# Low-Complexity Camera Ego-Motion Estimation Algorithm for Real Time Applications

Faisal Shafait, Marco Grimm, Rolf-Rainer Grigat
*Vision Systems Department,*
*Hamburg University of Science and Technology, Germany.*
{*faisal.shafait, grimm, grigat*}*@tu-harburg.de*

## Abstract

*This contribution presents a low-complexity camera ego-motion estimation algorithm for real-time applications. The algorithm uses a feature based approach for motion estimation. A new method is introduced for feature selection which limits the number of feature points to be tracked and has a low dependency on structure in the image. Both these factors are important in real time applications, as lesser features to track result in lower computational complexity and lesser dependency on image structure results in smaller variations in computational time for different images. This gain in speed is achieved at the cost of a slightly reduced robustness and accuracy. This trade-off between speed and accuracy pays off particularly in static scenes where high reduction in computational cost is achieved without the accuracy penalty. This algorithm can be used in applications where an estimate of camera motion is required and low computational complexity is of primary concern.*

## 1. Introduction

Motion in an image can be divided into global and local. Motion induced in the image due to camera movement is called global motion, whereas small moving objects in the scene result in local motion in the image. If the moving object is large enough to occupy the complete image, it will produce the same effect as camera movement for ambient illumination, resulting in global motion in the image. The goal of a motion estimation technique is to assign a motion vector (displacement) to each pixel in an image. The choice of a motion estimation approach strongly depends on the target application. A key issue when designing a motion estimation technique is its degree of efficiency with enough accuracy to serve the purpose of intended application. Difficulties in motion estimation arise from occlusion, noise, lack of image texture, and illumination changes.

Existing motion estimation techniques can be categorized into feature-based [1, 2, 3] and region-based [4, 5, 6] approaches. Broszio *et al.* [2] have presented a three-step feature-based approach for camera parameters estimation. In the first step corners are detected from each image. Then correspondences of features between an image and the next in the sequence are established. The final parameter estimation employs the random sampling and consensus procedure RANSAC [7]. Acceleration sensors in combination with image analysis have proved to be useful for camera ego-motion estimation [1]. Richter [3] proposed to divide the image into several tiles to get features from each image area, resulting in better global motion estimation. Region-based methods divide the image into regions and then estimate the motion of each region. The emerging H.264/AVC standard [4] proposes multi-resolution region-based motion estimation with sub-pixel accuracy.

Region-based global motion estimation algorithms are, in general, very accurate, but have high computational complexity. Feature-based methods are more suitable for use in real-time applications. We have used a variation of the three step approach by Broszio [2], so that we get only the necessary information for 2D camera parameter estimation. Then we use some approximations to reduce the computational complexity. Our approach is discussed in detail in section 2. Section 3 gives the computational complexity analysis along with the quantitative evaluation of the algorithm accuracy. Section 4 concludes the paper.

## 2. Camera ego-motion estimation

The motion observed in an image sequence can be caused by camera motion (ego-motion) and/or by motion of objects moving in the scene. In this paper the case of a camera moving in a static scene is addressed, but the range of robustness against object motion is also shown.

### 2.1. Motion Models

An image acquisition system projects the 3D world onto a 2D image plane with image coordinates $\vec{p} = (x,y)^T \in \Lambda$, where $\Lambda$ is an orthogonal sampling grid. Upon this projection, a moving camera results in 2D *motion trajectories* of image points. The 2D displacement can be expressed as

0-7803-8680-9/04/ $20.00 ©2004 IEEE

follows:

$$\vec{d}_{t,\tau}(\vec{p}) = \vec{p}(t+\tau) - \vec{p}(t) \quad (1)$$

where $\vec{p}(t)$ and $\vec{p}(t+\tau)$ represent the position of an image point at a time difference of $\tau$.

Different models exist in literature [8] to model the displacement $\vec{d}(\vec{p})$. Parametric models are most commonly used as they are easy to implement. We have used a translational parametric model for 2D motion. We make the assumption that the image intensity remains constant along the motion trajectory. This assumption implies that any intensity change is due to motion, that scene illumination is constant, and that object surfaces are opaque. Although these constraints are almost never satisfied exactly, the constant-intensity assumption approximately describes the dominant properties of natural image sequences, and motion estimation methods based on it work well.

