Recent Developments in Panoramic Image Generation and Sprite Coding

Dirk Farin #, Martin Haller *, Andreas Krutz *, Thomas Sikora *

# *Robert-Bosch GmbH, Corporate Research*[1]
P.O.Box 77 77 77, 31132 Hildesheim, Germany
dirk.farin@gmail.com

* *Communication System Group, Technical University of Berlin*
Einsteinufer 17, 10587 Berlin, Germany
{haller,krutz,sikora}@nue.tu-berlin.de

Abstract—[1] **The composition of panoramic images has recently received considerable attention. While panoramic images were first used mainly as a flexible visualization technique, they also found application in video coding, video enhancement, format conversion, and content analysis. The topic has enlarged and diverged into many specialized research directions, which makes it difficult to stay in touch with recent developments. This paper intends to give an overview of the current state of research, including recent developments. Two of the applications of sprite coding and global-motion estimation are presented in more detail to provide some insights into the system aspects.**

I. Introduction

The environment we live in is large; much larger than we can capture in a single image, especially if it is in 4:3 format. For this reason, the movie-making artists have developed techniques like panning or zooming into the scene to be able to gradually show it all on a limited display. While these cinematographic techniques help to draw our attention to part of a large scene, we lose the ability to perceive the scene as a whole.

The consequences for video-coding are that the cinematographic effects transform a previously simple, static scene into moving video that is relatively inefficient to code. Furthermore, we lose flexibility, i.e., the path along which we observe the scene is pre-defined by the video we watch, instead of giving the user the choice of selecting the view of the scene he is most interested to see. Panoramic imaging and sprite coding are techniques that aim at reversing the process, i.e., to reconstruct the large scene back from the limited input video or pictures in order to provide better visualization or increase the achievable compression ratio.

This paper intends to give an overview of the current state of research in each of the areas required for mosaic generation. Specifically, we consider the options for a suitable sprite representation, camera-motion estimation, and mosaic compositing. Furthermore, we give a brief list of possible applications and provide more details for two of them.

[1]This research was carried out while the first author was with the Technical University Eindhoven, Netherlands.

II. Motion and Projection Models

Probably the most central design decision in building a panoramic image generation program is the choice of a representation for the panoramic image. This choice depends on the type of scene observed, the types of camera motion we expect as input, and also on the *a-priori* information we have about the camera (e.g., is the focal-length known?).

A. Translational motion

The simplest assumption possible is that the transformation between frames is purely translational. Because of the small number of parameters, this model is generally very robust to compute. For this reason, it is often useful to first estimate the translatorial component of a transformation, and then extending the model to, e.g., affine or projective. Moreover, when the focal-length of the camera is known, the cylindrical projection (see below) can be reduced to the translational case by a pre-transformation.

B. Affine motion

The group of affine transformations includes all rigid transforms in the 2-D plane. However, it is popular to use the affine model as an approximation when the focal length is large and objects are at a far distance. In this case, the camera geometry approaches parallel projection, which results in affine motion in the image plane.

C. Projective motion

Projective motion is defined by

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \;\; ; \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}, \quad (1)$$

and is often normalized with $h_{22} = 1$. Affine motion is a subclass of projective motion for which $h_{20} = h_{21} = 0$. Projective motion can describe two common cases of 3-D motion: the motion of a planar scene under arbitrary rigid 3-D motion, and the motion of a general scene under rotational camera motion (including zoom). The former case is a good approximation for distant scenes, where the depth variation is small compared to the scene distance (e.g., aerial observations). The latter case is a very frequent situation covering scenes captured by cameras mounted on a fixed tripod.

