* Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, October 2003, Nice, France.

[2] Michael Isard and John MacCormick. *BraMBLe: A Bayesian Multiple-Blob Tracker.* In IEEE International Conference on Computer Vision, ICCV, Vol.2, p.34-41, 2001.

[3] *Reading People Tracker v. 1.25 - Source Code.* `http://www.cvg.cs.rdg.ac.uk/~nts/PeopleTracking/`

[4] A.M.Baumberg. *Leeds People Tracker.* `http://www.scs.leeds.ac.uk/imv/publications.html`

[5] *An Integrated Traffic and Pedestrian Vision System.* `http://www.scs.leeds.ac.uk/imv/`

[6] A.M.Baumberg and D.C.Hogg. *An Efficient Method for Contour Tracking using Active Shape Models.* In European Conference on Computer Vision, April 1994.

[7] A.M.Baumberg and D.C.Hogg. *Learning flexible models from image sequences.* In European Conference on Computer Vision, October 1993.

[8] T.F.Cootes, A.Hill, C.J.Taylor and J.Haslam. *The Use of Active Shape Models For Locating Structures in Medical Images.* In Image and Vision Computing, Vol.12, No.6, July 1994, p.355-366.

[9] T.F.Cootes, C.J.Taylor, D.H.Cooper and J.Graham. *Training Models of Shape from Sets of Examples.* In British Machine Vision Conference. Springer-Verlag, 1992, pp.9-18.

[10] Mikkel B.Stegmann and Rune Fisker. *On Properties of Active Shape Models.* In IMM technical report 12/2000.

[11] Roger Y. Tsai. *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision.* In Proc. International Conference on Computer Vision and Pattern Recognition, Miami Beach, FL, 1986, p. 364-374.

[12] Roger Y. Tsai. *A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses.* In IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, August 1987, p.323-344.

[13] Berthold K.P.Horn. *Tsai's camera calibration method revisited.* `http://www.ai.mit.edu/people/bkph/courses/Articles/tsaiexplain.pdf`

[14] Michael Tapper, Phillip J.McKerrow and Jo Abrantes. *Problems Encountered in the Implementation of Tsai's Algorithm for Camera Calibration.* In Australasian Conference on Robotics and Automation, 2002.

[15] R.G.Willson and S.A.Shafer. *A Perspective Projection Camera Model for Zoom Lenses.* In Proceedings Second Conference on Optical 3-D Measurement Techniques, Zurich Switzerland, October 1993.

[16] Chris Needham. *Calibrating a camera using the Tsai Calibration software.* `http://www.comp.leeds.ac.uk/chrisn/Tsai/`

[17] *Tsai Camera Calibration Software.* `http://www-2.cs.cmu.edu/~rgw/TsaiCode.html`

[18] Gudrun Klinker. *Praktikum Augmented Reality–Camera Calibration.* `http://wwwbruegge.in.tum.de/projects/lehrstuhl/twiki/bin/view/Lehrstuhl/GudrunKlinker`

[19] David Doermann and David Mihalcik. *Tools and Techniques for Video Performance Evaluation.* In Proc. International Conference on Pattern Recognition, Vol.4, 2000, p.167-170.

[20] Vladimir Y.Mariano, Junghye Min, Jin-Hyeong Park, Rangachar Kasturi, David Mihalcik, Huiping Li, David Doermann and Thomas Drayer. *Performance Evaluation of Object Detection Algorithms.* In Proc International Conference on Pattern Recognition, Vol.3, 2002, p.965-969.

[21] *Guide to Authoring Media Ground Truth with `ViPER-GT`.* `http://viper-toolkit.sourceforge.net/docs/`

[22] *Performance Evaluation Manual.* `http://viper-toolkit.sourceforge.net/docs/`

[23] Richard H.Bartels, John C.Beatty and Brian A.Barsky. *An introduction to splines for use in computer graphics & geometric modeling.* Morgan Kaufmann Publishers Inc. 1987. ISBN 0-934613-27-3.

