ESTIMATING PHYSICAL CAMERA PARAMETERS FOR 3DAV VIDEO CODING

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

[a]Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands
[b]LogicaCMG, 5605 JB Eindhoven, The Netherlands
d.s.farin@tue.nl and peter.de.with@logicacmg.com

While video images are currently considered as two-dimensional (2D) entities, a recent trend is to extend video towards a 3D representation. In current video-coding algorithms like MPEG-4, global camera-motion is described as a set of abstract parameters, which have no direct relation to the physical camera movement. To bring physical meaning to these parameters, we present a new algorithm for estimating the extrinsic camera parameters and the varying focal length of a rotating camera from the MPEG-4 perspective motion-model parameters. Most previous camera calibration algorithms did not consider calibration from video data and give inaccurate results if rotation angles are small. Our algorithm applies a global non-linear optimization that is more robust on video sequences and achieves a very high estimation accuracy. The optimization is carried out using a four-stage algorithm, hereby reducing the total number of iterations in the gradient-descent search by a factor of 150.

1. MOTIVATION

A recent trend in video coding is to think about video in a more general way instead of only moving rectangular pictures. One direction in this new view is object-oriented video coding, as has been proposed in MPEG-4. A further step, that is currently discussed in MPEG, is *3D Audio Visual* (3DAV) coding [5]. This widens the term *video* to include omnidirectional (panoramic) video, multi-view rendering, or free viewpoint video. In this process, the video representation gradually evolves from purely 2D signal to a 3D data. To achieve this evolution into the 3D world, not only the video data itself, but also meta-data about the recoding setup is required, specifically including camera calibration information. Descriptors for storing camera parameters like the rotation angles and focal length have already been proposed as an extension to MPEG-7.

Knowing the physical camera parameters instead of only an abstract representation allows for many interesting applications. One example could be to imitate the camera motion from the original sequence with an automatic pan/tilt/zoom-camera, placed in a new environment. The obtained background sequence could then be used to show the foreground objects in the new environment.

In the video coding community, camera motion in video sequences is usually described as a geometric transformation in the image plane. Popular models include affine motion or the perspective motion model employed in the MPEG-4 standard for the background sprite coding tools. The parameters of the affine motion model can be easily interpreted in terms of physical meaning, but the model is even not general enough to include camera panning, which is probably the most common camera operation. It is known that the perspective motion model can describe arbitrary camera motion except camera translation. However, it is only applicable to a limited field of view [4], and its parameters cannot be easily interpreted in a physical sense.

2. PROBLEM STATEMENT

Let us consider the image formation process of a rotating camera which is allowed to change its focal length. By choosing the world coordinate system such that the optical center of the camera is located in the origin, each coordinate $(x; y)$ in image i is mapped to coordinates $(x'; y') = (\hat{x}'/\hat{w}'; \hat{y}'/\hat{w}')$ in the reference image 0 according to

$$\begin{pmatrix} \hat{x}' \\ \hat{y}' \\ \hat{w}' \end{pmatrix} = \underbrace{\begin{bmatrix} f_0 & 0 & o_x \\ 0 & f_0 & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}_0} \cdot \underbrace{\mathbf{R}_\alpha^y \cdot \mathbf{R}_\beta^x \cdot \mathbf{R}_\gamma^z}_{\mathbf{R_i}=\{\mathbf{r_{kl}}\}} \cdot \underbrace{\begin{bmatrix} 1/f_i & 0 & -o_x/f_i \\ 0 & 1/f_i & -o_y/f_i \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}_i^{-1}} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad (1)$$

where f_0 and f_i are the focal lengths corresponding to the reference image and frame i, respectively, o_x, o_y denote the camera principal point which we assume to be in the image center, and $\mathbf{R}_\alpha^y, \mathbf{R}_\beta^x, \mathbf{R}_\gamma^z$ are the Euler-rotation matrices describing the rotation around the three axes [2]. Note that different orderings of the rotation matrices are also possible, which will result in differing rotation angles. We chose this specific order, since the obtained angles correspond well to human intuition about rotation. By moving the image coordinate-system origin into the image center, we can set $o_x = o_y = 0$ in the following, leading to a simplified notation. Furthermore, we denote the camera transformations according to Eq. (1)

by T_x, T_y for the horizontal and vertical components, so that we can write the transformation in short form as $(x';y') = \big(T_x(x,y), T_y(x,y)\big)$. By multiplying out the matrix equation (1), we obtain the perspective motion model that is usually employed to describe camera motion:

$$\tilde{x} = H_x(x,y) = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}, \quad \tilde{y} = H_y(x,y) = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}.$$

Usually, the normalization $h_{22} = 1$ is applied to remove the scaling invariance of the parameters. Describing camera motion by the abstract parameters $\{h_{kl}\}$ is a common way to describe global motion. Based on these abstract parameters, which we get from the global motion estimator, we want to obtain the underlying physical parameters $f_i, f_0, \alpha_i, \beta_i, \gamma_i$ used in the formulation of T_x, T_y.