D. Multi-Sprites

Applying the projective motion model can be envisioned as projecting the scene onto a flat painting or a window, through which we are looking at a real scene. The problem that we face with this representation is that when we turn the camera away from this fixed plane, the area required to project the camera picture onto the plane increases quickly. When we exceed a rotation angle of 90°, the image cannot be projected onto the sprite at all. Hence, while the projective model is theoretically able to represent rotational camera-motion, the representation is inefficient in terms of storage requirements. One solution would be to switch to a cube-map representation (see below), which is basically a collection of six planar sprites. However, the generation of a cube-map requires the focal length of the camera to be known and physical rotation angles, which are difficult to obtain with high precision.

A solution to this problem has been proposed in [1]. Instead of a single sprite to represent the background, multiple sprites are used that are optimally arranged such that the total area required to map the background scene onto the sprites is minimized. At the same time, the multi-sprite technique elegantly solves the zoom-problem, which refers to choosing a spatial resolution for the sprite at which no detail is lost, but which is still memory efficient. Note that this problem is not solved by using a cube-map representation.

Recently, alternative multi-sprite construction algorithms have been proposed that base the sprite separation on the physical camera rotation [2] or efficiently handle reoccuring backgrounds [3].

E. Biquadratic motion model

As an approximation to the projective motion model, a second-order Taylor expansion is sometimes employed [4]. This may result in simpler optimization techniques, but it should be noted that this model only provides enough accuracy for small motion and is unsuitable for panoramic image compositing. Another difficulty with this model is that it is not closed under inversion and concatenation.

F. Cylindrical projection

For a pure camera rotation around the vertical axis, the scene content can be mapped on the surface of a cylinder. The cylinder-coordinates (θ, h) can be computed from image coordinates (x, y) by

$$\theta = \theta_i + \arctan x/f \quad \text{and} \quad h = \frac{y \cdot r}{\sqrt{f^2 + x^2}}, \tag{2}$$

where f is the focal length of the camera and r is the radius of the virtual cylinder. The different rotation angles of the camera when capturing the individual images are denoted as θ_i. This transformation only has two unknowns per frame (f and θ_i). In many situations, f can be considered constant, so that only θ_i has to be computed for each frame. Since a change in θ_i only corresponds to a shift on the cylinder, this can be computed easily with a simple translational motion-estimator like proposed in [5]. Computation of f is more complex and should be computed globally to ensure that the cylinder is *closed* correctly.

G. Spherical projection

Instead of mapping the scene onto a cylinder, we can alternatively map it onto a sphere to cover also vertical rotation. While the mapping to spherical coordinates is not particularly difficult, the local resolution of the texture map varies considerably over the image and even shows singularities at the poles. Moreover, estimating the camera rotation angles can be tricky in the polar region when using an Euler-angle representation. A more stable computation can be obtained by representing the camera rotation with quaternions [6]. To avoid the large variation in resolution, the mosaic can also be projected onto a cube-map, which is also more efficient to process since each cube-surface can be handled with a projective transformation as described above.

III. Image Registration

In order to align all input frames into a larger mosaic image, the parameters of the chosen motion model have to be determined. The two main challenges are the high accuracy required to avoid a distorted or blurred mosaic, and the robustness to foreground-object motion than can confuse the motion-estimation process. The available motion-estimation algorithms can be classified into the two groups of *direct estimation* algorithms and *feature-based* algorithms. Each of these two classes shows different performance with respect to accuracy and robustness, and it can be advantageous to combine both in a hybrid motion-estimation algorithm. We will now describe these approaches in further detail and finally show how to upgrade algebraic parameters to physically meaningful parameters by camera auto-calibration.

A. Direct Estimation

The *direct methods* [4], [13] for motion estimation go back to the work of Horn and Schunck [14] in which they state the basic assumption that the brightness I of a moving pixel does not change over time. This can be expressed as $I_t(x, y) = I_{t+1}(x + u(x, y), y + v(x, y))$, where u, v denotes the motion field. Assuming small motion, this leads to one constraint for every pixel

$$\partial I/\partial x \cdot u + \partial I/\partial y \cdot v + \partial I/\partial t = 0. \tag{3}$$

Substituting a parametric motion-model for u, v reduces the number of parameters to far less than we have pixels in the image and we cannot expect to fulfill the equation perfectly. Hence, we can only sum the errors for every pixel and minimize this total error to find the best fitting parameters.