## 2.2. Method

Camera ego-motion can be estimated from the optical or normal flow derived between two frames [6], or from the correspondence of distinguished features (points, lines, contours) extracted from successive frames [2]. We have used a feature-based approach because of its low computational cost. The block diagram of the algorithm is as shown in Fig. 1.

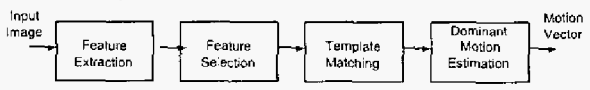


**Figure 1: Block diagram of proposed algorithm for camera ego-motion estimation.**

**2.2.1. Feature Extraction.** In the first stage corners are detected from the camera image based on SUSAN corner detector [9]. For each image of a sequence the list of corner coordinates $L_n = \left\{\vec{p}_{n,1}, \ldots, \vec{p}_{n,i}, \ldots, \vec{p}_{n,N}\right\}$ is extracted, where $\vec{p}_{n,i}$ are the image coordinates of a corner $i$ in image $n$, and $N$ is the total number of corners in the image $n$.

**2.2.2. Feature Selection.** For an image of a structured scene, the number of detected corners is quite high. So it is not possible to track all of them in real-time. Therefore $M$ out of $N$ corners are selected for further processing. In order to ensure getting features from every image area, the image can be divided into several tiles [3]. We perform corner selection based on nearest neighborhood criterion to a reference grid. The reference grid we have used is as shown in Fig. 2.

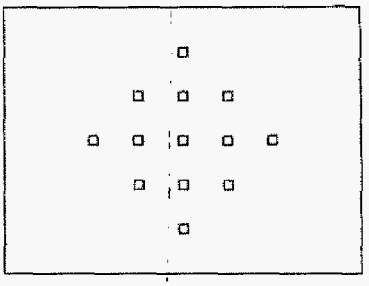

**Figure 2: Reference points used for corner selection. The center point of each $5 \times 5$ rectangle is used as a reference point.**

The corner selection is done such that for each reference point the nearest corner point is selected. Therefore for $j$th reference point $\vec{q}_j$, a corner point minimizing the distance $\left|\vec{p}_i - \vec{q}_j\right|$ is selected. The corner is marked so that it is not selected more than once. Hence we obtain a selected corner list $C_n = \left\{\vec{r}_{n,1}, \ldots, \vec{r}_{n,j}, \ldots, \vec{r}_{n,M}\right\}$, where $\vec{r}_{n,j}$ are the image coordinates of the corner nearest to $j$th reference point in image $n$, and $M$ is the total number of selected corners.

**2.2.3. Template Matching.** The selected corners $C_n$ are tracked in the next image, by a block-matching of $5 \times 5$ block in a $15 \times 15$ search window, instead of establishing correspondences with corner list $L_{n+1}$ of the next image. The reason is that due to camera noise and slight illumination changes, the SUSAN corner detector will not extract the same corners from a static scene. This is illustrated in Fig. 3 for the case of a still camera pointing toward a static scene.

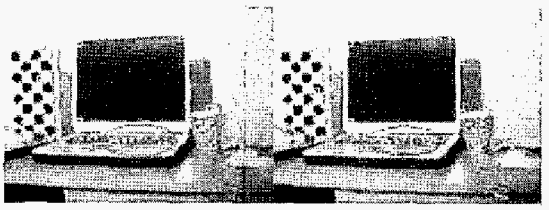

**Figure 3: Images to demonstrate instability of corners detected by SUSAN corner detector.**

The computation time required for performing a full search block-matching does not depend on texture in the image. This results in a uniform computation time for each image, which is a desired feature for real time applications. The other advantage of full search block-matching is that it ensures finding a global minimum within the search range as compared to a 3-step search method which can get trapped in a local minima. Let $(x, y)$ be the position of the center point of the $j$th reference template in the image $n$ at

time $t$. We want to find the best match for this template in image $n+1$ at time $t+\tau$. Mean absolute difference (MAD) as defined in Eq. 2 is used as the error criterion for selecting the best match

$$\mathrm{MAD}(u,v) = \frac{1}{B}\sum_{k,l\in B}|g_t(x+k,y+l) - g_{t+\tau}(x+u+k,y+v+l)| \quad (2)$$

where $g_t(\vec{p})$ is the image intensity at spatio-temporal position $(\vec{p},t)$, $B$ is the template size, and $(u,v)$ define the search region around the template center point. The location $(x+u,y+v)$ resulting in minimum value of $\mathrm{MAD}(u,v)$ (Eq. 2) is defined as the best match for the reference template. So a motion vector $\vec{m}_{e,i} = (u,v)\big|_{\mathrm{MAD}(u,v)=\min}$ is assigned to the template.

