Never mistake motion for action.
(Ernest Hemingway)

Chapter 4

Feature-Based Motion II: Parameter Estimation

This chapter describes the algorithm for computing the parameters of the projective motion model, based on the feature-correspondences that we obtained in the last chapter. We construct the algorithm step by step, starting with a simpler affine motion model, prior to considering the projective motion model. Whereas the parameter estimation for affine motion can be realized with linear least-squares, an equivalent problem formulation for projective motion leads to a non-linear optimization problem. Furthermore, we study the case of images with multiple independent motions in the same frame. To extract the dominant motion model in this case, we apply the RANSAC algorithm, which is a robust estimation algorithm that is not affected by outlier data. An evaluation of the robust estimation algorithm shows that the accuracy of the results in practice is worse than expected from a theoretic evaluation. However, after analyzing this discrepancy, we propose a modification to reach the theoretical performance.

4.1 Introduction

Chapter 2 presented how camera motion can be described with a projective motion model. In this chapter, we address algorithms to solve the inverse problem of estimating the model parameters from a set of point-correspondences.

We describe the algorithm in two steps. First, we assume that the video sequence shows only background motion without foreground objects that move in a different direction. This allows us to include all the correspondences in the parameter estimation. In Section 4.3, we describe an enhanced algorithm that generalizes the algorithm such that foreground motion and outlier data are excluded from to estimating the motion parameters. The applied algorithm is the RANSAC (Random Sample Consensus) [73] algorithm, which is a probabilistic algorithm only succeeding with a certain probability. Our experiments show that the practical performance does not reach the theoretically predicted probability of success. We will derive an explanation and propose a modification to increase the robustness in Section 4.3.3.

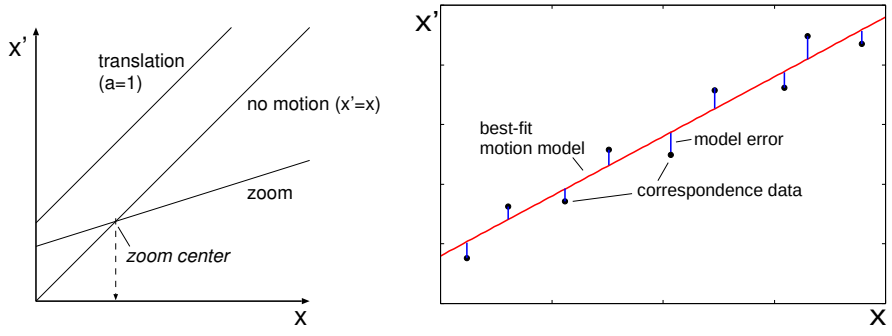
Besides the RANSAC algorithm, other robust estimation algorithms have been proposed. We conducted experiments with the LTS (Least Trimmed Squares) and LMedS (Least Median of Squares) algorithms. These algorithms are explained and compared to RANSAC in Appendix C.

4.2 Computing motion model parameters

In the following two sections, we will first consider the estimation problem for the affine motion model, since this can be solved with linear least-squares. After that, we will discuss the projective motion model in Sections 4.2.3 and 4.2.4.

4.2.1 One-dimensional affine motion

Let us first illustrate the principles for simple one-dimensional affine motion. If we denote the positions in the first (one-dimensional) picture by x_i and the corresponding position in the second picture by x'_i, we can formulate the affine motion model as $x'_i = a \cdot x_i + b$. Each selection of model parameters defines a line in an x, x' diagram which illustrates the corresponding positions between x and x'. The two possible types of motion which are possible with this simple transform are depicted in Figure 4.1(a). The two types of motion are *translatorial motion*, which is specified using the b parameter and *zoom*, which is specified with the a parameter.



(a) (b)

Figure 4.1: *One-dimensional affine motion. The horizontal axis shows the position x in the first picture while the vertical axis shows the corresponding position x' in the second picture. (a) Lines in the x, x' diagram depict the motion between x and x'. (b) From the feature-correspondence step, we get a set of (noisy) correspondences $\{x \leftrightarrow x'\}$ which are drawn as dots. A least-squares fit is carried out by minimizing the sum of model errors $|g(x_i) - x'_i|$.*

The center of the zoom is at the position where the model line crosses the identity line $x = x'$.

The parameter-estimation problem is now to obtain an estimate for the parameters a and b, based on a set of measured point-correspondences. We denote a set of points in the first image as $\{x_i\}$ and their corresponding points in the second image as $\{\hat{x}_i\}$. Since the point measurements are not exact, we cannot assume that they will all fit perfectly to the motion model. Hence, the best solution is to compute a least-squares fit to the data. We consequently define the model error as the sum of squared distances between the measured positions $\hat{x}_i$ and the positions obtained from the motion model (Figure 4.1(b)). This results in the definition of the model error as $E = \sum_i ((ax_i + b) - \hat{x}_i)^2$. To minimize the model error E, we take its derivatives with respect to the motion parameters

$$\frac{\partial E}{\partial a} = \sum_i 2(ax_i + b - \hat{x}_i)x_i \quad ; \quad \frac{\partial E}{\partial b} = \sum_i 2(ax_i + b - \hat{x}_i), \tag{4.1}$$

and set them to zero. This leads to the two equations

$$\sum_i (ax_i^2 + bx_i - \hat{x}_i x_i) = 0 \quad ; \quad \sum_i (ax_i + b - \hat{x}_i) = 0, \tag{4.2}$$

which can be written in matrix form as

$$\begin{bmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & \sum_i 1 \end{bmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_i \hat{x}_i x_i \\ \sum \hat{x}_i \end{pmatrix}. \tag{4.3}$$

By solving this linear equation system, we can determine the unknown model parameters a and b. Note that the problem solved so far is mathematically identical to the problem of simple linear regression.

4.2.2 Two-dimensional affine motion

In the two-dimensional case, our measurements consist of positions (x_i, y_i) in the first image and corresponding position $(\hat{x}_i, \hat{y}_i)$ in the second image. The position that is obtained by transforming (x_i, y_i) according to the motion model will be denoted as (x'_i, y'_i). Now, it is the 2-D affine motion model

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}, \tag{4.4}$$

for which we want to find a good estimate of the six parameters $\{a_{ik}\}, t_x, t_y$. As a direct generalization of the model error of the one-dimensional case, we can define the model error as: $E_2 = \sum_i (x'_i - \hat{x}_i)^2 + (y'_i - \hat{y}_i)^2$. In a geometrical sense, this is the sum of Euclidean distances between the measured positions in the second frame and the positions to which the features from the first image are transformed (Fig 4.2). Note that this definition assumes that the measurements in the first frame are exact and errors are only made in measuring the position in the second picture. Since this is not true in practice, it is proposed in [85] to use a more symmetric error definition like the *symmetric transfer error* or the *reprojection error*. However, this would lead to a more complicated solution with only very little increase of accuracy. Consequently, we will use the definition of Euclidean error E_2.