This approach usually leads to a very high accuracy, but has two main problems: it only works for small motions and it is sensitive to foreground-object motion, especially if this object has a large contrast to the background. The first problem can be approached by iterating the motion-estimation process several times until it converges and applying a hierarchical strategy, in which the parameters are first computed on a (potentially several times) downscaled version and then scaled to a higher resolution, at which the parameters are refined further [15]. Making the algorithm more robust to foreground motion can be realized by employing an M-estimator instead of finding the least-squares solution [16]. This reduces the influence of strong outliers, which are in this case the foreground-object pixels that have a significantly different color from the background.

B. Feature-Based Estimation

The complementary approach to motion estimation are the feature-based algorithms [17]. Here, a number of *feature-points* are detected in each frame, correspondences between the points are established for successive frames, and finally, a motion-model is fitted to the correspondences. Each of these three steps can be realized in a number of ways. For motion-estimation in video, a standard Harris-detector [18] with a straight-forward matching of image-patches around the features works well, because the motion between successive frames is small and smooth. For large transformations, other feature-detection and matching approaches are required. This has led to research in the area of affine-invariant features [19], [20]. One recently very successful approach are SIFT-features that have proven stable to a wide variety of transformations and also illumination changes [21].

The feature-matching step results in a set of corresponding coordinates $(x_i, y_i) \leftrightarrow (x'_i, y'_i)$ between the two frames. If enough of these correspondences are collected, an overdetermined system of equations $x_i + u(x_i, y_i) - x'_i = 0$, $y_i + v(x_i, y_i) - y'_i = 0$ is obtained, which can be solved in a least-squares sense to determine the parameters of u, v.

One advantage of the feature-based motion estimation is that it offers good control to separate coexistent motions in the same image. The standard approach is to use a RANSAC-like algorithm [22], [23] that allows to compute the parameters of the dominant motion, i.e., the motion-field that covers the most feature-vectors. Assuming that background-motion is dominant in the image, the algorithm computes its motion-parameters untainted with foreground-motion.

C. Hybrid Estimation

In general, the feature-based motion-estimators do not reach the accuracy of the direct estimators. This is caused first by the feature-point detection, which always has some noise in the localization of features, especially when two corresponding features between geometrically transformed images are considered. On the other hand, for some motion models (including the projective motion), a minimization of the Euclidean-error metric is difficult, so that in practice a geometrically non-sense, but easier to handle algebraic error is minimized, leading to a bias in the estimated parameters.

The idea of hybrid estimation is to combine a feature-based estimation with a successive direct estimation step. The feature-based estimation first computes the coarse motion even in the presence of foreground objects. These parameters are then refined in the direct estimation step. This approach compensates the disadvantages of the direct estimation (small convergence radius, sensitivity to foreground motion) with the feature-based estimation which provides a good initialisation. In fact, the property of the direct method to have a small convergence radius becomes an advantage now, because it ensures a correct convergence to the motion found by the feature-based estimator, independent of the foreground motion.

D. Camera Calibration

In some applications, it is desirable to know the physical camera-parameters (rotation angles and focal length). Also some motion models are based on physical parameters (cylindrical transform requires a known focal length, spherical projection involves estimating the camera rotation angles). Estimating these parameters directly is difficult and numerically sensitive when the motion is small. A different approach is to start with a parametric motion model like the projective model and then compute the physical parameters starting from those [24], [5]. The advantage is that many short-term transformations can be integrated in the computation and additional constraints like a fixed focal length can be exploited. There exists a vast literature on camera calibration, where auto-calibration of a rotation-only camera is the most important for panoramic image generation [25], [26].

