Evaluation of a Feature-Based Global-Motion Estimation System

Dirk Farin[a*] and Peter H.N. de With[a,b]

[a]Eindhoven University of Technology, PO Box 513, 5600 MB, The Netherlands
[b]LogicaCMG, PO Box 7089, 5600 JB Eindhoven, The Netherlands

ABSTRACT

Global-motion estimators are an important part of current video-coding systems like MPEG-4, content analysis and description systems like MPEG-7, and many video-object segmentation algorithms. Feature-based motion estimators use the motion vectors obtained for a set of selected points to calculate the parameters of the global-motion model. This involves the detection of feature points, the computation of correspondences between two sets of features, and the motion parameter estimation. In this paper, we will present a feature-based global-motion estimation system and discuss each of its parts in detail. The idea is to provide an overview of a general purpose feature-based motion estimator and to point out the important design aspects. We evaluate the performance of different feature detection algorithms, propose an efficient feature-correspondence algorithm, and we compare the difference between a non-linear parameter estimation and a linear approximation. Finally, the RANSAC based robust parameter estimation is examined, we show why it does not reach its theoretical performance, and propose a modification to increase its accuracy. Our global-motion estimator has an average accuracy of ≈ 0.15 pixels with real-time execution.

Keywords: Global-motion estimation, feature-point detection, robust estimation, feature correspondences.

1. INTRODUCTION

Video sequences usually comprise a mixture of foreground object motion and camera motion. While general non-rigid object motion is complex to describe, camera motion can be modeled with a small set of parameters since the variety of possible motions is limited. An important class of camera motion is the motion of a rotational camera, since it enables techniques like the generation of a panoramic scene backgrounds, as they are used in the MPEG-4 sprite coding tools.[1] Furthermore, these camera-motion parameters are used in video analysis systems as a feature to describe video contents. For example, these global-motion descriptors have been defined in MPEG-7. Many video-object segmentation algorithms also rely on camera-motion compensated input, because they assume a virtually static camera.[2]

The motion field of a rotational camera can be modeled with a projective transform. Often, this model is written as the coordinate transformation

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

with the eight parameters h_{ij}. A more convenient formulation is obtained by writing the point positions with homogeneous coordinates $\mathbf{p} = (x, y, 1)^\top$ and $\mathbf{p}' = (x'', y'', w'')^\top$. This allows to write the projective transformation as a matrix multiplication $\mathbf{p}' = \mathbf{H}\mathbf{p}$ with the parameter matrix $\mathbf{H} = (h_{ij})_{ij}$. The homogeneous formulation is equivalent to Eq. (1) if we set $x' = x''/w'', y' = y''/w''$ and normalize $\mathbf{H}$ such that $h_{22} = 1$.

One class of algorithms that has been applied to global-motion estimation is the class of feature-based motion estimators. These estimators comprise three main algorithms. The first identifies a set of feature points in each of the input images. Feature points are placed at locations with unique image content that can be easily found back in the successive frame. The second algorithm establishes correspondences between the feature points detected in a pair of images. Each correspondence can be considered as the motion vector of this specific feature point. Because the feature points were selected to show unique image content, these correspondences have a high

[*] email: `d.s.farin@tue.nl`

probability to show the correct motion. The third algorithm estimates the motion-model parameters from this set of feature point correspondences. The main problem in this third algorithm is to separate vectors belonging to foreground motion from background motion vectors. This problem is approached with a robust estimation algorithm that identifies the largest subset of motion vectors that can be described with a single set of camera-motion parameters. Under the assumption that camera motion is the dominant motion in the image, the robust estimation will extract the correct camera-motion parameters.

In this paper, we consider aspects of each part of the feature-based motion estimator in detail. In particular, we discuss the following topics.

- **Feature point detection** (Section 2). The important aspect for global-motion estimators is that the detector extracts a similar set of corners in each of the images (feature *repeatability*). Furthermore, the feature positions should not deviate from their corresponding positions even if the considered images have been geometrically transformed (feature localization *accuracy*), We evaluate a variety of feature point detectors, including the Harris corner detector, the Shi-Tomasi, the SUSAN, and the Moravec detector. For all of these detectors, we measured the repeatability and accuracy by comparing the detection result with the ground-truth motion.
- **Computing feature correspondences** (Section 3). The design of the correspondence algorithm has to consider two aspects. First, as the number of detected features can be in the order of several hundreds to thousands, an efficient implementation should be found. Second, the calculated correspondences should include as few wrong matches as possible, because these outliers decrease the robustness of the successive parameter estimation. We propose a fast algorithm that employs motion prediction to reduce the area to search for correspondences, and also to reduce the number of false matches that do not comply with the camera motion.
- **Robust parameter estimation** (Section 4) For the parameter estimation, we first consider the separation of the correspondences into camera motion and object motion. This separation is based on the RANSAC (RANdom SAmple Consensus) algorithm, which is a robust estimation algorithm that estimates the parameters of the dominant subset of the input data. In our case, we assume that the dominant motion is the camera motion. RANSAC is a probabilistic algorithm that only succeeds with a certain probability. Our evaluation shows that in practice, the probability of success is lower than the theoretically predicted performance. We point out the reason for this degraded performance and we provide a modification to the RANSAC algorithm to increase its probability of success to, or even exceeding, the theoretical performance.

  In Section 5, we evaluate the accuracy of the least-squares parameter estimation. We compare the accuracy of a non-linear estimation which minimizes the Euclidean reprojection error, with a linear approximation that minimizes an algebraic cost.

The value of the proposal that we are going to present in the upcoming sections is that we present a complete system in which in every stage the algorithms are optimized to decrease the computational complexity and provide a higher robustness. For robustness reasons, we evaluated each algorithm with a variety of test sequences[†] covering a wide range of possible applications.