To solve for the minimum error E_2, we again take the partial derivatives with respect to the model parameters $a\{_{ij}\}, t_x, t_y$ and set them to zero. This gives the equation system

$$\begin{bmatrix} \sum_i x_i^2 & \sum_i x_i y_i & \sum_i x_i & 0 & 0 & 0 \\ \sum_i x_i y_i & \sum_i y_i^2 & \sum_i y_i & 0 & 0 & 0 \\ \sum_i x_i & \sum_i y_i & \sum_i 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sum_i x_i^2 & \sum_i x_i y_i & \sum_i x_i \\ 0 & 0 & 0 & \sum_i x_i y_i & \sum_i y_i^2 & \sum_i y_i \\ 0 & 0 & 0 & \sum_i x_i & \sum_i y_i & \sum_i 1 \end{bmatrix} \begin{pmatrix} a_{00} \\ a_{01} \\ t_x \\ a_{10} \\ a_{11} \\ t_y \end{pmatrix} = \begin{pmatrix} \sum_i \hat{x}_i x_i \\ \sum \hat{x}_i y_i \\ \sum \hat{x}_i \\ \sum_i \hat{y}_i x_i \\ \sum \hat{y}_i y_i \\ \sum \hat{y}_i \end{pmatrix},$$

which obviously can be solved more easily by splitting the equation system into two independent systems. The first one determines the parameters for

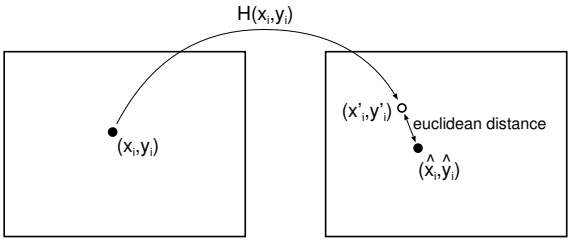


Figure 4.2: *The model error is specified as the Euclidean distance between the detected feature position in the second frame and the ideal feature position according to the motion model.*

the horizontal motion component a_{00}, a_{01}, t_x, while the second one determines parameters a_{10}, a_{11}, t_y for the vertical component.

4.2.3 One-dimensional projective motion

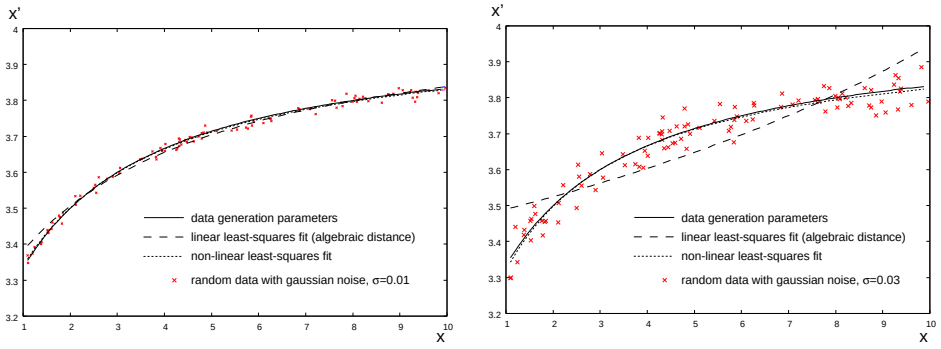
Let us now switch from the affine model to projective motion. We again consider the one-dimensional case first, for which we use a one-dimensional projective motion model in the inhomogeneous representation $x'_i = (a \cdot x + b)/(c \cdot x + 1)$. The most important difference for the estimation is the fact that this motion model is not linear anymore. Would we use the same model error definition as above and proceed with the same approach, we would get a non-linear equation system which is much more difficult to solve. However, we can apply a trick to linearize the equation system by modifying the model error definition. Instead of using the Euclidean distance

$$E_2(a,b,c) = \sum_i (x'_i - \hat{x}_i)^2 = \sum_i \left(\frac{ax_i + b}{cx_i + 1} - \hat{x}_i \right)^2, \tag{4.5}$$

we multiply with the nominator of the motion model and obtain the algebraic error

$$E_a(a,b,c) = \sum_i \left((\frac{ax_i + b}{cx_i + 1} - \hat{x}_i) \cdot (cx_i + 1) \right)^2 = \sum_i \left(ax_i + b - \hat{x}_i(cx_i + 1) \right)^2. \tag{4.6}$$

With this new error definition, we can again compute the partial derivatives and set them to zero to obtain the optimal parameter estimate. After reordering the obtained equations, we can write them as the equation sys-



(a) Low noise. (b) High noise.

Figure 4.3: *Estimation of parameters for the perspective motion model using linear least-squares on a algebraic distance, and using non-linear least-squares on the Euclidean distance.*

tem

$$\begin{bmatrix} \sum_i x_i^2 & \sum_i x_i & \sum_i -x_i^2\hat{x}_i \\ \sum_i x_i & \sum_i 1 & \sum_i -x_i\hat{x}_i \\ \sum_i x_i^2\hat{x}_i & \sum_i x_i\hat{x}_i & \sum_i -x_i^2\hat{x}_i^2 \end{bmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum_i x_i\hat{x}_i \\ \sum_i \hat{x}_i \\ \sum_i x_i\hat{x}_i^2 \end{pmatrix}. \qquad (4.7)$$

The use of the algebraic error instead of the Euclidean error enables a more easy computation of the parameters, since only a small linear equation system has to be solved. However, the penalty for this simplification of the computation is a reduction of parameter accuracy. Because of the changed definition of our model error, we now optimize a geometrically meaningless algebraic distance. As long as the noise level in the data is low, the difference between both models is small, but it increases with a larger noise variance. This behaviour is illustrated in Figure 4.3, where random sample data was generated for an example model with the parameters $a = 2, b = 3, c = 0.5$. In Fig 4.3(a), the data was distorted by Gaussian noise with $\sigma = 0.01$ and in Fig 4.3(b), a higher noise level of $\sigma = 0.03$ was chosen. It can be seen that the non-linear least-squares fit using squared Euclidean distances closely approximates the internal parameters. The fit using algebraic distances results in a reasonable solution for low noise, but it is strongly biased in the case of high noise.

4.2.4 Two-dimensional projective motion

Let us now extend the one-dimensional case to estimating the parameters of two-dimensional projective motion. Recall that we want to determine the homography matrix $\mathbf{H}$, describing the motion from points $\mathbf{p}_i$ in one frame

to points $\mathbf{p}'_i = \mathbf{H}\mathbf{p}_i$. When estimating the parameters $\{h_{ik}\}$, we have to consider that the parameters are scaling invariant. In the previous section, we adopted the inhomogeneous representation of the motion model. In the two-dimensional case, we pursue a similar approach by assuming that $h_{22} = 1$. As we have seen in Section 2.3.3, this normalization fails for the case where the horizon line includes the coordinate origin, since in that case $h_{22} = 0$. An alternative is to use the overcomplete parameterization and to impose additional constraints like the unit norm $||\mathbf{H}||_F = 1$ where $||\cdot||_F$ is the Frobenius norm. This second approach imposes no restrictions on the transform and thus works in any case. However, it is computationally more complex since it leads to computing a Singular Value Decomposition [85]. For inter-frame motion, usage of the inhomogeneous formulation is no problem because the motion is relatively small. The problem only becomes apparent for large rotation angles (as we will see in Chapter 6).