IV. Mosaic-image Composition

Having aligned all frames in a global coordinate frame should theoretically result in a seamless mosaic, but changes in global lighting or the camera gain control can lead to inconsistent colors. Moreover, when composing the input frames into the mosaic, small misalignments from the motion-estimation can lead to broken texture. A good composition should strive to hide these easily observable misalignments.

The second challenge for the composition step is that the scene might not be completely static. Some applications require that foreground objects are removed from the mosaic, so that a static background-only mosaic is obtained. Other applications might prefer to see the foreground as part of the mosaic, which results in a dynamic mosaic in which changing parts are replaced over time [27].

It also should not be neglected that the composition step can have consequences for motion estimation if a long-term motion estimation (relative to the mosaic constructed so far) is carried out. Take for example the simple approach to compose a mosaic by continuously copying the latest input frame over the previous mosaic content. The problem with this approach is not only that seams between temporally distant frames are pronounced, but also that a long-term motion estimation will fail because in every step, most of the mosaic content that is overlapping with the next frame is replaced with the previous frame. The long-term motion estimation is in fact degenerating to a short-term motion estimation.

One solution to this effect is to set only those pixels in the mosaic to the new value that are so far undefined. This leads to good long-term motion-estimation, but shows similar seams along neighboring patches.

V. Applications

Building a panoramic mosaic is a component of many possible applications, where the mosaic is either a central part of the visualization itself (like generating panoramic pictures [29], or format conversion), or they are used transparently without being visible directly (e.g., when used for video coding).

A. Video Enhancement, Format Conversion

The extra information that becomes available from the mosaic image can be used to extend the normal view of the video. More specifically, it can be used to show video in 4:3 format on a 16:9 display and extend the video horizontally with more content that is usually invisible, but which can be extracted from the mosaic image if the camera made a panning move [30]. Similarly, this technique can be used in camera stabilization, where image content has to be filled in at the borders occasionally, too.

B. Sprite Coding (MPEG-4)

Sprite Coding [31], [32], [33], [34] was developed and standardized in MPEG-4 some years ago. The main idea of this coding approach is to segment the video content of the input video into two parts, foreground object and background object, in a pre-processing step. Furthermore, a so-called background sprite image is generated which contains all the background information of a certain number of frames of the sequence. Global motion estimation (GME) techniques initiate this sprite generation process. The background sprite image and the foreground objects are then coded separately. At the decoder, original images from the video sequence are reconstructed from the background sprite image (containing only background information). These reconstructed frames are merged with the foreground objects and the whole sequence is decoded. The sprite coding within the MPEG-4 standard is described in [35] more in detail. It has been recently shown that the latest standardized motion compensated DCT-based video codec [36] can be improved by these sprite-based object representation of the video sequence and video content [37]. This approach is described next.

C. Object-based coding using H.264/AVC

The sprite coding approach using H.264/AVC is presented in the following. An overview of the whole codec is shown in Fig. 1. After the multiple sprite generation, sprite-based robust foreground segmentation is used. Sprites, foreground textures, mask information and transformation parameters are then coded independently.

1) Object Segmentation: The segmentation approach is based on pre-computed error frames. For improving robustness and reliability, error-frames are computed for the re-projected sprite images and for the compensated adjacent frames using short-term transformation parameters. The former one is more

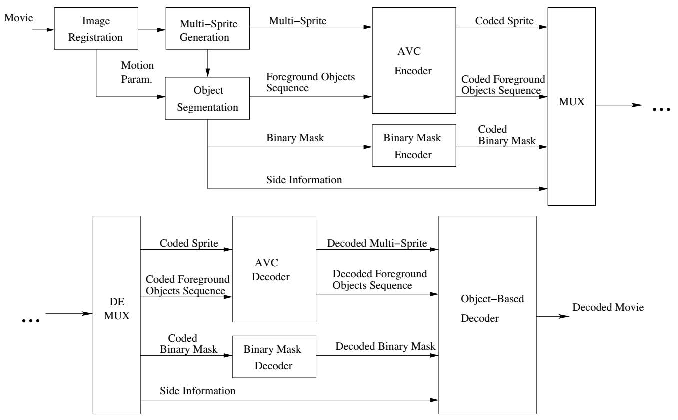

Fig. 1. Object-based video codec (OBVC) using multiple sprites and inherent segmentation