2. FEATURE POINT DETECTION

In the past, a large number of interest point detectors have been proposed and this topic is still an active area of research. One of the first interest point detectors was the Moravec detector.[3] It detects image locations that show luminance variations in many different directions. Shi and Tomasi[4] describe an interest point detector, that is derived directly by identifying the sort of texture for which the motion-estimation process is well defined.

[†]Throughout the paper, we will use the four test sequences (*roma*, *rail*, *nature*, and *opera*) for the quantitative results. All of these sequences show pure camera motion. The *roma* sequence is a slow horizontal camera pan over a very textured scene (Fig. 4) while *rail* and *nature* show complicated camera motion (fast rotation, zooming). The *opera* sequence (Fig. 6(a)) is a fast camera pan in a scene with only little texture. Additionally, we use the *stefan* sequence for the experiments that require a combination of foreground motion and background motion.

Many authors propose to use interest point detectors that are in fact corner point detectors. Two commonly used corner detectors are the *Harris* (also known as *Plessey*) corner detector[5] and the SUSAN corner detector.[6]

Recent work concentrates on designing interest point detectors that are invariant to some classes of image transformations. Since image transformations are usually small between successive frames of a video sequence, ordinary interest point detectors work well for our application and we will not consider these more complex invariant detectors here. For further information see, e.g.,.[7, 8]

We implemented the Shi-Tomasi, Harris, Moravec, and SUSAN feature point detectors. The SUSAN detector uses the original source code that was provided by the authors. The Harris detector was implemented in two variants. A version using full-pixel accuracy and, a second version, using a modification that locates the corner with sub-pixel accuracy by fitting a second-order polynomial to the decision function and computing the minimum of this polynomial.

2.1. Ground-truth motion and reference correspondences

For the successive evaluations, we often need to know the ground-truth motion parameters $\mathbf{H}^\star$. Since these true motion parameters are difficult to acquire, we combined the described feature-based global-motion estimator with a successive gradient-based motion estimation. This second estimator computes a long-term alignment of all input frames to a single, stitched panoramic image.[2] Because visual inspection of these alignments showed no alignment errors, we take the parameters from this alignment as our ground-truth $\mathbf{H}^\star$.

From this ground-truth motion, we can derive ground-truth feature correspondences for a pair of images. To distinguish these correspondences from the correspondences that will be computed by the successive feature-matching step, we denote them as *reference correspondences*. These reference correspondences are defined using the ground-truth motion $\mathbf{H}^\star$ between the two images. Let us denote the set of feature points in the first image t as $\mathcal{F}_t = \{\mathbf{p}_i\}$ and as $\mathcal{F}_{t+1} = \{\hat{\mathbf{p}}_k\}$ for the second image $t+1$. Two corresponding points $\mathbf{p}_i$, $\hat{\mathbf{p}}_k$ are written as a point-correspondence $\mathbf{p}_i \leftrightarrow \hat{\mathbf{p}}_k$. We define the set of reference correspondences basically as $\mathcal{R}_\epsilon = \{\mathbf{p}_i \leftrightarrow \hat{\mathbf{p}}_k \mid d(\hat{\mathbf{p}}_k, \mathbf{H}^\star \cdot \mathbf{p}_i) < \epsilon\}$, but with the additional constraint that each feature point is only included once. Note that the parameter ϵ defines the maximum displacement error that is allowed for a correspondence. A good choice for ϵ will become apparent after we evaluated the repeatability of the feature point detectors.

2.2. Criteria for good detectors

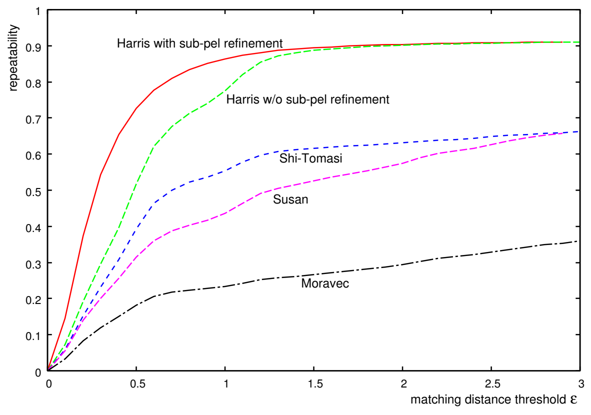
Feature points should be placed at positions where the image content around the feature allows to reliably identify the corresponding position in a second image. We evaluate the performance of the feature point detectors according to the following two criteria.

- **Repeatability.** Features that are only found in one of the images cannot be part of a point correspondence and consequently, they are useless for the succeeding motion analysis. Even worse, unmatched features will contaminate the data with noise and may lead to wrong correspondences, which will then again make the robust estimation harder. Hence, we desire to extract the same set of feature points from the input images. To quantify this property, we define the *repeatability* of a feature detector as the fraction of detected feature points in one frame that can also be found in the other frame:

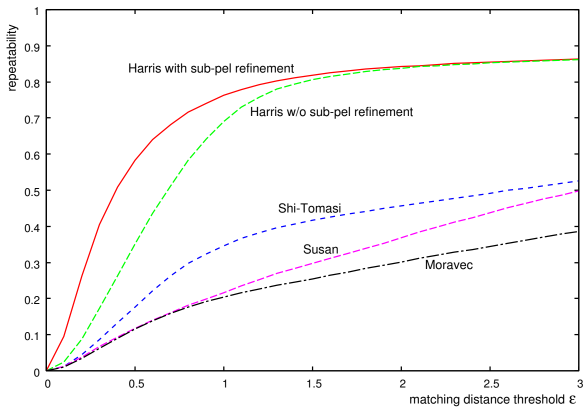
$$\textit{repeatability} = \frac{|\mathcal{R}_\epsilon|}{\min(|\mathcal{F}_t|, |\mathcal{F}_{t+1}|)}. \tag{2}$$

  This is the number of matched feature points, normalized by the minimum number of feature points in the two input frames. Since we cannot get more correspondences than the minimum number of feature points, the maximum value for *repeatability* is 1.0.