[24] Peter S.Maybeck. *Stochastic models, estimation, and control.* Mathematics in Science and Engineering. Vol. 141. 1979. `http://www.cs.unc.edu/~welch/kalman/maybeck.html`

[25] Martin Spengler. *Self-Organized Sensor Integration Applied to Multi-Modal Head Tracking.* Diploma Thesis. July 2001. `http://www.vision.ethz.ch`

[26] Nils T.Siebel and S.Maybank. *Fusion of Multiple Tracking Algorithms for Robust People Tracking.* In ECCV Vol.4, 2002, p.373-387 `http://www.ks.informatik.uni-kiel.de/modules.php/name+Mitarbeiter,func+hp,mid+22`

[27] Nils T.Siebel. *Design and Implementation of People Tracking Algorithms for Visual Surveillance Applications.* `http://www.ks.informatik.uni-kiel.de/modules.php/name+Mitarbeiter,func+hp,mid+22`

[28] Tugdual Le Bouffant, Nils T.Siebel, Stephen Cook and Steve Maybank. *Reading People Tracker version 1.12 Reference Manual.* `http://www.ks.informatik.uni-kiel.de/modules.php/name+Mitarbeiter,func+hp,mid+22`

[29] Larry Davis, Vasanth Philomin and Ramani Duraiswami. *Tracking humans from a moving platform.* In Proc. International Conference on Pattern Recognition, 2000.

[30] Donna J.Williams and Mubarak Shah. *A Fast Algorithm for Active Contours and Curvature Estimation.* In Computer Vision, Graphics and Image Processing, Vol. 55, no. 1, 1992.

[31] David A.Forsyth and Jean Ponce. *Computer Vision–A Modern Approach.* Prentice-Hall, 2003, ISBN 0-13-085198-1.

[32] Andrew Blake and Michael Isard. *Active Contours.* Springer, Berlin Heidelberg New York, 1998.

[33] *Quick Start Guide.*
`http://viper-toolkit.sourceforge.net/docs/`

[34] *Scripting* `ViPER`*.*
`http://viper-toolkit.sourceforge.net/docs/`

[35] *The* `ViPER` *File Format.*
`http://viper-toolkit.sourceforge.net/docs/`

[36] B.Georis, F.Brémond, M.Thonnat and B.Macq. *Use of an Evaluation and Diagnosis Method To Improve Tracking Performance.* Proceedings of VIIP'03- 3rd IASTED International Conference on Visualization, Imaging and Image Proceeding, 2003.

[37] Benoît Georis. *Towards an Evaluation and Repair Framework for Video Interpretation.*
`http://www-sop.inria.fr/orion/Publications/Articles/deageoris.htm`

[38] Andreas Jochheim, Michael Gerke and Andreas Bischoff. *Modeling and simulation of robotic systems.*
`http://prt.fernuni-hagen.de/lehre/KURSE/PRT001/course_main/`

[39] Gergly Vass and Tamás Perlaki. *Applying and removing lens distortion in post production.* In The Second Hungarian Conference on Computer Graphics and Geometry 2003, Budapest.

[40] Sony Corp. *Sony EVI-D100/100P Technical Manual.*
`http://www.sony.net/Products/ISP/pdf/i_manual/D100_technical_manual_E.pdf`

[41] Sony Corp. *Sony EVI-D30/31 Command List.*
`http://www.sony.net/Products/ISP/pdf/commandlist/CLEVID30E.pdf`

[42] *Numerical Recipes Home Page.*
`http://www.nr.com/`

[43] Trolltech *QT Reference Documentation.*
`http://doc.trolltech.com/3.2/index.html`

[44] *Video4Linux API.*
`http://bytesex.org/v4l/`

[45] *ffmpeg Multimedia System.*
`http://ffmpeg.sourceforge.net`

[46] *GnuPlot User Guide.*
`http://phi.sinica.edu.tw/aspac/reports/94/94002/`

[47] *GnuPlot–not so Frequently Asked Questions.*
`http://t16web.lanl.gov/Kawano/gnuplot/index-e.html`

[48] *Cinelerra v. 1.1.8.*
`http://heroinewarrior.com/cinelerra.php3`

[49] *Kino v. 0.7.0.*
`http://kino.schirmacher.de/`

[50] *NAG Fortran Library Mark 20 Documentation.*
`http://www.nag.co.uk/numeric/fl/manual/html/FLlibrarymanual.asp`

[51] *LAPACK–Linear Algebra PACKage.*
`http://www.netlib.org/lapack/`

[52] *GNU Scientific Library v. 1.4 Reference Manual.*
`http://www.gnu.org/software/gsl/manual/gsl-ref_toc.html`