sensitive in image regions where foreground objects appear while the latter is almost errorless in the background region. Combining the results yields very precise segmentation masks. After generating the error image on the luminance channel, anisotropic Gaussian filtering is performed on the absolute values. This is achieved by locally solving the anisotropic diffusion function

$$I_t = \operatorname{div}\left(\rho(|\nabla I|)\nabla I\right). \tag{4}$$

The images are then binarized applying an adaptive threshold depending on the overall error frame energy. After morphological filtering of the binary images and removing small objects the masks are combined using a logical AND-operator.

2) Texture and Binary Mask Coding: The textures of the sprite and the foreground objects are independently coded using H.264/AVC (see Figure 1). As coding software the JSVM version 6.5 is used. To efficiently code the foreground the objects are expanded to fit to macro block structure. A hierarchical B picture scheme IBBB... is applied with a GOP length of 15 frames, i.e. direct access every 0.5 seconds [39]. The sprite itself is coded as I picture.

The binary mask coding scheme uses the binary arithmetic coder "M-Coder", which is specified in the H.264/AVC standard. Eight contexts are initialized to model 8 possible spatial states for every pixel to code. In advance to the coding, the mask is divided into 16x16 blocks. The transformation parameters are coded as four frame corner point correspondences using 3 bytes per corner point and build the side information.

D. Camera motion characterization

Camera movements within a video sequence can be analyzed by methods for camera motion characterization. A sequence of segments with homogeneous camera motion for each segment and assigned types of motion such as pan, tilt, zoom, and rotation etc. would be the ideal result.

To obtain such results automatically, the global motion has to be estimated at first. Techniques for camera motion characterization can then use global motion parameters for a characterization. The quality of analysis results depends not only on the approach for characterization. In fact, the robustness and precision of global motion estimation technique is critical. In the following four recent approaches based on global motion parameters are addressed. The last one in a little more detail.

A method applies hypothesis-driven likelihood ratio tests to six different combinations of two translational, divergence, curl, and two hyperbolic terms. The last four terms are derived from affine motion parameters as subtraction or addition of two parameters. The six combinations are associated with six classes of camera motion defined by Bouthemy et al. [40].

The characterization approach for camera motion by Porter et al. [41] uses a Douglas-Peucker algorithm for line simplification to identify segment boundaries. The approach is based on motion parameters for scale and the two translational directions. The types of camera motion are finally classified with a set of rules using the average motion parameters for each segment.

A technique for camera motion characterization based on transferable belief model has been proposed by Guironnet et al. [42] and comprises of three stages: combination based on heuristic rules, static/dynamic separation, and temporal integration of zoom and translation motion.

A method proposed by Haller et al. [43] uses a machine learning strategy to characterize the camera motion with a feature extraction, a classification, and a temporal segmentation stage. The method uses affine global motion parameters as input. These parameters are then factorized using the Singular Value Decomposition (SVD) into scaling, rotation, and skewing components. For each camera motion type, suitable features for classification are extracted from these components and the translational parameters.

Exemplarily, the features extraction for pan and tilt uses the two translational parameters $h_{02,l}$ and $h_{12,l}$ to compute four features for each motion type. The index l addresses all motion parameters for frames l and $(l+1)$. The complex normalized value

$$\underline{t}_l = \frac{h_{02,l}}{w} + j\frac{h_{12,l}}{h} \tag{5}$$

is used to determine the median angle $\phi_{\text{t,med},l}$ of translational motion and the medians $t_{\text{x,med},l}$ and $t_{\text{y,med},l}$ with