Recall the normalized perspective motion equations

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1}, \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + 1}. \tag{4.8}$$

Since the definition of an Euclidean error measure $E_2 = \sum_i (x'_i - \hat{x}_i)^2 + (y'_i - \hat{y}_i)^2$ would again lead to a complicated non-linear equation system, we will use an algebraic error in a similar way as in the previous section by defining

$$\begin{aligned} E_a &= \sum_i \underbrace{\left((x'_i - \hat{x}_i)^2 + (y'_i - \hat{y}_i)^2\right)}_{\text{Euclidean distance}} (h_{20}x + h_{21}y + 1)^2 \\ &= (h_{00}x + h_{01}y + h_{02} - \hat{x}_i(h_{20}x + h_{21}y + 1))^2 + \\ &\quad (h_{10}x + h_{11}y + h_{12} - \hat{y}_i(h_{20}x + h_{21}y + 1))^2. \end{aligned} \tag{4.9}$$

Imposing the necessary condition $\partial E_a / \partial h_{ik} = 0$ for a minimum error results in the linear equation system of the form

$$\left(\sum_i \mathbf{A}_i\right) \mathbf{h} = \sum_i \mathbf{b}_i, \tag{4.10}$$

consisting of a sum of matrices $\mathbf{A}_i$ and a sum of vectors $\mathbf{b}_i$ on the right-hand side. Using the abbreviation $\hat{s}_i = (\hat{x}^2 + \hat{y}^2)$, the $\mathbf{A}_i$ and $\mathbf{b}_i$ evaluate

(a) Detected correspondences.

(b) Motion field estimated with linear least-squares.

Figure 4.4: *(a) The correspondences obtained from the previous processing steps. Most of the correspondence vectors are correct, only a few are established between unmatching feature-points. (b) The projective motion-model fitted to the correspondences using linear least-squares with the algebraic distance measure.*

as

$$\mathbf{A}_i = \begin{bmatrix} x_i^2 & x_i y_i & x_i & 0 & 0 & 0 & -x_i^2\hat{x}_i & -x_i y_i \hat{x}_i \\ x_i y_i & y_i^2 & y_i & 0 & 0 & 0 & -x_i y_i \hat{x}_i & -y_i^2 \hat{x}_i \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i \hat{x}_i & -y_i \hat{x}_i \\ 0 & 0 & 0 & x_i^2 & x_i y_i & x_i & -x_i^2 \hat{y}_i & -x_i y_i \hat{y}_i \\ 0 & 0 & 0 & x_i y_i & y_i^2 & y_i & -x_i y_i \hat{y}_i & -y_i^2 \hat{y}_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i \hat{y}_i & -y_i \hat{y}_i \\ x_i^2 \hat{x}_i & x_i y_i \hat{x}_i & x_i \hat{x}_i & x_i^2 \hat{y}_i & x_i y_i \hat{y}_i & x_i \hat{y}_i & -x_i^2 \hat{s}_i & -x_i y_i \hat{s}_i \\ x_i y_i \hat{x}_i & y_i^2 \hat{x}_i & y_i \hat{x}_i & x_i y_i \hat{y}_i & y_i^2 \hat{y}_i & y_i \hat{y}_i & -x_i y_i \hat{s}_i & -y_i^2 \hat{s}_i \end{bmatrix} \tag{4.11}$$

and

$$\mathbf{b}_i = \begin{pmatrix} x_i\hat{x}_i & y_i\hat{x}_i & \hat{x}_i & x_i\hat{y}_i & y_i\hat{y}_i & \hat{y}_i & x_i\hat{s}_i & y_i\hat{s}_i \end{pmatrix}^\top. \tag{4.12}$$

The solution is collected in the parameter vector

$$\mathbf{h} = \begin{pmatrix} h_{00} & h_{01} & h_{02} & h_{10} & h_{11} & h_{12} & h_{20} & h_{21} \end{pmatrix}^\top. \tag{4.13}$$

Figure 4.4 shows an example result of applying the linear least-squares fitting algorithm for the perspective motion model. The *rail* sequence is a pure background sequence without foreground objects, so no correspondence outliers from foreground motion are present. However, there is a

small number of outliers that are errors of the feature-correspondence algorithm. These few outliers have not much influence since the number of correct correspondences is significantly larger.

4.2.5 Non-linear least-squares estimation

We have seen previously that the algebraic error measure can result in an inaccurate estimate if the noise in the data is high. Consequently, we will develop an alternative least-squares estimator in this section which directly uses the Euclidean error metric.

Let $\mathcal{C} = \{\mathbf{x}_i \leftrightarrow \hat{\mathbf{x}}_i\}$ be a set of point-correspondences. We want to find parameters for $\mathbf{H}$ to minimize the error defined as

$$E_2 = \sum_i e_{i;x}^2 + e_{i;y}^2, \tag{4.14}$$

where $e_{i;x}$ and $e_{i;y}$ are the residuals of a single point-correspondence in horizontal and vertical direction:

$$e_{i;x} = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + 1} - \hat{x}_i \quad ; \quad e_{i;y} = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + 1} - \hat{y}_i. \tag{4.15}$$

To find a solution, we use the Levenberg-Marquardt algorithm [151]. This algorithm is a combination of a gradient-descent and Newton-like algorithm. Apart from the error function, the algorithm also requires the partial derivatives with respect to the parameters (for the gradient-descent) and the Hessian matrix (for the Newton optimization-algorithm). Using the abbreviations $D = h_{20}x_i + h_{21}y_i + 1$, $N_x = h_{00}x_i + h_{01}y_i + h_{02}$, and $N_y = h_{10}x_i + h_{11}y_i + h_{12}$, we can determine the derivatives as

$$\begin{aligned}
\frac{\partial e_{i;x}}{\partial h_{00}} &= \frac{\partial e_{i;y}}{\partial h_{10}} = x_i/D \quad ; \quad \frac{\partial e_{i;y}}{\partial h_{00}} = \frac{\partial e_{i;x}}{\partial h_{10}} = 0 \\
\frac{\partial e_{i;x}}{\partial h_{01}} &= \frac{\partial e_{i;y}}{\partial h_{11}} = y_i/D \quad ; \quad \frac{\partial e_{i;y}}{\partial h_{01}} = \frac{\partial e_{i;x}}{\partial h_{11}} = 0 \\
\frac{\partial e_{i;x}}{\partial h_{02}} &= \frac{\partial e_{i;y}}{\partial h_{12}} = 1/D \quad ; \quad \frac{\partial e_{i;y}}{\partial h_{02}} = \frac{\partial e_{i;x}}{\partial h_{12}} = 0 \\
\frac{\partial e_{i;x}}{\partial h_{20}} &= -N_x x_i/D^2 \quad ; \quad \frac{\partial e_{i;y}}{\partial h_{20}} = -N_y x_i/D^2 \\
\frac{\partial e_{i;x}}{\partial h_{21}} &= -N_x y_i/D^2 \quad ; \quad \frac{\partial e_{i;y}}{\partial h_{21}} = -N_y y_i/D^2.
\end{aligned} \tag{4.16}$$

Based on these derivatives, we obtain the gradient vector and Hessian matrix for each iteration step. The optimization can be started with $\mathbf{H}$ equal

	Avg. algebr.	Avg. nonlin.	Avg. RANSAC	Max. algebr.	Max. nonlin.	Max. RANSAC
roma	0.150	0.150	0.455	0.513	0.509	1.014
rail	0.112	0.112	0.569	0.450	0.449	1.659
opera4	0.600	0.600	1.122	3.177	3.180	3.682
nature2	0.176	0.176	0.541	0.795	0.803	1.710

Table 4.1: *Motion model error E_v for different estimation techniques. Shown are the average and maximum values, computed over the complete sequence. Note that the RANSAC column shows the model error for the drawn sample excluding the re-estimation step using all inliers.*

to the identity matrix as the initial starting condition. Note that the minimization of E_2 is considerably more complex than for the linear case, because it is an iterative process and in each iteration, the derivatives of Eq. (4.16) have to be computed and summed over all feature-points.