**2.2.4. Dominant Motion Estimation.** The template matching module gives us one motion vector for each selected corner. The next task is to estimate the 2D camera motion, i.e. translation in x and y directions, from these motion vectors. Choon *et al.* [6] have proposed median filtering for dominant motion estimation. This requires at least more than half of the vectors to have similar value for correct estimation.

We use the concept of simple majority for dominant motion estimation. This introduces more flexibility by having freedom to choose threshold and tolerance for selection among candidate vectors. The concept of tolerance implies that while evaluating a particular candidate, exact matches as well as matches within a user-defined tolerance are counted. Thresholding means that the number of matches for a selected candidate (the candidate with highest number of matches) must be higher than the user-defined threshold. Otherwise the resultant motion vector is set to zero.

## 2.3. Error Metrics

We have used two different error metrics to evaluate our algorithm. In order to examine the performance of our technique on real image sequences for which ground truth 2D motion fields are not known, we minimize the *displaced frame difference* (DFD) error. Let $g_t(\vec{p})$ be the image intensity at spatio-temporal position $(\vec{p},t)$. Then DFD error is defined as

$$J(d) = \frac{1}{R}\sum_{\vec{p}\in A}\left|g_t(\vec{p}) - g_{t+\tau}(\vec{p}-d_{t,\tau}(\vec{p}))\right| \quad (3)$$

where $g_{t+\tau}(\vec{p}-d_{t,\tau}(\vec{p}))$ is called a motion-compensated prediction of $g_t(\vec{p})$, and $R$ is the number of pixels in the image. DFD-error is also the criterion of choice for evaluating motion estimation algorithms in practical video coders today.

In order to examine the performance of our technique on real sequences for which ground truth 2D motion fields are known, we use vector differences as an error measure. Let $\vec{m}_c$ be the correct displacement, and $\vec{m}_e$ be the estimated displacement. Then the motion estimation error $E_m$ is

$$E_m = |\vec{m}_c - \vec{m}_e| \quad (4)$$

# 3. Results

## 3.1. Computational Complexity Analysis

The proposed algorithm is an extension of the approach by Broszio [2] and has a significantly lower computational cost as shown in table 1. The computational complexity $c_r$ for the reference algorithm can be written as

$$c_r = Rk_H + 2N^2B + (N-1)^2 \quad (5)$$

where $R$ is the number of pixels in the whole image, $k_H$ is the number of operations per pixel required to find Harris corners, $N$ is the total number of corners in the image, and $B$ is the template size. Similarly, the computational complexity $c_p$ for the proposed algorithm can be written as

$$c_p = Rk_S + 2NM + 2SMB + S \quad (6)$$

where $k_S$ is the number of operations per pixel required to find SUSAN corners, $M$ is the number of selected corners, and $S$ is the search range.

**Table 2: Parameters for the computational complexity analysis as described in table I.**

| Parameter | Explanantion |
|---|---|
| $B$ | Block size in pixels ($B = 25$ for a $5 \times 5$ block) |
| $N$ | Total number of corners in the image |
| $M$ | Number of selected corners ($M \leq N$) |
| $S$ | Search window size in pixels |
| $R$ | Image size in pixels |
| $k_H$ | Operations per pixel required for Harris corner detector |
| $k_S$ | Operations per pixel required for SUSAN corner detector |

For an image resolution $R = 320 \times 240$, block size $B = 5 \times 5$, search window size $S = 15 \times 15$, and the number