$$\begin{aligned}\phi_{\text{t,med},l} &= \operatorname*{median}_{s_l \le k \le e_l}\left(\arg\left(\underline{t}_k\right)\right) & (6)\\ t_{\text{x,med},l} &= \operatorname*{median}_{s_l \le k \le e_l}\left(h_{02,k}\right) & (7)\\ t_{\text{y,med},l} &= \operatorname*{median}_{s_l \le k \le e_l}\left(h_{12,k}\right), & (8)\end{aligned}$$

where w and h are the image width and height and s_l and e_l are given as

$$s_l = l - W_{\text{med}} + 1 \quad ; \quad e_l = l + W_{\text{med}}.$$

The median filtered parameters are robust against possible outliers coming from GME. The used windowed median filter has a length of $W_{\text{med}} = \lfloor R_f/2 \rfloor$ with R_f as frame rate per second of the video sequence.

Short-time translational angle histograms based on $\phi_{\text{t,med},l}$ are determined to obtain more robust features for the direction of translational motion. The derived rates $R_{\text{TAHPL},l}$, $R_{\text{TAHPR},l}$, $R_{\text{TAHTU},l}$, and $R_{\text{TAHTD},l}$ represent the occurrence of angles for pan left/right and tilt up/down in the respective range of angles normalized to the window length W for the histogram computation. The used overlap of windows is extensive for a proper temporal resolution.

The zero-crossing rates for horizontal translational motion parameters are defined by

$$Z_{\text{x},l} = \frac{1}{2W}\sum_{i=s_l}^{e_l}\left|\operatorname{sgn}\left(h_{02,i}\right) - \operatorname{sgn}\left(h_{02,i-1}\right)\right| \tag{9}$$

and similarly for vertical motion. These zero-crossing rates capture the reliability of intended translational motion within the window of the length W.

The complete four-dimensional feature vectors for pan and tilt are as follows

$$\begin{aligned}\mathbf{x}_{\text{pan},l} &= \begin{pmatrix} t_{\text{x,med},l} & R_{\text{TAHPL},l} & R_{\text{TAHPR},l} & Z_{\text{x},l}\end{pmatrix}^{\mathrm{T}} & (10)\\ \mathbf{x}_{\text{tilt},l} &= \begin{pmatrix} t_{\text{y,med},l} & R_{\text{TAHTU},l} & R_{\text{TAHTD},l} & Z_{\text{y},l}\end{pmatrix}^{\mathrm{T}} & (11)\end{aligned}$$

Further details for zoom features can be found in [43].

Three multi-class SVMs (M-SVMs) are then used to detect the camera motion types pan, tilt, and zoom independently for an image pair. Each M-SVM distinguishes between the occurrence and the direction of each motion type. Thus, each of the three M-SVMs provides a result with three possible states, e.g. for pan: pan left/right or no pan. Changes of camera motion are identified as boundaries of segments. Each segment has then a homogeneous camera motion.

In [43], only pan, tilt, and zoom are considered. However, this method is extendable to other camera motion types by adding a set of appropriate features for classification and obtaining enough training data to build a sufficient model.

It remains an open question, whether likelihood ratios, a set of rules, the transferable belief model, or multi-class SVM-based characterization of camera motion is the option of choice. A common set of ground truth data on a sub-shot level for camera motion characterization would be very useful for a better evaluation of different approaches and further development.

VI. Conclusion

This paper has addressed the generation of image mosaics (sprites) and some of its applications. We have tried to give a global overview that points out the essential problems to consider and various approaches for their solution. We also pointed out new developments in each area, like multi-sprite background models, SIFT features for image registration, or graph-cut based compositing.

Panoramic images have become a standard technique for visualization, video editing, and content adaptation. Sprite coding also stays a promising approach for new video coding standards despite the high computational cost and the difficulty of obtaining a good segmentation. For video coding, online sprite generation and object segmentation with short delay remains an interesting open problem.