- **Accuracy.** A good feature point detector provides unbiased positions that are invariant to the image transform that is present between the two images. Clearly, a more consistent placement of feature points leads to a higher accuracy of the motion-estimation parameters since the displacement errors are smaller. We define the accuracy of a feature point detector as the mean distance of the feature point position $\hat{\mathbf{p}}_\mathbf{k}$ in the second frame and the position of the corresponding feature point in the first frame, mapped onto



(a) *roma*



(b) *nature*

Figure 1. Repeatability of the detected feature points: fraction of matched feature points.

the second frame by $\mathbf{H}^\star$. Feature points which have no counterpart in the other image are not considered. This gives the definition of the accuracy as

$$accuracy = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{p_i} \leftrightarrow \hat{\mathbf{p}}_\mathbf{k} \in \mathcal{R}} d(\hat{\mathbf{p}}_k \ , \ \mathbf{H}^\star \cdot \mathbf{p}_i). \tag{3}$$

2.3. Evaluation and Discussion

The diagrams in Figure 1 show the *repeatability* for two test sequences, using ϵ as a parameter. In each diagram, the repeatability is averaged over the complete sequence length. From these two diagrams and the results of other sequences, we see that the Harris detector reaches a repeatability between $80 - 90\%$, where saturation is almost reached for $\epsilon \approx 1.5$. Note that a repeatability near 100% cannot be reached since new image content appears and old content disappears during the sequences. Especially the feature points near the image border are often impossible to match since their counterparts are outside of the visible image area.

The repeatability of all other detectors are clearly below the result of the Harris detector and they also do not reach a clear saturation, which is due to a lower localization accuracy. The Shi-Tomasi and SUSAN detectors reach both a repeatability rate between $40 - 70\%$, while the Moravec operator only reaches about $30 - 40\%$.

The results for the *accuracy* of the four sequences are summarized in Table 1. We see that the Harris detector again shows the best performance and that its accuracy can indeed be increased by our sub-pel refinement.

Summarizing, we can conclude that according to our experiments, the Harris detector shows the best performance for both repeatability and accuracy. It can also be seen that the sub-pel refinement can in fact increase the accuracy of the Harris detector by about 0.2 pixels. While the accuracy is sufficient for all the detectors, the repeatability is not satisfactory except for the Harris detector. As a consequence, we selected the Harris detector for our motion estimation system and we will not consider the other detectors in the successive sections.

3. COMPUTING FEATURE CORRESPONDENCES

After feature points have been extracted for each video frame, we have to establish correspondences between the feature points. Computing the correspondences is subject to a number of problems. As we have seen in the last section, no feature point detector is able to provide an ideal repeatability. This means that our data sets will always include feature points that have no matching counterpart in the other image. However, this is not only due to imperfections of the detection algorithms, but it is also caused by the video content itself. One reason are foreground objects that occlude areas that have still been visible in the last frame, or objects that uncover previously hidden areas.

Several algorithms for computing feature point correspondences have been proposed in the literature.[9, 10] Some of them are especially designed to accomodate for large motions.[11] However, for our application, the motion is relatively small and well predictable from previous frames. More important in practice is the fact that the number of feature points is usually very high (about 1000-2000 for CIF resolution video), which raises the need for a computationally efficient algorithm.

3.1. Fast greedy feature matching

In the following, we first describe the algorithm core, which we then modify to decrease the computational complexity and also to decrease the number of outlier matches.

Let us denote the Euclidean distance between the points $\mathbf{p}_i$ in the first image and points $\hat{\mathbf{p}}_k$ in the second image as $d(\mathbf{p}_i, \hat{\mathbf{p}}_k)$. Under the assumption that the motion between successive frames is small, we only consider feature correspondences between points that are within a search-range d_{max}. The purpose of this threshold is to reduce the computational complexity and also to block correspondences between points that are obviously too far away. During a camera pan motion, for example, feature points disappear at one side of the image and new points appear on the other side. Without limiting the feature point distance, matches between these points on opposite sides of the image would be possible. We start the algorithm by establishing the set of all admissible candidate correspondences

$$\mathcal{C}_0 = \left\{ \mathbf{p}_i \leftrightarrow \hat{\mathbf{p}}_k \;\middle|\; d(\mathbf{p}_i, \hat{\mathbf{p}}_k) \le d_{max} \right\}. \tag{4}$$

For each of the correspondences $\mathbf{p}_i \leftrightarrow \hat{\mathbf{p}}_k$ in $\mathcal{C}_0$, we compute the sum-of-absolute-differences (SAD) of the image luminance in a small window around the feature point. In a greedy approach, we iteratively select the candidate correspondence $c = \mathbf{p}_i \leftrightarrow \hat{\mathbf{p}}_k$ with the lowest SAD and add that correspondence to the final set of correspondences: $\mathcal{C} := \mathcal{C} \cup \{c\}$. Since each feature point may only participate in one correspondence, we remove all candidate correspondences from $\mathcal{C}_0$ that have a feature point in common with the selected correspondence:

$$\mathcal{C}_0 := \left\{ \mathbf{p}_c \leftrightarrow \hat{\mathbf{p}}_d \, \in \mathcal{C}_0 \;\middle|\; \mathbf{p}_c \ne \mathbf{p}_i \wedge \hat{\mathbf{p}}_d \ne \hat{\mathbf{p}}_k \right\}. \tag{5}$$

The algorithm ends when $\mathcal{C}_0$ is empty or the dissimilarity errors exceed a threshold m_{max}. This threshold m_{max} is required to prevent the algorithm from associating completely dissimilar feature points that may remain after most correct correspondences have been found.