3. PREVIOUS WORK

Camera calibration is an active field of research, but up to now, it has been applied mostly to determine the intrinsic camera parameters for static setups. Usually, a calibration pattern is recorded several times at varying positions to draw conclusions about the camera's internal parameters $\mathbf{F_0}$. In [2], it is observed that for a rotating camera, the perspective transformation parameters are related to the image formation equation $\mathbf{H}_i = \mathbf{F}_0\mathbf{R}_i{\mathbf{F}_i}^{-1}$ by

$$\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \sim \begin{bmatrix} r_{00} & r_{01} & r_{02}f_0 \\ r_{10} & r_{11} & r_{12}f_0 \\ r_{20}/f_i & r_{21}/f_i & r_{22}f_0/f_i \end{bmatrix}, \tag{2}$$

and we can exploit the fact that the rotation matrix $\{r_{kl}\}$ is orthonormal. By forming equations describing the constraints of orthogonality and equal norm of the first two columns of the rotation matrix $\{r_{kl}\}$, we get

$$h_{00}h_{10} + h_{01}h_{11} + h_{02}h_{12}/f_0^2 = 0 \quad \text{(for orthogonality)} \quad \text{and} \tag{3}$$

$$h_{00}^2 + h_{01}^2 + h_{02}^2/f_0^2 = h_{10}^2 + h_{11}^2 + h_{12}^2/f_0^2 \quad \text{(for equal norms)}. \tag{4}$$

After some manipulation, we obtain the following expressions for estimating f_0:

$$f_0 = \sqrt{\frac{-h_{02}h_{12}}{h_{00}h_{10} + h_{01}h_{11}}} \quad \text{and} \quad f_0 = \sqrt{\frac{h_{02}^2 - h_{12}^2}{h_{10}^2 + h_{11}^2 - h_{00}^2 - h_{01}^2}}. \tag{5}$$

Similar equations can be obtained to calculate f_i. While this seems to be an easy way to determine f_0, the computation is not robust to noise in the input parameters, especially if the camera rotation angle is small. To see this, note that $\{r_{kl}\}$ approaches the identity matrix for small rotation angles and hence, $r_{kl} \approx 0$ for $k \neq l$, and $r_{kk} \approx 1$. Consequently, both numerators and denominators approach zero and the equation gets numerically unstable.

Agapito *et al.* [3] also present an algorithm for camera calibration with varying intrinsic parameters. They propose to estimate input parameters for all frames of a sequence at once by minimizing the algebraic distance

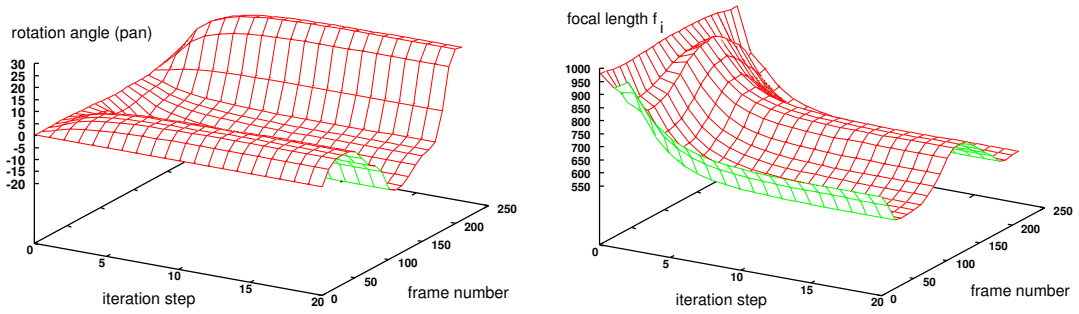
$$\sum_{i}^{n} ||F_i F_i^T - H_i F_0 F_0^T H_i^T||_F^2, \tag{6}$$

where $|| \cdot ||_F^2$ is the Frobenius norm. While their global optimization framework alleviates the numerical instabilities, their minimization criterion is an algebraic distance, which is not related to physical meaning. This reduces the accuracy, since it shows unequal weighting of the error for different kinds of motion.