References

[1] D. Farin and P. H. N. de With, “Enabling arbitrary rotational camera-motion using multi-sprites with minimum coding-cost,” *IEEE Transactions on Circuits and Systems for Video Technology*, accepted for publication.

[2] M. Kunter, A. Krutz, M. Mandal, and T. Sikora, “Optimal multiple sprite generation based on physical camera parameter estimation,” in *Visual Communications and Image Processing 2007. Edited by Chen, Chang Wen; Schonfeld, Dan; Luo, Jiebo. Proceedings of the SPIE, Volume 6508, pp. 65080B (2007).*, ser. Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, vol. 6508, Jan. 2007.

[3] G. Ye, “High-Resolution Multi-sprite Generation for Background Sprite Coding,” *Lecture Notes in Computer Science*, vol. 4678, p. 756, 2007.

[4] S. Mann and R. W. Picard, “Video orbits of the projective group: A simple approach to featureless estimation of parameters,” *IEEE Transactions on Image Processing*, vol. 6, no. 9, Sep. 1999.

[5] R. Szeliski and H.-Y. Shum, “Creating full view panoramic image mosaics and environment maps,” in *SIGGRAPH ’97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 251–258.

[6] S. Coorg and S. Teller, “Spherical mosaics with quaternions and dense correlation,” *International Journal on Computer Vision*, vol. 37, no. 3, pp. 259–273, 2000.

[7] J. Y. A. Wang and E. H. Adelson, “Representing moving images with layers,” *The IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, vol. 3, no. 5, pp. 625–638, September 1994.

[8] P. S. Heckbert, “Fundamentals of texture mapping and image warping,” Master’s thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720, Jun. 1989.

[9] S. B. Kang and R. Weiss, “Characterization of errors in compositing panoramic images,” in *CVPR ’97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR ’97)*. Washington, DC, USA: IEEE Computer Society, 1997, p. 103.

[10] S. Peleg and J. Herman, “Panoramic mosaics by manifold projection,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 338–343, 1997.

[11] H.-Y. Shum and R. Szeliski, “Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment,” *Int. J. Comput. Vision*, vol. 36, no. 2, pp. 101–130, 2000.

[12] L. Zelnik-Manor, G. Peters, and P. Perona, “Squaring the circles in panoramas,” in *ICCV ’05: Proceedings of the Tenth IEEE International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1292–1299.

[13] M. Irani and P. Anandan, “All about direct methods,” in *Vision Algorithms: Theory and practice*, W. Triggs, A. Zisserman, and R. Szeliski, Eds. Springer-Verlag, 1999. [Online]. Available: citeseer.ist.psu.edu/irani99all.html

[14] B. Horn and B. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[15] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani, “Hierarchical model-based motion estimation,” in *ECCV ’92: Proceedings of the Second European Conference on Computer Vision*. London, UK: Springer-Verlag, 1992, pp. 237–252.

[16] A. Smolic and J. Ohm, “Robust global motion estimation using a simplified M-estimator approach,” in *IEEE International Conference on Image Processing (ICIP)*, vol. 1, sep 2000, pp. 868–871.

[17] D. Farin and P. H. N. de With, “Evaluation of a feature-based global-motion estimation system,” in *SPIE Visual Communications and Image Processing*, Jul. 2005, pp. 1331–1342.

[18] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of The Fourth Alvey Vision Conference, Manchester*, 1988, pp. 147–151.

[19] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[20] ——, “An affine invariant interest point detector,” in *European Conference on Computer Vision (ECCV)*. Springer, 2002, pp. 128–142, copenhagen. [Online]. Available: http://www.inrialpes.fr/movi/publi/Publications/2002/MS02

[21] D. Lowe, “Distinctive image features from scale-invariant keypoints,” in *International Journal of Computer Vision*, vol. 20, 2003, pp. 91–110. [Online]. Available: citeseer.ist.psu.edu/lowe04distinctive.html

[22] 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*, vol. 24, no. 6, pp. 381–395, 1981.