**Table 1: Computational complexity for each stage of the reference algorithm [2] and the proposed algorithm. Parameters are described in table II.**

| Stage | Reference Algorithm [2] | | Proposed Algorithm | |
|---|---|---|---|---|
| | Technique used | Computational Complexity | Technique used | Computational Complexity |
| Feature Extraction | Harris corner detector | 95 Add and 22 Mult. ops/pixel | SUSAN corner detector | 32.25 Add and 0.75 Mult. ops/pixel |
| Feature Selection | Not used | | Nearest Neigh-bourhood | $NM$ square root and comparisons (worst case) |
| Template Matching | Corres-pondence Analysis | $2N^2B$ Add/Sub and $(N-1)^2$ comparisons | Full-Search Block Matching | $2SMB$ Add/Sub and $S$ comparisons |
| Dominant Motion Estimation | RANSAC | Not fixed | Simple Majority | Not fixed |

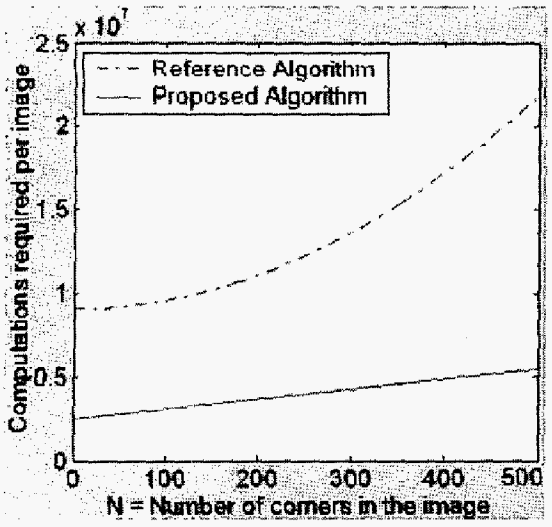


**Figure 4: Comparison of computations required per image for the reference and proposed algorithm.**

of selected corners $M = 13$, Eqs. 5 and 6 are plotted in Fig. 4. The computational complexity is plotted in terms of operations per image, as the exact time required for each operation is hardware dependent.

Figure 4 shows that the computational effort required for the camera ego-motion estimation is significantly reduced in the proposed algorithm as compared to the reference algorithm. The dependence of the computational cost on number of corners in the image is also reduced from $O(N^2)$ to $O(N)$.

### 3.2. Performance Evaluation

The performance of the proposed algorithm was evaluated with four image sequences. Image sequences 1 and 2 contain a camera movement, where the precise 2D motion fields were not known. For those image sequences *displaced frame difference* (Eq. 3) was used as the error metric. The image sequences 3 and 4 contain an object motion. The purpose of evaluation on these image sequences is to examine the effect of object motion on the camera ego-motion estimation algorithm. Ideally, the algorithm should not report any camera motion in this case, and Eq. 4 is used as error metric. These image sequences were obtained from [10].

**Translating Camera:** In this image sequence, the cam-

era moves orthogonally to its line of sight. The scene is static with no object motion. The length of the image sequence is 100 frames, and 25th frame in the sequence is shown in Fig. 5. This sequence was obtained from [11].

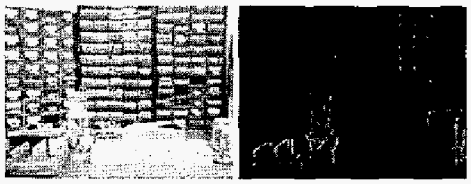

**Figure 5: Frames 25 (left) of the image sequence "Translating Camera", with the motion vectors extracted, and the *displaced frame difference* image(right).**

**Television:** In this image sequence, the camera moves orthogonally to its line of sight, in front of a television set. The purpose of this test sequence was to investigate the effect of brightness changes due to the presence of a television screen in the image, which violates the constant-intensity assumption. This sequence will evaluate the performance of the algorithm in typical indoor environments with low brightness level and poor texture. The length of the image sequence is 15 frames, and the 5th frame in the sequence is shown in Fig. 6.

**Figure 6: Frames 25 (left) of the image sequence "Television" with the motion vectors extracted, and the *displaced frame difference* image(right).**

**Walk Straight:** In this image sequence, a person walks straight from right to left in front of a stationary camera. There is no other object motion except the shadow of the walking person on a glass window in the background. This image sequence addresses the case of a poorly structured object moving in front of a well structured background. The complete image sequence consists of 125 frames, and frames 28 and 85 of the sequence are shown in Fig. 7.

**7 UP:** In this image sequence, a can is moved from left to right while rotating it clockwise in front of a stationary camera. The object covers a bigger area in the image ($\approx$ 20%) and is well structured. The complete image sequence consists of 400 frames, and frames 25 and 175 of the sequence are shown in Fig. 8.