Comparison to linear least-squares

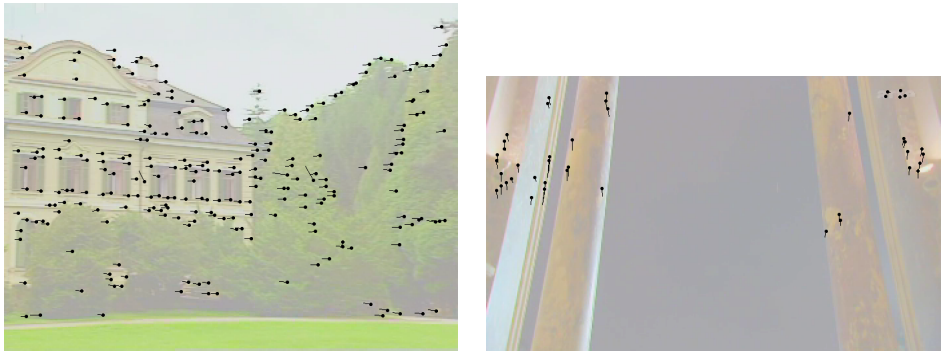
To decide if the simplified linear estimation algorithm using algebraic distances can be used instead of the more complex non-linear algorithm, we compared the difference between the estimated motion model and the reference model $\mathbf{H}^\star$. Since the reference model is unknown, we instead use the result of our complete motion-estimation system including the parameter refinement from Section 5.2. We assume that these parameters are very accurate, since no alignment errors are visible in the reconstructed sprite image, which is based on these motion parameters.

We quantify the distance between two transforms by transforming a point using both transforms, computing the distance between the two resulting positions and averaging over the image area. More specifically, if $\mathcal{A}$ is the image area and $\mathbf{H}$ and $\mathbf{H}^\star$ are the two transforms, we define the transform distance E_v as

$$E_v = \frac{1}{|\mathcal{A}|} \iint_{\mathcal{A}} d(\mathbf{Hp}, \mathbf{H}^\star\mathbf{p}) \, \mathrm{d}x \, \mathrm{d}y. \tag{4.17}$$

We computed the transform distance E_v for the four test sequences and computed the average and the maximum value over all frames. The results are shown in Table 4.1 (the RANSAC column will be discussed later).

It is clearly visible that the results obtained with the algebraic distance do not differ much from the results obtained with the Euclidean distance.



(a) Typical frame of *nature2*. (b) Critical frame in *opera4*.

Figure 4.5: *Detected feature-correspondences. While the sequence (a) has many features that are well distributed over the frame, only a few features could be found in sequence (b). Moreover, the features are not distributed uniformly over the image area. Consequently, the accuracy of the motion estimation is lower for sequence (b). See also Table 4.1.*

Apparently, the noise in the feature location is so small that no difference is observable between both parameter estimation algorithms. We can conclude that the simpler algebraic distance can be used without sacrifying accuracy.[1]

4.3 Robust estimation algorithms

As long as we can assume that the only source of errors are inaccuracies in the feature-point positions, the parameters can be determined using a least-squares approximation as described above. Unfortunately, this is only the case for video sequences showing pure camera motion and no independent object motion. In most practical situations, the data is disturbed by gross outliers or it comprises multiple concurrent motions, so that robust estimation algorithms have to be applied. The purpose of the robust estimation algorithms is to fit a given function to a set of data points, even if

[1] Also visible in the table is the unusually high error for the *opera4* sequence. The reason for this is the fact that the sequence shows very little texture (see Fig. 4.5(b)), so that only few feature-points are generated. Moreover, these features are not distributed equally over the image. A motion model that is derived from only these features will have a larger error at positions that are distant to the detected features. We will evaluate the problem of parameter estimation from a poor set of features in detail in Section 4.3.3. In most cases, these errors in the feature-based motion estimator can be corrected in the direct estimation algorithm that will be described in Chapter 5.

the data is contaminated with a considerable number of outliers.

In this section, we present a robust estimation algorithm that extracts the dominant motion model from the a mixture of different motions. The robust estimation algorithm separates the input data into inliers (part of the dominant motion), and outliers (non-dominant motion or erroneous correspondences). For the estimation of the motion parameters from the inlier data, we apply the parameter-estimation algorithm derived in Section 4.2.

4.3.1 Breakdown of least-squares fit on data with outliers

The direct least-squares approach for parameter estimation works well for a small number of outliers that do not deviate too much from the correct motion. However, the result is significantly distorted when the number of outliers is larger, or the motion is very different from the correct camera motion. Especially if the sequence shows independent object motions, a least-squares fit to the complete data would try to include all visible object motions into a single motion model. Obviously, this cannot give a reasonable result.

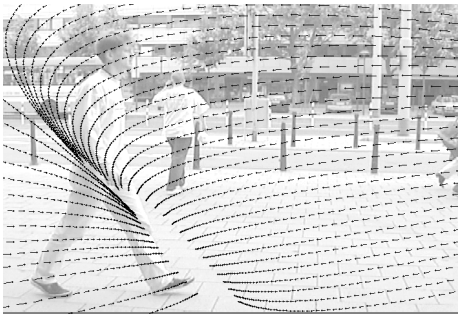
Figure 4.6 shows an example taken from a sequence with panning camera motion (background moves to the left) and object motion (human walks to the right) at the same time. The result of fitting the model to all correspondences is shown in Fig 4.6(b). This non-sense result presents a motion field which indeed moves to the left at the right part of the picture (where mostly camera motion is visible) and in the other direction at the left side (where a large object is visible). However, this motion field is neither a good representation for the camera motion nor for the object motion.

The solution to this problem is to separate feature-correspondences that originate from different motions and to compute independent motion fields for each set of correspondences. However, this is a chicken-and-egg problem. How can we classify the correspondences into different motion types if the motion fields are unknown, and on the other hand, how can we compute the motion-field parameters, if the sets of consistent feature-correspondences are unknown? This problem is addressed in the following sections.

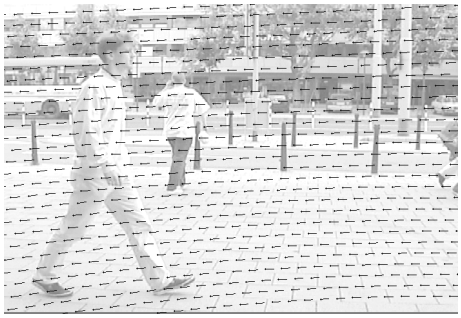
4.3.2 Robust estimation using RANSAC

We consider the following inverse problem. We are given two video frames that contain several areas with different motions. Two motions are considered different if the motions cannot be explained by a single projective motion model. The apparent motion model parameters as well as the segmentation into differently moving image areas are unknown. The only input is a sparse set of samples of the image motion. The objective is to obtain

(a) Detected correspondences (outliers found by RANSAC are drawn in white color).



(b) Motion field computed from all correspondences.



(c) Motion field computed using inlier correspondences only.