[23] P. H. S. Torr and A. Zisserman, “MLESAC: a new robust estimator with application to estimating image geometry,” *Compututer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[24] D. Farin and P. H. N. de With, “Estimating physical camera parameters based on multi-sprite motion estimation,” in *SPIE Image and Video Communications and Processing, Vol. 5685*, vol. 5685, Jan. 2005, pp. 489–500.

[25] L. Agapito, E. Hayman, and I. Reid, “Self-calibration of rotating and zooming cameras,” *International Journal of Computer Vision*, vol. 45, no. 2, pp. 107–127, 2001.

[26] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[27] M. Irani, P. Anandan, and S. Hsu, “Mosaic based representations of video sequences and their applications,” *International Conference on Computer Vision*, pp. 605–611, 1995.

[28] R. Kumar, P. Anandan, M. Irani, J. Bergen, and K. Hanna, “Representation of scenes from collections of images,” *IEEE Workshop on Representations of Visual Scenes*, pp. 10–17, 1995.

[29] M. Massey and W. Bender, “Salient stills: Process and practice,” *IBM Systems Journal*, vol. 35, no. 3&4, pp. 557–573, 1996.

[30] P. C. McLean, “Structured video coding,” Master’s thesis, Media Arts and Sciences Section, School of Architecture and Planning, Massachusetts Institute of Technology, Jun. 1991.

[31] F. Dufaux and F. Moscheni, “Background mosaicking for low bit rate video coding,” *Image Processing, 1996. Proceedings., International Conference on*, vol. 1, 1996.

[32] A. Smolic, T. Sikora, and J.-R. Ohm, “Long-term global motion estimation and its application for sprite coding,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, no. 8, pp. 1227–1242, December 1998.

[33] M. C. Lee, W. Chen, C. B. Lin, C. Gu, T. Markoc, S. I. Zabinsky, and R. Szeliski, “A layered video object coding system using sprite and affine motion model,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 130–145, Feb. 1997.

[34] T. Sikora, “Trends and perspectives in image and video coding,” *Proceedings of the IEEE*, vol. 93, pp. 6–17, January 2005.

[35] F. Pereira and T. Ebrahimil, *The MPEG-4 Book*. IMSC press, 2002.

[36] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the h.264/avc video coding standard,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 302–311, July 2003.

[37] M. Kunter, A. Krutz, M. Droese, M. Frater, and T. Sikora, “Object-based multiple sprite coding of unsegmented videos using h.264/avc,” in *IEEE Int. Conf. on Image Processing (ICIP2007)*, San Antonio, USA, Sep. 2007.

[38] D. Farin, P. H. N. de With, and W. Effelsberg, “Video-object segmentation using multi-sprite background subtraction,” in *IEEE International Conference on Multimedia and Expo (ICME)*, vol. 1, Jun. 2004, pp. 343–346.

[39] H. Schwarz, D. Marpe, and T. Wiegand, “Analysis of hierarchical b pictures and mctf,” in *IEEE Int. Conference on Multimedia & Expo (ICME’06)*, Toronto, Canada, Jul. 2006.

[40] P. Bouthemy, M. Gelgon, and F. Ganansia, “A unified approach to shot change detection and camera motion characterization,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 1030–1044, 1999.

[41] S. Porter, M. Mirmehdi, and B. Thomas, “Video indexing using motion estimation,” in *Proceedings of the 14th Bristish Machine Vision Conference*, Sep. 2003, pp. 659–668. [Online]. Available: http://www.bmva.ac.uk/bmvc/2003/papers/112/paper112.pdf

[42] M. Guironnet, D. Pellerin, and M. Rombaut, “Camera motion classification based on transferable belief model,” in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Sep. 2006.

[43] M. Haller, A. Krutz, and T. Sikora, “A generic approach for motion-based video parsing,” in *Proceedings of the 15th European Signal Processing Conference (EUSIPCO)*, Poznań, Poland, Sep. 2007, pp. 1377–1381.