**Figure 7: Frames 28 (left) and 85 (right) of the image sequence "Walk Straight".**

**Figure 8: Frames 25 (left) and 175 (right) of the image sequence "7 UP".**

### 3.3. Error Measurement

The error of the camera ego-motion estimation for the sequences 1 and 2 is calculated using the displaced frame difference (Eq. 3). For the sequences 3 and 4, the vector difference (Eq. 4) was calculated. Table 3 summarizes all results of error measurement. The average DFD-error for the image sequence "Television" was higher than the "Translating Camera" image sequence due to the presence of television set in the image. The DFD-images for each sequence are shown in figures 5 and 6 respectively. The algorithm performed quite well in suppressing the object motion in the image sequence "Walk Straight", detecting no global motion for all images in the sequence. This resulted in both mean error and standard deviation of 0.0 pixels. However for bigger object motion as in the case of image sequence "7 UP", the algorithm was fooled by object motion especially when the moving object was near the image center. This was because of the feature selection procedure in which corners near image center were selected with higher priority. Hence the measured error was higher than that for the "Walk Straight" image sequence.

## 4. Conclusion

In this paper we have presented a camera ego-motion estimation algorithm for real time applications. We have used a new feature selection method which lowers the computational complexity. Another advantage of using the feature selection method is that the dependence of computational time on structure in the image is reduced from $O(N^2)$ to $O(N)$. For image sequences with unknown 2D motion fields, displaced frame difference (DFD) was used

**Table 3: Error measurement results for the tested image sequences.**

| Image Sequence | Error Metric | Average Error | Standard Deviation |
|---|---|---|---|
| Translating Camera | DFD | 6.39 per pixel | 1.19 per pixel |
| Television | DFD | 6.63 per pixel | 2.21 per pixel |
| Walk Straight | Vector Difference $E_m$ | 0.0 pixels | 0.0 pixels |
| 7 UP | $E_m$ | 0.374 pixels | 0.490 pixels |

as the criteria for evaluating the algorithm, whereas vector difference between known and estimated displacements was used as an error measure for image sequences with known 2D motion fields.

Results show that this algorithm accurately estimates camera ego-motion even for image sequences having small object motion. If there is a big moving object near the image center, the algorithm is sometimes fooled by the object motion and considers it as camera motion. The accuracy of the algorithm for static scenes was quite high, and the achieved reduction in computational cost made the trade off well justified.

## References


[1] Marco Grimm and Rolf-Rainer Grigat. Real-time hybrid pose estimation from vision and inertial data. *Proceedings of the 1st IEEE Canadian Conference on Computer and Robot Vision (CRV), London, Canada*, pages 480–486, May 2004.

[2] H. Broszio and O. Grau. Robust estimation of camera parameters pan, tilt and zoom for integration of virtual objects into video sequences. *International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging, Santorini, Greece*, September 1999.

[3] H. Richter, A. Smolic, B. Stabernack, and E. Mueller. Real time global motion estimation for an MPEG-4 video encoder. *Proceedings of PCS2001, Seoul, Korea.*, April 2001.

[4] R. Schaefer, T. Wiegand, and H. Schwarz. The emerging H.264/AVC standard. *EBU TECHNICAL REVIEW*, Jan. 2003.

[5] Ryan C. Jones, D. DeMenthon, and D. S. Doermann. Building mosaics from video using MPEG motion vectors. *University of Maryland, College Park, Language and Media Processing Laboratory No. 035*, 1999.

[6] Jae-Choon Chon, Hyong-Suk Kim, Chung-Hyun Ahn, and Kyung-Ok Kim. The computation of the real-time camera motion through a correlation and a dynamic model of the system. *8th IEEE conference on Mechatronics and Machine Vision in practice, Hong Kong.*, 2001.

[7] R. M. A. Fischler and C. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[8] Christoph Stiller and Janusz Konrad. Estimating motion in image sequences: A tutorial on modeling and computation of 2D motion. *IEEE Signal Processing Magazine*, July 1999.

[9] S. M. Smith and M. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[10] `http://www.cs.brown.edu/people/black/images.html`.

[11] `http://www.cse.cuhk.edu.hk/~vision/demo/inmsfm/`.