4. OUR CAMERA-PARAMETER ESTIMATION ALGORITHM

In our algorithm, we estimate the camera parameters directly using a non-linear optimization framework. After the preceding global motion-estimation stage, we have a set of parameters $\{h_{kl}\}$ for each frame, which relate a pixel position $(x; y)$ in frame i to a corresponding position in the reference frame $(\tilde{x}; \tilde{y}) = (H_x^{(i)}; H_y^{(i)})$. Note that the transformation between each pair of images involves two focal lengths. If we would perform the transformation independently for each pair of images, we would get conflicting focal length estimates for the same image. However, by forming the additional constraints requiring that all focal length estimates for a single image should be equal, we can increase the estimation accuracy. On the other hand, since the constraint links the parameters between images, the optimization cannot be carried out independently on pairs of images anymore, but must now be computed over all parameters simultaneously.

As we have seen in the last section, the focal lengths can be estimated more accurately for larger rotation angles. Hence, we chose to compute all parameters relative to a fixed reference frame instead of pairs of neighboring frames. Specifically, we fix the first frame as the reference frame and compute all parameters relative to this frame. As a natural way to define the fitting accuracy between two frames, we can measure the displacement of the same image position between the



(a) Rotation parameters. (b) Focal length parameters.

Figure 1: Convergence speed of rotation angles and focal length ratios. A good solution is reached after only a few iterations.

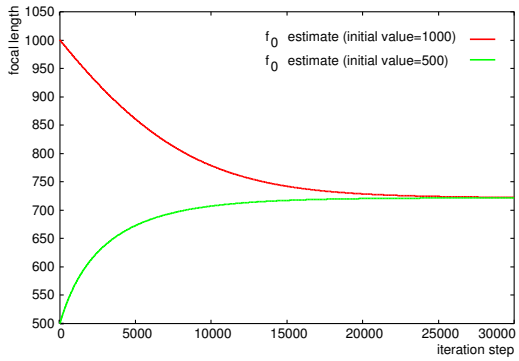
original position after global motion compensation $(H_x(x,y), H_y(x,y))$ and the transformed coordinate using physical camera parameters $(T_y(x,y), T_y(x,y))$. In order to achieve high accuracy, we directly use this definition of accuracy in our parameter estimation process. More specifically, we define the matching error as the sum of the squared displacements at the image corner positions. We sum the total estimation errors over the complete sequence to define the cost functional

$$E = \sum_{i\in[1;N]} \sum_{(x;y)\in P} \left(H_x^{(i)}(x,y) - T_x^{(i)}(x,y)\right)^2 + \left(H_y^{(i)}(x,y) - T_y^{(i)}(x,y)\right)^2. \quad (7)$$

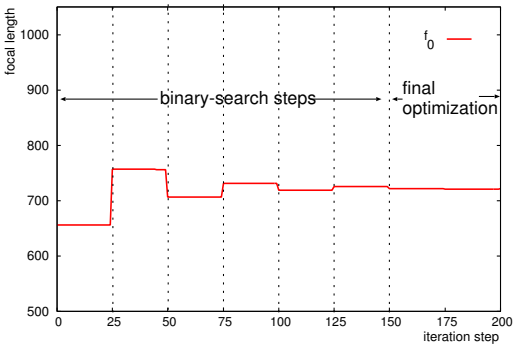
$P = \{(\pm w; \pm h)\}$ denotes the set of all image corner coordinates and w, h represent the image resolution. The parameters for all image transformations $T^{(i)}$ are collected in a large parameter vector $v = (f_0; f_1; \alpha_1; \beta_1; \gamma_1; f_2; \alpha_2; \beta_2; \gamma_2; \ldots)$ of length $4N-3$. In order to find the best parameter vector $v^\star = \arg\min_v E$, the Levenberg-Marquardt algorithm can be used.

5. CONVERGENCE BEHAVIOUR AND FAST ALGORITHM

In our first experiments, we initialized the optimization with all rotation angles set to zero and a fixed focal length set to some sensible but arbitrary value. We observed the following convergence behaviour. The rotation angles and ratios of focal lengths f_0/f_i converge after only about 10 iterations (Fig. 1) up to an overall scaling factor, which is depending on the absolute focal length. On the other hand, the convergence of the absolute focal length is much slower. Usually, up to 50,000 iterations were required, which makes the algorithm impractical (Fig. 2a).