3.2. Reducing the search range with motion prediction

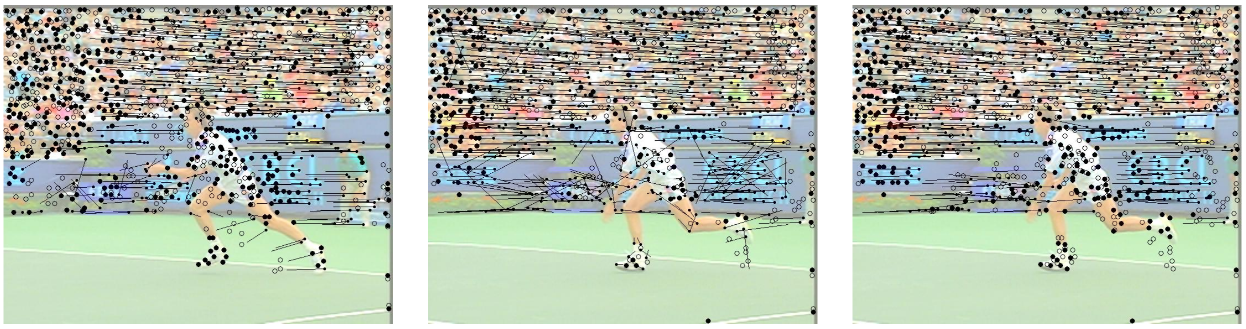
The computational complexity of the above correspondence algorithm is primarily influenced by the size of the initial candidate set $\mathcal{C}_0$. One approach to reduce the candidate set could be to reduce the search-range d_{max}, but this would make the above algorithm unusable for sequences with fast motion. However, we can assume that the motion between successive frames is smooth without abrupt changes. This allows us to use the motion model from the previous pair of frames as a predictor to estimate the position of the feature points in the current frame. Hence, we can limit the search range for matching features to only a small neighborhood around this estimated feature position. Using the motion model that we computed for the previous pair of images, we can replace Eq. (4) by

$$\mathcal{C}_0 = \left\{ \mathbf{p}_i \leftrightarrow \hat{\mathbf{p}}_k \;\middle|\; d(\mathbf{H}_0\mathbf{p}_i, \hat{\mathbf{p}}_k) \le d'_{max} \right\}, \tag{6}$$

where $\mathbf{H}_0$ is the motion model prediction and d'_{max} is the radius of the search range around the predicted position. Because the predicted feature position will be closer to the actual position, we can choose d'_{max} smaller than d_{max}, which reduces the computation complexity without sacrifying the ability to handle fast motions. Note that using the prediction, only the maximum change of speed is limited, not the maximum motion speed.

Table 1. Accuracy results for the four test sequences *roma*, *rail*, *opera*, and *nature*.

accuracy (pixels)	roma	rail	opera	nature	average
Harris (sub-pel)	0.34	0.56	0.51	0.46	0.47
Harris (integer)	0.53	0.74	0.65	0.67	0.65
Shi-Tomasi	0.53	0.80	0.95	0.83	0.78
SUSAN	0.75	1.13	0.98	1.11	0.99
Moravec	0.74	1.05	0.76	0.98	0.88



(a) No motion prediction, search range is 40 pixels.
(b) No motion prediction, search-range is 70 pixels.
(c) Enabled motion prediction, search-range is 8 pixels.

Figure 2. Comparison of the correspondence algorithm without and with motion prediction for a scene with fast motion.

3.3. Evaluation

The results of some interesting situations are visualized in Figure 2. The correspondences are drawn with lines, while unmatched features points are drawn with large circles. Let us first consider the algorithm without motion prediction. As soon as the speed of camera motion exceeds the search range, features remain unmatched (left side of Fig. 2(a)). Increasing the search-range has three effects. The algorithm can find correspondences for faster camera motion, but it requires more computation time. Moreover, a large search range also increases the probability to find non-correct matches (Fig. 2(b)).

However, if we use motion prediction to relocate the center of the search range, we can obtain the correct correspondences with a much smaller search range (Fig. 2(c)). At the same time, the number of wrong matches is reduced, since the algorithm locks to the camera motion $\mathbf{H}_0$ and since it only detects correspondences that fit into the global-motion model.

The diagrams in Figure 3 depict the fraction of correctly found correspondences, depending on the search-range. From these experiments, we see that a small search range of about 2 pixels is enough when motion prediction is used, while a much larger search range is required without prediction. Also note that the number of wrong correspondences found ($|\mathcal{C}|/|\mathcal{R}| - |\mathcal{C} \cap \mathcal{R}|/|\mathcal{R}|$) mainly depends on the search-range. Consequently, we have fewer wrong correspondences when we use prediction, since the algorithm can operate with a smaller search-range.

4. ROBUST GLOBAL-MOTION ESTIMATION

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. However, 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.

4.1. Robust estimation using RANSAC

The goal is to obtain 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 most popular approach for this robust estimation problem is the RANSAC (RANdom SAmple Consensus)[12] algorithm, even though several variants have been developed.[13–15] The idea is to repeatedly guess a set of model parameters using small subsets of correspondences that are drawn randomly from the input set $\mathcal{C}$. With a large number of draws, there is a high probability to draw a subset of correspondences that are part of the same motion model. After each subset draw, the motion parameters for this subset are determined and the number of correspondences in $\mathcal{C}$ that are consistent with these parameters is counted. The set of model parameters with the largest support is considered to be the most dominant motion model visible in the image. The RANSAC algorithm can be described with the following steps.