Figure 4.6: *Example from the human sequence. The computed correspondences are shown in (a). They are classified as either inliers (black) or outliers (white) by a RANSAC algorithm. (b) shows the result of fitting a projective motion model on the whole data-set using a least-squares estimation with algebraic distance measure. (c) shows the result of using the same estimation technique, but fitting only to the inlier correspondences.*

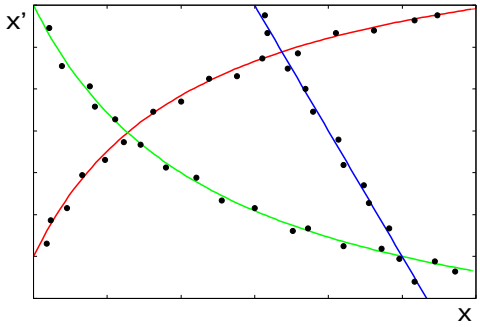


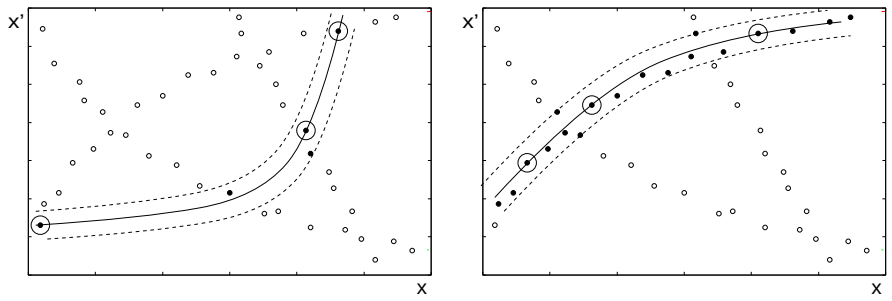
Figure 4.7: *Illustration of multiple motions. Each point represents the motion of one feature-correspondence. Correspondences for different motion models lie on different manifolds.*

the model parameters for the dominant motion, i.e., the motion model that has the largest support of input data. In practice, this dominant motion is usually the camera motion.

The main difference to the last section is that we now have a mixture of several motions with unknown parameters. For the one-dimensional case, this is visualized in Figure 4.7. As mentioned earlier, we cannot start with estimating motion parameters for one of the models, since the partitioning into uniform motion areas is still unknown, and we also cannot start with the partitioning until the motion model parameters are known. This deadlock situation can be solved with robust estimation algorithms, of which RANSAC (RANdom SAmple Consensus) [73] is the most prominent one (other approaches [175, 183, 159] are described in Appendix C). The idea is to repeatedly guess a set of model parameters using small subsets of data that are drawn randomly from the input. The hope is to draw a subset with samples that are part of the same motion model. After each subset draw, the motion parameters for this subset are determined and the amount of input data that is consistent with these parameters is counted. The set of model parameters that has the largest support of input data is considered to be the most dominant motion model visible in the image.

Introductory examples

Let us consider again the previous example of estimating a one-dimensional perspective motion model. Since we have three free parameters, we also need three input correspondences to determine one set of parameters. Consequently, every draw from the input data must contain three samples. From these samples, we can directly calculate the motion parameters. Now,



(a) Non-fitting sample. (b) Good sample.

Figure 4.8: *Two steps of the RANSAC algorithm. A sample set of size three is drawn to compute the parameters of a one-dimensional perspective motion model. All input data that is close to the model computed from the drawn samples is considered as inliers (black dots). Circles mark the outlier data.*

basically two cases are possible. If we are unlucky, the samples will be drawn from different motions (Figure 4.8(a)) and their support of inlier input data (the data which is close to the computed motion model) is small. However, if we draw the samples from a consistent motion (Figure 4.8(b)), the obtained parameter set will have a larger support. To increase the probability of finding a consistent set of samples, we have to repeat the random drawing of subsets several times where the number depends on the fraction of inlier data. Finally, we select the largest set of inliers and assume that it mainly consists of data from only one motion model. Consequently, we can now use a standard least-squares estimation on this inlier data to obtain an accurate parameter set for the motion model.

RANSAC algorithm

Let us now describe the RANSAC algorithm for the special case of estimating the parameters of a two-dimensional perspective motion model. We denote the set of correspondences, which we use as algorithm input, by $\mathcal{C} = \{\mathbf{p}_i \leftrightarrow \hat{\mathbf{p}}_i\}$, and we further denote the Euclidean distance between two points $\mathbf{p}_i$ and $\mathbf{p}_k$ as $d(\mathbf{p}_i, \mathbf{p}_k)$. The RANSAC algorithm can then be described with the following steps.

1. Draw a subset $\mathcal{S}$ of size $|\mathcal{S}| = 4$ from $\mathcal{C}$. Four correspondences are required to solve for the eight free parameters of the motion model.

2. Compute the parameters $\{h_{jk}\}$ of the motion model $\mathbf{H}$ from the correspondences in $\mathcal{S}$ using the linear system in Eq. (3.2).

3. Determine the set of inliers $\mathcal{I} = \{\mathbf{p}_i \leftrightarrow \hat{\mathbf{p}}_i \ \in \mathcal{C} \mid d(\hat{\mathbf{p}}_i, \mathbf{H}\mathbf{p}_i) < \epsilon\}$ which is the set of correspondences that comply with the motion model. In other words, this means that we use the current set of parameters to transform the features from the first image into the second and compare this with the measured positions. If the distance is low, then the pair of points is assumed to comply with the motion model, and it is selected as an inlier.

4. Repeat Steps 1–3 several times (N) and choose the set of inliers for which $|\mathcal{I}|$ is largest.

5. Perform a least-squares approximation of the motion parameters with the set of inliers as described in Equation (4.10). The solution is the result of the RANSAC algorithm.

The RANSAC algorithm has two parameters that have to be chosen initially: the number of draws N and the inlier threshold ϵ. A good value for the inlier threshold can be obtained from the evaluation of the feature-point detector. The more accurate it can locate the features, the smaller ϵ can be chosen. Section 3.2.5 showed that the number of found correspondences by increasing ϵ saturates very quickly. Hence, we have chosen a small value around 1.5 for ϵ, but the right selection of ϵ is not critical. If it is chosen too low, some correct correspondences will be sorted out as outliers, but usually the set of inliers is still large enough to estimate accurate model parameters. If it is chosen too high, some outlier data will be included, but since these outliers cannot differ much from the inliers (their error is below ϵ), their influence in the least-squares approximation will be limited.

The required number of draws N primarily depends on the percentage of outliers p_o we expect in the input, and it also depends on the maximum probability for algorithm failure that is acceptable. This probability P that the RANSAC algorithm will fail computes as

$$P(p_o, |\mathcal{S}|, N) = \underbrace{\Big(1 - \underbrace{\underbrace{\big(\underbrace{1 - p_o}_{\text{percentage of inliers}}\big)^{|\mathcal{S}|}}_{\text{probability to draw set of inliers}}}_{\text{probability to draw set with at least one outlier}}\Big)^{N}}_{\text{probability to get only outlier sets after all draws}}, \tag{4.18}$$

where $|\mathcal{S}|$ is the size of the subset to be drawn (four in our case). By fixing a probability $P(o, |\mathcal{S}|, N)$ of algorithm failure, we can compute the required number of draws N. Clearly, we can always increase N to be more robust, however, this will also increase the required computation time. Let

us assume as an example that we have an outlier percentage of 30%, then only 20 draws would be enough to reduce the probability of algorithm failure to 0.004. Since the inner loop of the RANSAC algorithm is not very computationally expensive, we can even choose a larger number of draws, like 50. Section 3.3.1 discussed that by using motion prediction, the correspondences will *lock* to the camera motion and fewer correspondences will be generated for foreground objects. This favourable effect can reduce the number of outliers beforehand, so that the typical percentage of outliers is even lower than in the example.