(a) Standard Levenberg-Marquardt optimization (30,000 iterations).



(b) Our combined 4-stage optimization (200 iterations).

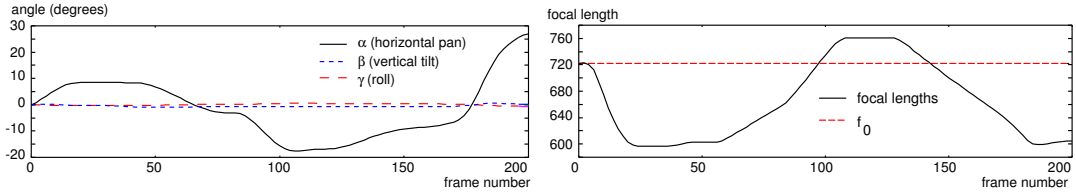
Figure 2: Convergence speed of f_0 parameter. Note the different scaling at the horizontal axes.

While convergence of rotation angles is fast, the main problem is to determine the correct focal lengths quickly. Since it is not the ratio between focal lengths f_i/f_0 that is difficult to estimate, but their absolute size, it is sufficient to get a fast solution for f_0. We apply the four-stage algorithm illustrated in Figure 5 to search for f_0. Herein, the block *ShortESTIM* denotes 25 iterations of the Levenberg-Marquardt algorithm with a fixed f_0 during the first 20 iterations. From the behaviour of f_0 during the last iterations, it can be concluded whether its value should be decreased or increased. Figure 2b illustrates the values of f_0 during the fast convergence process.

6. EXPERIMENTS AND RESULTS

We applied our algorithm to the well known *stefan* test-sequence. The obtained camera parameters are shown in Figure 3. Since there is no ground-truth data available for the sequence, we can only judge the result by visual examination. According to this, the result accurately corresponds to the visible camera motion. The obtained camera parameters can be used, e.g., to place the input image planes into a 3D space such that the placement corresponds to their virtual recording position. Figure 4 shows a visualization for the *stefan* sequence. Viewed from the camera position, all images fit together without alignment errors.

The computation time for the algorithm (excluding the preceding global-motion estimation) on a 2.8 GHz Pentium IV is about 2 seconds for 20 frames (calculated over a 200 frame sequence, only considering every 10^{th} frame).



(a) Rotation angles. (b) Focal lengths.

Figure 3: Computed camera parameters for the first 200 frames of the *stefan* sequence.

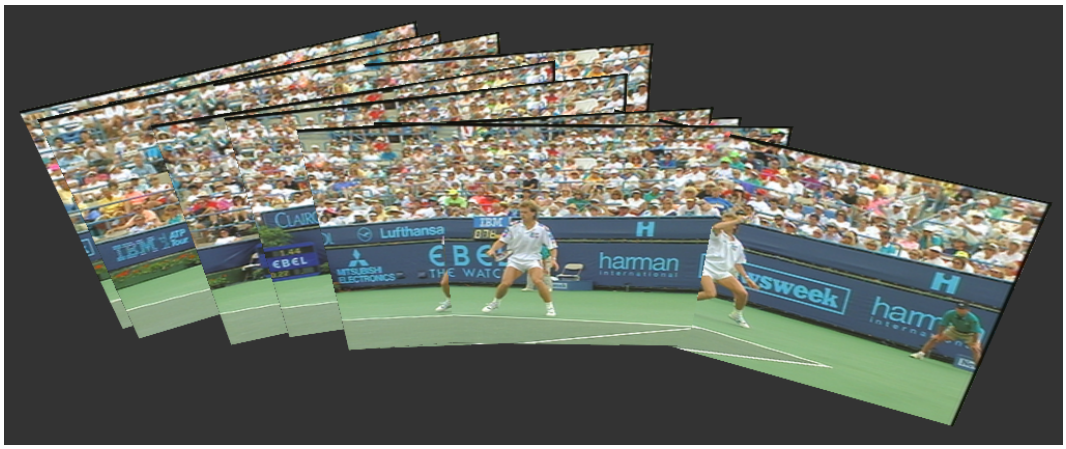


Figure 4: Image planes of every 10^{th} frame from the *Stefan*-sequence.