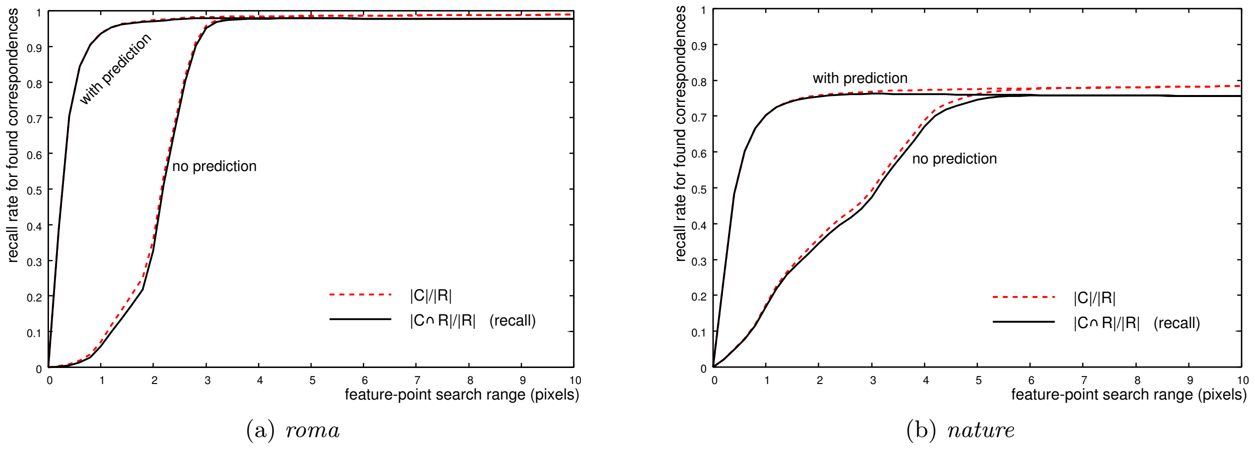


Figure 3. Recall rate of correct correspondences for different search ranges d_{max} (solid line). The total number of returned correspondences (also normalized to $|\mathcal{R}|$) is shown with a dashed line. The fraction of wrong correspondences are represented by the difference between the two values.

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 $\mathcal{S}$.
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.
4. Repeat Steps 1–3 several times (N) and choose the set of inliers for which $|\mathcal{I}|$ is largest.
5. Do a least-squares approximation of the motion parameters with the set of inliers (see Section 5). The solution is the result of the RANSAC algorithm.

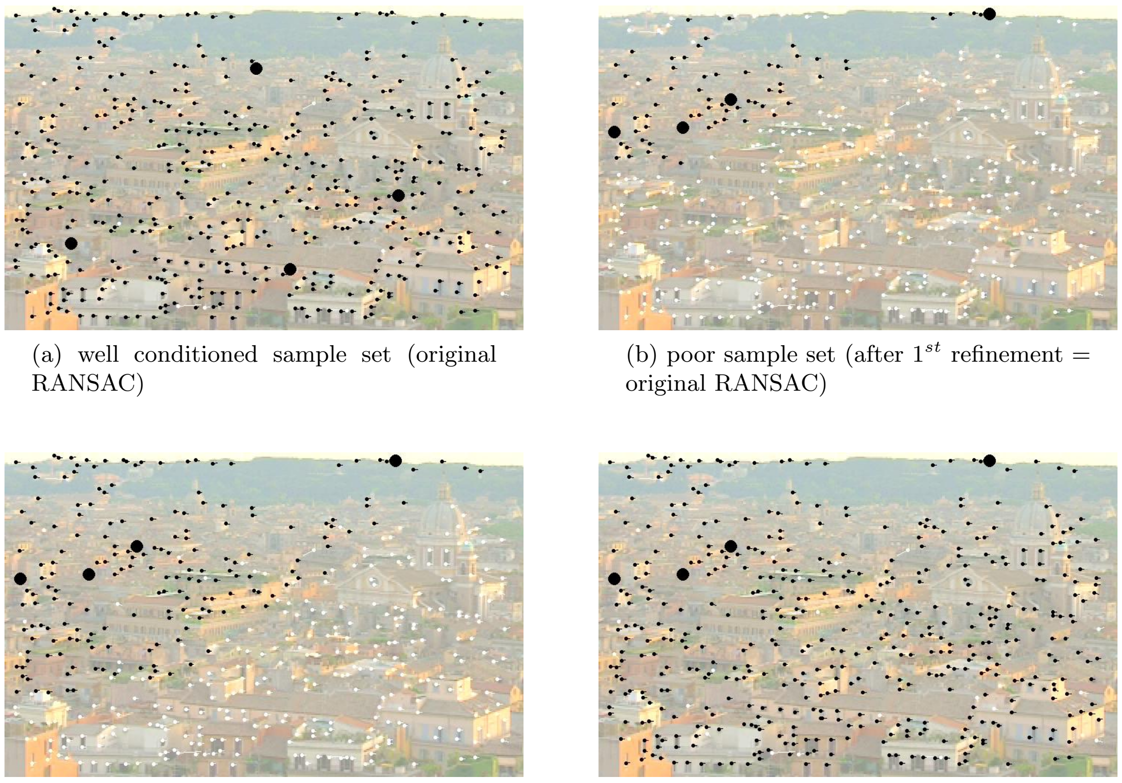
The RANSAC algorithm has two parameters that have to be chosen: the number of draws N and the inlier threshold ϵ. A good value for the inlier threshold can be obtained from the evaluation of the feature point detector. Hence, we have chosen a small value around 1.5 for ϵ, but the right selection of ϵ 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 determine the 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 small.