4.3.3 Robustness of the RANSAC algorithm

The RANSAC algorithm is a probabilistic technique that is not always successful. However, by increasing the number of draws, the probability of failure can be reduced to arbitrarily small values. Using Eq. (4.18) implies that the number of draws N depends on the fraction of outliers p_o, the sample subset size $|\mathcal{S}|$ and the maximum allowed probability of failure P. For simplicity, we will abbreviate the probability that a non-fitting subset is drawn by $p_f = 1-(1-p_o)^{|\mathcal{S}|}$ during the following discussion. Consequently, to achieve a maximum error rate of not more than P, we need at least $N = \log P / \log p_f$ draws.

Robustness against outliers

To validate the theoretical derivation of the probability of success, we generated synthetic input data, consisting of a fraction p_i of inlier correspondences (motion vectors) that were consistent with a given motion model. Furthermore, this set of data was contaminated with a fraction p_n of random motion vectors, and a fraction p_2 of *object* motion vectors that are consistent with a second motion model. In total, this gives an outlier fraction of $p_o = p_2 + p_n$. We carried out a large number of random draws and compared the obtained motion model with the predefined inlier model, which gave us a measured probability p'_f to draw a non-fitting subset. The obtained p'_f was very close to the theoretical value p_f. The result did not depend on the type of outlier (noise or secondary motion model). RANSAC could also successfully find the correct motion model if $p_m > 50\%$. However, it should be noted that the fraction of secondary motion data must be smaller than the fraction of inliers ($p_2 < p_i$), since otherwise the second motion model is the dominant one.

Difference between theory and practice

In a second set of experiments, we measured the robustness of the RANSAC algorithm for noisy real-world data. We selected sequences for which the correct motion model $\mathbf{H}^\star$ was previously computed using our complete motion-estimation system. We computed feature-correspondences and classified them into inliers $\mathcal{I}^\star$ and outliers based on the precomputed accurate motion model. This gave us the fraction of outliers p_o in our input data. Afterwards, the RANSAC algorithm was executed with a large number of subset draws. For each subset (not only for the best one), the refined motion model was computed as described in Step 5 of the RANSAC algorithm and another set of inliers $\mathcal{I}_R$ was determined based on the refined motion model. If this set of inliers was equal for more than 90% to the set of inliers $\mathcal{I}^\star$ obtained with the accurate motion model, the computed motion model was considered to be correct. Note that a direct comparison between motion models is not possible because of small differences in the parameters. The fraction p'_f of incorrect motion models that we obtained in the simulation should approximately equal the theoretical fraction p_f. However, we noticed that the actual probability to draw a non-fitting subset is much higher (see Table 4.2; compare columns *theory* vs. *refinement steps=1*). As a consequence, an inaccurate motion model is often computed even if all of the four correspondences in our subset are inliers. This effect will now be further analyzed.

Dependency of the failure probability on the sample distances

In order to find the reason for this degraded performance, we marked the randomly drawn subset and the obtained set of inliers in the input image (see Fig. 4.12(b)). It can be seen that the inliers are spatially concentrated with an almost clear border to the area with outliers. Moreover, it can also be verified that the inlier area is larger if the points from the drawn subset are spatially distant (Fig. 4.12(a)).

The reason for this behaviour are numerical instabilities that can be easily visualized in the simpler one-dimensional affine case (Fig. 4.9). In this case, a linear model is computed through two sample points. However, the position of the sample points is distorted by some noise. This uncertainty of the sample positions has a higher influence on the obtained model parameters if the samples have a smaller distance. In our one-dimensional case, this means that the slope of the model line will be inaccurate and only a few points near the two samples will be classified as inliers.

To validate this explanation, we have further analyzed the dependency between the probability of having found a successful set of parameters and

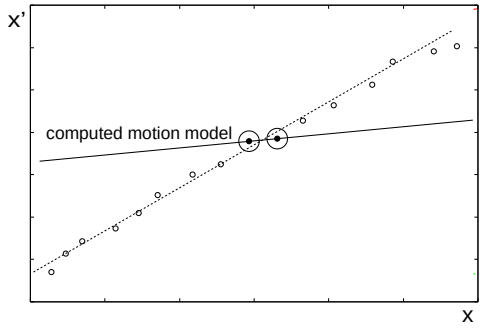


Figure 4.9: *RANSAC for a linear estimation problem. Even though both selected points are inliers, the model defined by these points differs much from the optimum model. Inaccuracies in the point positions have a large influence on the model if the points are close together.*

the distance between the samples from which the parameters were derived. In order to show this dependency, we computed the total sample distance

$$d_s = \frac{1}{2} \sum_{i,k \in \{1,2,3,4\}} d(\mathbf{p}_{s_i}, \mathbf{p}_{s_k}) \tag{4.19}$$

for each selected subset $\{s_1, s_2, s_3, s_4\}$ and plotted the measured probability of failure p'_f depending on d_s. It can be observed (Fig. 4.10) that the probability of failure indeed decreases with larger sample distances. On the other hand, the estimation will almost certainly fail if the distances are very small.

Improving RANSAC by equalizing the sample distribution

One possible solution (even though described for the related problem of computing a *fundamental matrix*) has been proposed in [201]. The idea is to disable the selection of samples which are too close by dividing the image into a grid of rectangular buckets, similar to the technique described in Section 3.3.1. Random samples are now obtained in two steps. First, a bucket is randomly selected, followed by a random selection of a feature-point within this bucket. Since the number of points in the buckets are unequal, the selection is weighted by the number of points. To get spatially distant samples, a bucket may only be chosen once in each iteration.

We do not follow this technique, since it favours the selection of distant feature-points, but it cannot prevent that the position of the sample set is degenerated. For example, all feature-points could lie in one line.

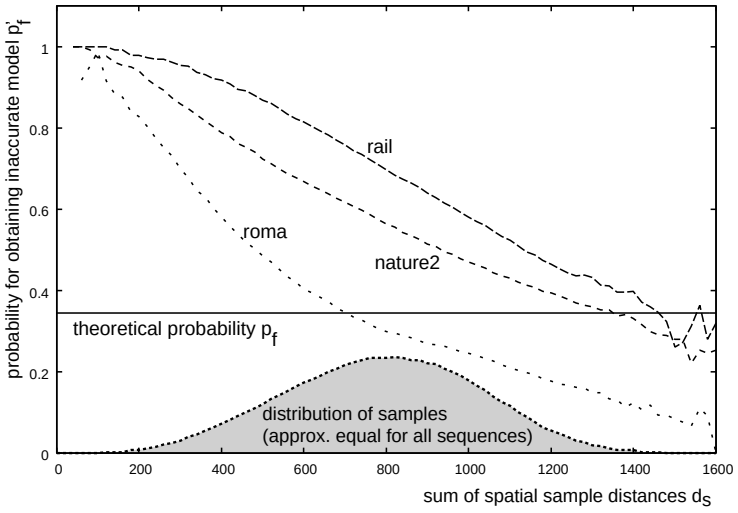


Figure 4.10: *Probability of generating an inaccurate motion model depending on the distance between the samples in the drawn subset. The* opera4 *sequence is not included since the number of features is very low and unequally distributed.*

Improving RANSAC using iterative model refinement

As an alternative solution, we propose to keep the original random sample selection strategy, but to carry out the motion-parameter refinement (Step 5) of the RANSAC algorithm several times. The idea is that the initially obtained motion model is not always accurate, but it still includes a considerable number of inliers. Each time the model parameters are adapted to the newly obtained set of inliers, the number of inliers will increase.