7. CONCLUSIONS

We presented a new algorithm to obtain the physical camera parameters (three rotation angles and the focal length) of each frame of a video sequence for a rotating camera. Differing from previous work, we apply a global optimization which achieves high accuracy since global consistency is obeyed. Using a specialized optimization algorithm, it was possible to decrease the number of iterations from about 30,000 for the unmodified algorithm to only 200, which is a factor of 150. This reduces the computation time to about 2 seconds for 20 input frames on a 2.8 GHz Pentium IV.

Since the algorithm is only based on the parameters of the perspective motion model, which is also used in MPEG-4 for transmitting current camera positions in sprite mode, the algorithm can also be used in the receiver on the decoded data. Possible applications include the generation of spherical panoramas from MPEG-4 background sprites without requiring complex recomputation of the global camera motion.

REFERENCES

[1] Richard Hartley, Andrew Zisserman. Multiple View Geometry in Computer Vision. *Cambridge University Press.*

[2] Richard Szeliski, Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Computer Graphics, Annual Conference Series*, pp. 251-258, 1997.

[3] L. De Agapito, E. Hayman, I. D. Reid. Self-Calibration of a Rotating Camera with Varying Intrinsic Parameters. In *Proc.* 9^{th} *British Machine Vision Conference (BMVC)*, 1998.

[4] Dirk Farin, Peter H. N. de With, Wolfgang Effelsberg. Minimizing MPEG-4 Sprite Coding-Cost Using Multi-Sprites. In *SPIE Proc. Visual Communications and Image Processing*, 2004.

[5] A. Smolic, H. Kimata. Report on 3DAV Exploration. MPEG ISO/IEC JTC1/SC29/WG11 Document N5878, July 2003.

[6] R. I. Hartley Self-calibration from multiple views of a rotating camera. In *Third European Conference on Computer Vision (ECCV'94)*, volume 1, pp. 471–478.

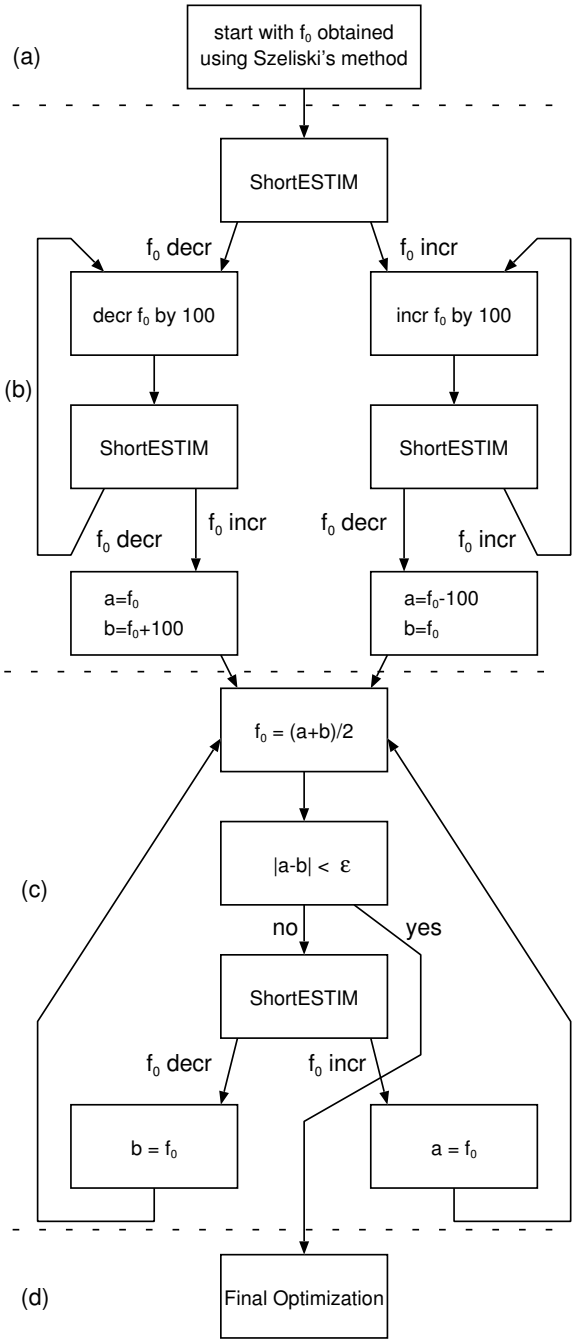


Figure 5: Algorithm for fast computation of focal lengths. (a) Initialization of f_0. (b) Searching for a range $a; b$, so that $a \leq f_0 \leq b$. (c) Applying a binary search to determine f_0. (d) Global optimization over all parameters.