The required number of draws N is primarily dependent on the percentage of outliers p_o in the input, and it also depends on the maximum probability for algorithm failure that we are going to accept. This probability P that the RANSAC algorithm will fail computes as $P(p_o, N) = p_f^N$, where $p_f = 1 - (1 - p_o)^4$ is the probability that a poor subset is drawn. By fixing a probability $P(p_o, N)$ of algorithm failure, we can compute the required number of draws N. To limit the maximum error rate to P, we need at least $N = \log P / \log p_f$ draws.

4.2. Evaluation of the robustness of the RANSAC algorithm

To measure the robustness of the RANSAC algorithm, we assume again that the correct motion model $\mathbf{H}^\star$ is known. Using this ground-truth motion model, we can classify the computed feature correspondences into inliers $\mathcal{I}^\star$ and outliers. This also gives us the fraction of outliers p_o in our input data.

To evaluate the probability of success for the RANSAC algorithm, many iterations N were computed. In each iteration (instead of only for the best draw), a refined motion-model was computed as described in Step 5 of the RANSAC algorithm. Afterwards, this model is used to determine a set of inliers $\mathcal{I}_k$. If this set of inliers $\mathcal{I}_k$


(a) well conditioned sample set (original RANSAC)

(b) poor sample set (after 1st refinement = original RANSAC)

(c) poor sample set (after 2nd refinement)

(d) poor sample set (after 3rd refinement)

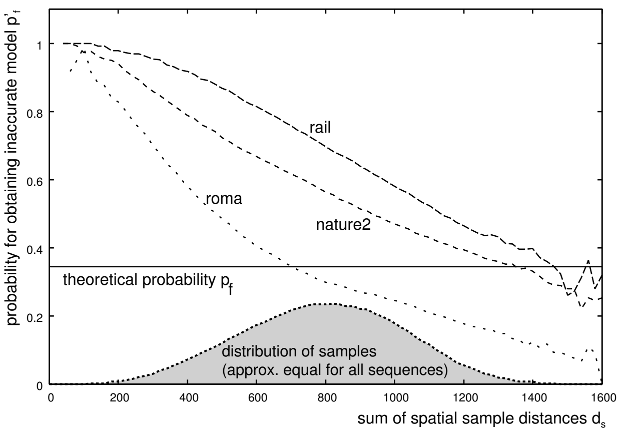
Figure 4. Examples for obtained sets of inliers (black color). (a) A sample set with widely spaced feature points provides an accurate motion model. (b)-(d) A poor sample gives inaccurate motion parameters, but they can be improved by additional refinement steps.

is equal for more than 90% compared to the set of inliers $\mathcal{I}^\star$ obtained with the accurate motion model, then we consider the computed motion-model to be correct. The fraction of incorrect models obtained in the simulation is denoted p'_f. Theoretically, p'_f should equal the theoretical fraction p_f. However, we noticed that the actual probability to draw a poor subset is much higher (see Table 2; compare the column *theory* for p_f, with the column *refinement steps=1* for p'_f). As a consequence, this means that in many cases, an inaccurate motion model is computed even if all of the four correspondences in our subset are inliers. This effect will be further analyzed in the following.

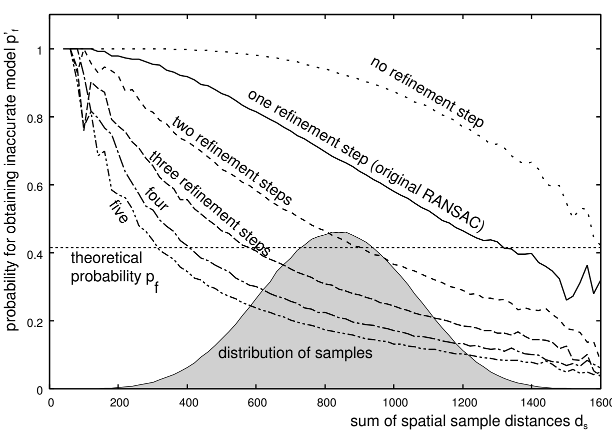
4.3. Dependency of the failure probability on the sample distances

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(b)). It can be seen that the inliers are spatially concentrated with an almost clear border to the area with outliers. Moreover, it can be seen that the inlier area is larger if the points from the drawn subset are spatially distant (Fig. 4(a)).

The reason for this behaviour are numerical instabilities originating from the noise in the input data. If the samples are all close together along one direction, small errors in their position are extrapolated more than if there errors are spread over a larger distance. To validate this explanation, we have further analyzed the dependency between the probability of having found a successful set of parameters and the distance between the samples from which the parameters were derived. In order to show this dependency, we computed the total



(a) Original RANSAC.



(b) The *rail* sequence with a varying number of refinement steps.

Figure 5. Probability of generating an inaccurate motion model depending on the distance between the four samples in the drawn subset.

sample distance

$$d_s = \frac{1}{2} \sum_{i,k \in \{1,2,3,4\}} d(\mathbf{p}_{s_i}, \mathbf{p}_{s_k}) \tag{7}$$

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. 5) that the probability of failure indeed decreases with larger sample distances.

4.3.1. Improving RANSAC using iterative model refinement

A set of inliers that was computed with a poorly conditioned set of samples will often not cover the complete true set of inliers, but it will approximately cover the area around the drawn samples. The refined motion parameters that are computed from this set of inliers have a higher accuracy, since they were obtained with a least squares approximation over a larger set of input data. Using these refined motion parameters to find again a new set of inliers will result in a better coverage. If we repeat these refinements, the set of inliers will ultimately converge to a full coverage.

A sample result is shown in Figure 4(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 5 and Table 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 sometimes the refinement iterations converge to the correct motion model even if the initial sample included outlier correspondences.