A sample result is shown in Figure 4.12(b)-(d), where the set of inliers after each of the refinement steps is marked. It is clearly visible that the area of inliers grows with each refinement step. The measured probabilities of failure p'_f for the improved algorithm are shown in Figure 4.11 and Table 4.2. It is interesting to note that for a larger number of refinement steps (≥ 3), the measured probabilities of failure are even below the theoretical value. The reason for this is that some of the outlier correspondences are very close to being classified as inliers. Consequently, even if one of these *almost-inliers* is selected, the motion model still converges to the correct model.

Because each additional refinement step might improve the final motion model, the probability of failure p'_f decreases which also means that

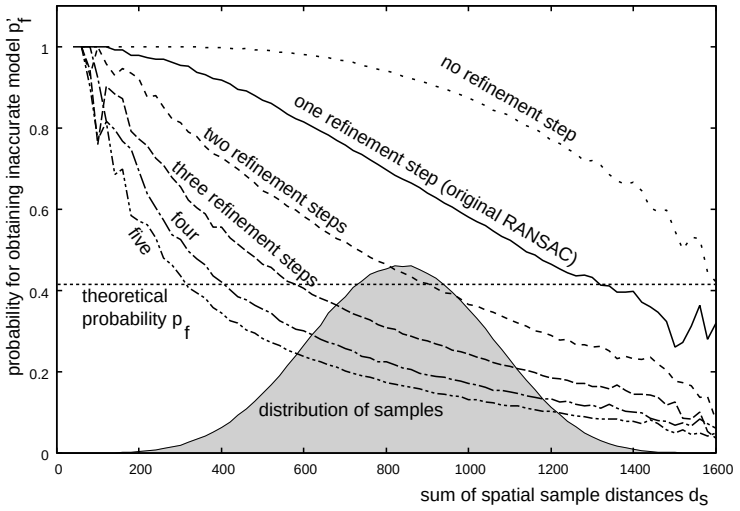
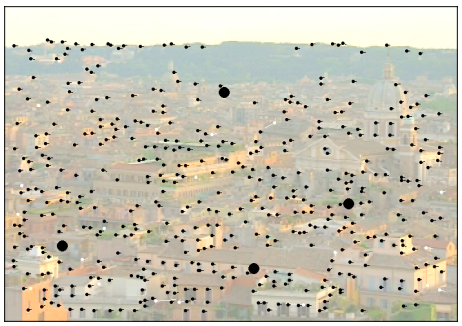


Figure 4.11: *Probability of generating an inaccurate motion model depending on the distance between the four samples in the drawn subset. The values are based on the rail sequence. The probability is drawn for different numbers of model refinement steps (the original RANSAC uses a single step). Also shown is the distribution of the sample distances as they were drawn randomly from the image.*

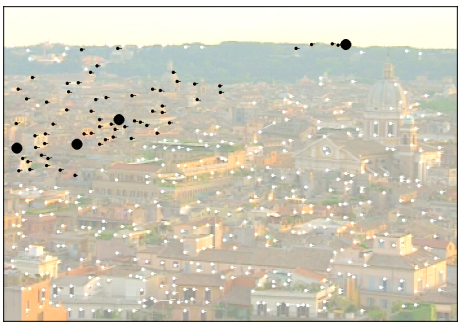
the number of required subset draws N can be reduced. On the other hand, each refinement step requires some additional computation time. Hence, the question arises what the optimum number of refinement steps is. Since the most computational intensive step in the RANSAC algorithm is the separation of the samples into inliers and outliers, we count the total required computation time in units of these classification steps to be performed. If we denote the number of refinement steps as R, we get the total computation time C as

$$C = (R + 1) \cdot \lceil \log P / \log p'_f \rceil. \tag{4.20}$$

After conducting experiments on several test-sequences, we could see that three refinement steps resulted in the lowest computation time. Since the probability of failure p'_f for three refinement steps is usually close to or smaller than the theoretically determined value p_f, we can use the theoretically computed number of iterations.

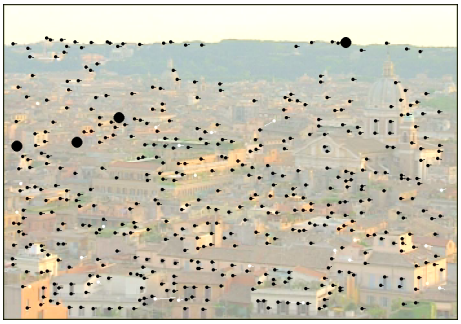


(a) Good sample set (original RANSAC).



(b) Degenerate sample set (after 1^{st} refinement = original RANSAC).

(c) Degenerate sample set (after 2^{nd} refinement).



(d) Degenerate sample set (after 3^{rd} refinement).

Figure 4.12: *Examples for obtained sets of inliers (black color). (a) A good sample set provides an accurate motion model. (b)-(d) A non-fitting sample gives inaccurate motion parameters, but they can be improved by additional refinement steps.*

4.4 Summary

This chapter described the second half of the feature-based camera-motion estimation. Whereas the previous chapter presented the computation of feature-point correspondences, the current chapter explored the estimation of motion parameters from feature-point correspondences.

First, we considered the parameter estimation for scenes in which only camera motion is present. We found that a simple linear algorithm can be used for affine motion models, but that non-linear optimization is required for the projective motion model. However, a comparison between the non-linear parameter estimation and a linear approximation showed that the accuracy of the linear-approximation algorithm is comparable.

Afterwards, we extended the algorithm to differentiate between fore-

Failure risk $P = 0.1\%$		Theory	Refinement steps 1	2	3	4	5
rail	p'_f	**42.7%**	**66.9%**	46.5%	36.9%	22.7%	17.9%
$p_o = 13\%$	N	8	18	9	7	5	4
roma	p'_f	**14.6%**	**31.6%**	13.7%	7.9%	6.1%	4.7%
$p_o = 4\%$	N	4	6	4	3	3	3
opera4	p'_f	**54.1%**	**62.3%**	48.8%	42.0%	38.5%	36.3%
$p_o = 18\%$	N	12	15	10	8	8	7
nature2	p'_f	**31.7%**	**55.3%**	32.8%	21.8%	17.0%	14.8%
$p_o = 9\%$	N	6	12	7	5	4	4

Table 4.2: *Probability of degenerated subset draws for different number of refinement steps. The original RANSAC algorithm corresponds to* refinement steps=1. *Also shown is the number of draws N that are needed to reach an algorithm failure rate below* 0.1%.

ground motion and background motion, such that the camera-motion parameters can also be estimated even when the camera motion is mixed with object motion. We applied the RANSAC algorithm[2] to detect the dominant motion and to compute its model parameters. The RANSAC algorithm is a probabilistic algorithm that succeeds only with a certain probability, which can be increased arbitrarily by carrying out more program iterations. However, experiments showed that the probability of failure was larger than predicted by a theoretical analysis. It was found that the reason for the reduced performance are degenerate sets of samples, which lead to numerical instabilities in the parameter estimation. We addressed this problem by re-estimating the parameters based on the obtained inlier and then recomputing the set of inliers for a small number of iterations. This increases the set of inliers in each iteration such that the parameter estimation is based on more input data, resulting in a more accurate estimation. With this modification to the RANSAC algorithm, we could increase the probability of success to reach or even exceed the theoretically predicted performance.