With each additional refinement step, the probability of failure p'_f decreases. Hence, the question about the best balance between the number of draws N and the number of refinement steps R arises. Since the most computationally intensive step is the computation of a least-squares fit of the motion parameters, we take them as the unit of computation time. Hence, we get the total computation time C as $C = (R+1) \cdot \lceil \log P / \log p'_f \rceil$. After conducting experiments on several test-sequences, we could see that three refinement steps resulted in the lowest computation time. For three refinement steps, the probability of failure p'_f is also close to the theoretically determined value p_f, such that we can use the theoretically computed number of iterations.

5. MOTION PARAMETER ESTIMATION

In the previous section, we already employed an algorithm to compute the global-motion parameters from a set of point correspondences, but we did not yet present this algorithm. In this section, we close this gap by comparing two approaches and by evaluating the accuracy of both.

Table 2. Probability of poor 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 to push the algorithm failure rate below 0.1%.

failure risk		original RANSAC		number of refinement steps			
$P = 0.1\%$		theory	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
opera	p'_f	**54.1%**	**62.3%**	48.8%	42.0%	38.5%	36.3%
$p_o = 18\%$	N	12	15	10	8	8	7
nature	p'_f	**31.7%**	**55.3%**	32.8%	21.8%	17.0%	14.8%
$p_o = 9\%$	N	6	12	7	5	4	4

Let $\mathcal{I} = \{\mathbf{x}_i \leftrightarrow \hat{\mathbf{x}}_i\}$ be the set of background motion vectors from which we want to compute the motion parameters $\{h_{ij}\}$. These parameters can be estimated by minimizing the Euclidean reprojection error, which is defined as $E_2 = \sum_i (x'_i - \hat{x}_i)^2 + (y'_i - \hat{y}_i)^2$. However, this leads to a non-linear optimization, which we solved using the Levenberg-Marquardt algorithm.

To avoid the requirement of a non-linear optimization, we can replace the Euclidean error E_2 with an algebraic error that leads to a linear least squares problem. Multiplying the transform equation with its denominator results in the algebraic error

$$\begin{aligned} E_a &= \sum_i \underbrace{\left((x'_i - \hat{x}_i)^2 + (y'_i - \hat{y}_i)^2\right)}_{\text{Euclidean distance}} (h_{20}x + h_{21}y + 1)^2 \\ &= (h_{00}x + h_{01}y + h_{02} - \hat{x}_i(h_{20}x + h_{21}y + 1))^2 + (h_{10}x + h_{11}y + h_{12} - \hat{y}_i(h_{20}x + h_{21}y + 1))^2. \end{aligned} \tag{8}$$

After imposing the necessary condition $\partial E_a / \partial h_{ik} = 0$ for an error minimum, this results in a linear equation system from which we can also obtain $\{h_{ij}\}$.

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 accuracy between the estimated motion model and the reference model $\mathbf{H}^\star$. We quantify the accuracy difference between the estimated $\mathbf{H}$ and the ground-truth $\mathbf{H}^\star$ 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, we define the transform distance E_v as

$$E_v = \frac{1}{|\mathcal{A}|} \iint_{\mathcal{A}} d(\mathbf{Hp}, \mathbf{H}^\star\mathbf{p})\, \mathrm{d}x\, \mathrm{d}y. \tag{9}$$

We computed the transform accuracy E_v for the four test sequences and computed the average and the maximum value over all frames. The results are shown in Table 3. For comparison, we also included a column

Table 3. Localization error E_v for different estimation techniques. The table shows the average and maximum value, computed over the complete sequence. Note that the RANSAC column shows the model error for the drawn sample excluding any least-squares estimates over the inliers.

	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
opera	0.600	0.600	1.122	3.177	3.180	3.682
nature	0.176	0.176	0.541	0.795	0.803	1.710

showing the accuracy of the motion model that is obtained by the RANSAC algorithm if no refinement step is carried out (i.e., the motion parameters are computed directly from the best set of samples).

It is clearly visible that the results obtained with the algebraic distance do not differ much from the results obtained with the Euclidean distance. Apparently, the noise in the feature location is so small that no difference is observable between both parameter estimation algorithms. For us, this has the nice conclusion that we can use the simpler algebraic distance without sacrifying accuracy.

Also visible in the table is the unusually high error for the *opera* sequence. The reason for this is the fact that the sequence shows very little texture, (see Fig. 6(a)) so that only few feature points are generated. Moreover, these features are not distributed equally over the image. As discussed in Section 4.3, a motion model that is derived from only these features will have a larger error at positions that are distant to the detected features.

6. OVERALL RESULTS

Additionally to the presented sequences, the proposed global-motion estimator has been tested on many sequences that were either recorded from regular DVB broadcasts, sequences recorded with a camcorder, and also some standard test sequences. The algorithm proved to be very robust with most sequences. Problems only arise if the video contains too few features in the background, or if it has a very low contrast such that the feature point extraction cannot find good features to track.

Figure 6 depicts some example results for sequences in which background motion as well as foreground motion is present. In Figures 6(b) to (d), surveillance-type scenes were recorded with a hand-held camera. Finally, Figures 6(e) and (f) show two scenes of the *stefan* test sequence, one with slow camera motion and one with a very fast camera pan.

All sequences were non-interlaced and had approximately CIF resolution. 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. Additionally to the foreground motion, a small number of outliers can be observed that result from incorrect feature correspondences. An interesting effect is visible in Fig. 6(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 it decreases the number of outliers in the input for the RANSAC algorithm.

7. CONCLUSION

A feature-based global-motion estimation system is described and evaluated. Design alternatives for each of its three main parts are presented and compared. In our evaluation of feature-point detectors, the Harris detector showed the best localization accuracy of about 0.5 pixels and also the highest feature repeatability of 85-90%. Furthermore, a prediction technique for the feature-correspondence algorithm is proposed that reduces the required search-range approximately by a factor of 6, and at the same time improves the robustness of the algorithm since it reduces the number of incorrect feature matches. Finally, a discrepancy between the theoretical and the practical performance of the RANSAC algorithm is revealed and a modification to prevent this effect is proposed. This modification reduces the probability to compute an inaccurate motion model from a selection of correspondences by a factor of three. In total, a complete global-motion estimation system for rotational camera motion is described that is robust to foreground object motion and that can be integrated into applications like video coding, video object segmentation, or video content analysis. The proposed global-motion estimation system has a high accuracy of about 0.15 pixels and at CIF resolution, it operates in real-time on a standard Pentium-4 based PC at 2 GHz.

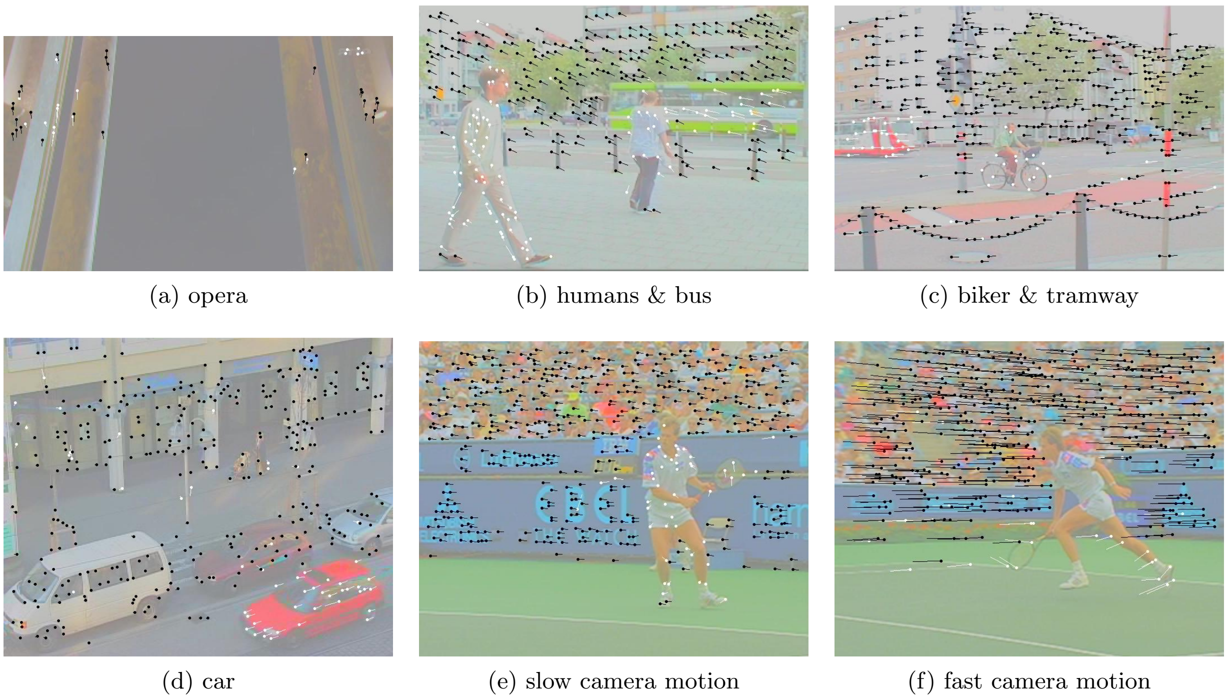

(a) opera (b) humans & bus (c) biker & tramway (d) car (e) slow camera motion (f) fast camera motion

Figure 6. Inliers (black) and outliers (white) as detected by the RANSAC algorithm for a collection of scenes.

REFERENCES

1. K. Jinzenji, H. Watanabe, S. Okada, and N. Kobayashi, "MPEG-4 very low bit-rate video compression using sprite coding," in *Proc. IEEE International Conference on Multimedia and Expo*, pp. 4–7, Aug. 2001.
2. D. Farin, P. H. N. de With, and W. Effelsberg, "Video-object segmentation using multi-sprite background subtraction," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, **1**, pp. 343–346, June 2004.
3. H. Moravec, "Visual mapping by a robot rover," in *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pp. 599–601, August 1979.
4. J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593–600, 1994.
5. C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of The Fourth Alvey Vision Conference, Manchester*, pp. 147–151, 1988.
6. S. M. Smith and J. M. Brady, "SUSAN — a new approach to low level image processing," *International Journal of Computer Vision (IJCV)* **23**, pp. 45–78, May 1997.
7. K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *European Conference on Computer Vision (ECCV)*, pp. 128–142, Springer, 2002. Copenhagen.
8. R. Laganière and E. Vincent, "Wedge-based corner model for widely separated views matching.," in *IEEE International Conference on Pattern Recognition*, **3**, pp. 672–675, 2002.
9. J. Maciel and J. P. Costeira, "A global solution to sparse correspondence problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**, pp. 187–199, Feb. 2003.
10. M. Pilu, "A direct method for stereo correspondence based on singular value decomposition," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 261–266, June 1997.
11. P. H. S. Torr and C. Davidson, "IMPSAC: synthesis of importance sampling and random sample consensus," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**, pp. 354–364, Mar. 2003.
12. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM* **24**(6), pp. 381–395, 1981.
13. C. V. Stewart, "MINPRAN: a new robust estimator for computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**, pp. 925–938, Oct. 1995.
14. P. H. S. Torr and A. Zisserman, "MLESAC: a new robust estimator with application to estimating image geometry," *Compututer Vision and Image Understanding* **78**(1), pp. 138–156, 2000.
15. P. J. Rousseeuw and K. Van Driessen, "Computing LTS regression for large data sets," *Institute of Mathematical Statistics Bulletin* **27**(6), 1998.