Resulting algorithm flow-graph

The data-flow of the motion-parameter estimation is depicted in Figure 4.13. It shows the RANSAC algorithm with an additional loop for the refinement steps. The algorithm input is formed by the feature-point correspondences

[2]See Appendix C for a description of alternative algorithms.

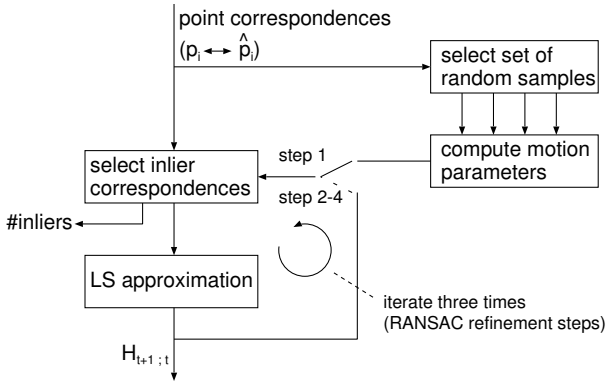


Figure 4.13: *Estimation of motion parameters based on a RANSAC algorithm. Depicted is only one iteration. The algorithm is repeated several times and the solution with the largest number of inliers is selected.*

that are extracted in the previous step (see Chapter 3). Four random samples are selected and a candidate motion model is computed from these samples. All input correspondences are compared with this motion model to separate them into an inlier set and the outliers. After this, refined motion parameters are computed with a least-squares approximation on all inliers. The last two steps, selection of inliers, and least-squares approximation, is repeated three times to converge to a maximum coverage of feature-points. This whole process is repeated several times, and only the motion model that had the largest number of inliers is returned as result.

Experimental results

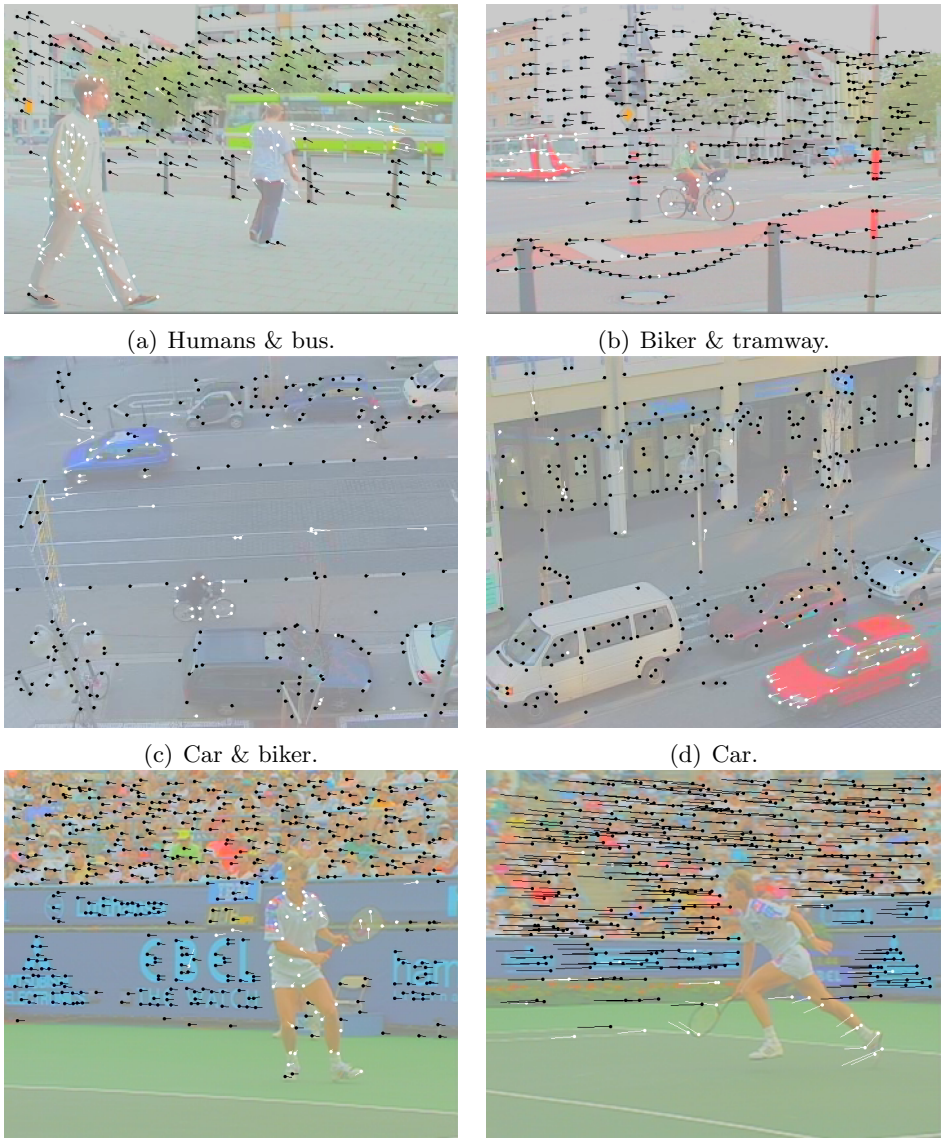
The camera-parameter estimator has been tested on many sequences that were either recorded from regular DVB broadcasts, or recorded with a camcorder. Additionally we also used some standard test sequences. The algorithm proved to be very robust on most sequences. Problems only arose if the video contained too few features in the background, or if it had a very low contrast such that the feature-point extraction could not find good features to track. In most cases, errors in the feature-based motion estimator could be corrected in the direct motion estimator that will be explained in the next chapter.

Some example results, in which both background and foreground motion are present, are depicted in Figure 4.14. In Figures 4.14(a) to (d), surveillance-type scenes were recorded with a hand-held camera. In a real application, this could be a remotely controlled pan-tilt-zoom camera in a

surveillance system. Finally, Figures 4.14(e) and (f) show two scenes of the *stefan* test sequence, one with a slow camera motion and one with a very fast camera pan.

For the experiments, we selected the Harris algorithm to detect features, the search-range of the feature-matching algorithm was set to 16 pixels around the predicted feature position, and the RANSAC algorithm used 25 iterations with 3 refinement steps. All algorithm parameters were fixed for all sequences. The pictures show the inlier (background motion) vectors in black color and the outliers (foreground motion and erroneous vectors) in white color.

The examples show that foreground motion and background motion are well separated. In addition to the foreground motion, a small number of outliers can be observed that result from bad feature-correspondences. An interesting effect is visible in Fig. 4.14(f): the foreground object contains almost no features. The reason is that the feature-correspondence algorithm only searches for matching correspondences in a small neighborhood around the predicted feature position. Since the feature positions are predicted with the camera motion parameters, the predicted position is far away from the object motion. Consequently, the algorithm does not find the correspondences for the object motion. For our application, this is an advantage because the number of outliers in the input for the RANSAC algorithm is decreased.



(a) Humans & bus.

(b) Biker & tramway.

(c) Car & biker.

(d) Car.

(e) Slow camera motion.

(f) Fast camera motion.

Figure 4.14: *Inliers (black) and outliers (white) as detected by the RANSAC algorithm for different scenes with foreground objects. See Section 4.4 for more details about (f).*