Application of epipolar rectification algorithm in 3D Television

Jakub Stankowski and Krzysztof Klimaszewski

Poznan University of Technology Chair of Multimedia Telecommunictaions and Microelectronics Polanka 3, 60-965 Poznan, Poland
jstankowski@multimedia.edu.pl, kklima@et.put.poznan.pl

Summary. In the paper a method for performing rectification of video from cameras in 3D television parallel multi-camera setups is presented. Using the camera calibration data gathered, a new coordinate system is calculated, in which virtual camera positions and orientations are calculated. A rectifying perspective transform is calculated that performs transformation from the real camera coordinate system to a new system. Transformed images correspond to images from virtual rectified camera setup. The results obtained using the method described are verified by computation of depth maps using the rectified video.

1 Introduction

In 3D television systems considered in contemporary research activities, a video of a scene is captured by a number of synchronized cameras. In most cases 3 or more cameras are used. This enables to produce virtual views from the data gathered.

The virtual view generation from raw views is a complex process, so to lighten the computational burden and to make it possible for a simple 3D television receiver to be able to produce virtual views, simplifying assumptions are made. The first one is that all cameras are placed along a straight line and have parallel optical axes. The second assumption is that 3D scene structure is known and given in a form of depth map. This approach is used in the MPEG group research activities regarding 3D television systems [1].

Efficient computation of depth maps is possible when all scene elements are projected to the images from consecutive cameras in a way that they lie on the same image line. This ensures that proper correspondences between camera views can be found in the same line of images from all cameras. In order to satisfy this requirement, all cameras need to be aligned so that their direction is identical and perpendicular to the straight line, along which the cameras are placed. This way of camera placement makes it easy to compute depth maps and, additionally, satisfies the first assumption specified above. Unfortunately,

it is practically impossible to position the cameras in such way. This is because cameras have different intrinsic and extrinsic parameters. Image sensors and optical system in the cameras, even from the same production batch, can differ in orientation and position significantly. It is also very difficult to ensure proper camera body alignment. A difference of 2 angular minutes results in a 1 pixel shift of the whole image.

Therefore images acquired by a camera system have to be transformed in a way that simulates the ideal camera position and orientation. Such preprocessing is called rectification.

The purpose of rectification is to produce artificial views that would be captured by hypothetical cameras with parallel camera axes, identical intrinsic parameters, and camera centers positioned along a straight line with all the image horizontal borders being parallel to the line of camera centers.

The simplest possible case, with 2 cameras in the system, has a solution that assures ideal rectification using perspective image transform. Ideal rectification is possible because two points in space always lie on a straight line, therefore it is only necessary to compensate intrinsic camera parameters differences and camera rotations. Algorithms for estimation of the camera parameters are given in [8][9][12], while the rectification algorithms for 2 camera case can be found in [2][10][11].

For 3 and more cameras there exist no theoretical solution for ideal rectification using image perspective transform. Therefore, the proposed algorithm, that is based on perspective transform, can only produce approximations of ideally rectified images. Despite of that, it performs very well, giving useful results.

Only little information about other approximate solutions can be found in literature, like those presented in [6][7], however, according to authors knowledge, no implementation of this method is publicly available. Moreover, the article does not describe the whole process of performing rectification. In this article we present and describe in detail our original proposal of an algorithm of multiple camera system rectification algorithm.

2 Pinhole camera model

In our work we use the most popular camera model, which is the pinhole camera model [5]. In this model one assumes that projection of a scene on an image plane can be described by the following equation:

$$z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \, [\mathbf{R}|\mathbf{T}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{1}$$

where: $\mathbf{A}$ is [3x3] intrinsic matrix, containing information about camera principal point and focal length. It also contains information about physical image

cell proportions and skew of the consecutive image lines, **R** is [3x3] rotation matrix, carrying information about camera orientation, **T** is [3x1] translation vector, containing information about camera location, X,Y,Z are real world 3D coordinates of a point, u, v are image coordinates, z is scaling coefficient.

To model the distortions caused by the real camera optics, additional distortion parameters are introduced.

3 Proposed rectification algorithm

The proposed algorithm was developed for the parallel multi-camera system, like the one used in Poznan University of Technology [14]. The algorithm makes use of procedures implemented in OpenCV library [3].

3.1 Camera calibration

The first step in the solution proposed is to calibrate all cameras. We need to calculate values of intrinsic matrix **A**, distortion coefficients **D**, rotation matrix **R** and translation matrix **T** for every camera. One of the possible ways to obtain those parameters is to use an algorithm proposed in [4] that makes use of images of a known chessboard pattern placed in front of the camera and is available as a part of OpenCV [3] library.

First we independently calibrate every camera. This operation is performed in order to obtain accurate intrinsic matrix and distortion coefficients for every camera. We use two-dimensional, high-contrast chessboard calibration pattern. The size of this pattern allows us to fill most frame area by the pattern image and improves distortion estimation, especially near the borders of image. The OpenCV corner detection algorithm is used to find corners coordinates on all captured images with sub-pixel precision. Corners coordinates, averaged over 100 frames of the same pattern position, were used to perform camera calibration. This was done using algorithm proposed in [4]. Calculated intrinsic matrices and distortion coefficients are saved and used in further calculations.

The next step is to find translation and rotation matrices. This operation is performed by capturing calibration pattern simultaneously by all cameras in the system. We use the same calibration pattern, capturing procedure and corners coordinates estimation method as in previously. Calculated coordinates and already known cameras intrinsic parameters are used to find extrinsic cameras parameters. The extrinsic parameters (**R**, **T**) are calculated with respect to chessboard pattern using the OpenCV library. Calculated translation **T** vectors and rotation **R** matrices are the base for next steps.

3.2 Coordinate systems transposition

After extrinsic parameters calculation we proceed to estimate relations between all cameras in a multi-view system. Translation vectors **T** and rotation

matrices $\mathbf{R}$ were calculated in every camera own coordinate system, so the important operation is to present these parameters into one coordinate system. In the first step of finding inter-camera relations, we use first camera (camera 0) coordinate system as a common base for the whole camera system. We calculate relative rotation ($\mathbf{R}_{rel}$) and relative translation ($\mathbf{T}_{rel}$) between camera 0 and other cameras:

$$\begin{aligned} \mathbf{R}_{rel}(0 \rightarrow i) &= \mathbf{R}(0) \cdot \mathbf{R}^{-1}(i), \\ \mathbf{T}_{rel}(0 \rightarrow i) &= \mathbf{R}_{rel}(0 \rightarrow i) \cdot \mathbf{T}(i) - \mathbf{T}(0). \end{aligned} \tag{2}$$

where $\mathbf{R}_{rel}(0 \rightarrow i)$ is rotation matrix between camera 0 and cam. i, $\mathbf{R}(i)$ is camera i rotation matrix, $\mathbf{T}_{rel}(0 \rightarrow i)$ is translation matrix between camera 0 and camera i, $\mathbf{T}(i)$ is cam. i translation matrix. Those calculated inter-camera relations are still related to camera 0 position and direction, so the next important operation is to find a coordinate system that does not promote any camera. The new coordinate system has to be independent of any single camera coordinate system. We find this system by using linear regression and try to estimate a line nearest to center points of all cameras. This line forms the x-axis of a new coordinate system. Linear regression is calculated separately for y(x) and z(x) relations between cameras center points coordinates. As a result of the regression analysis, we get a_y, b_y, a_z and b_z coefficients:

$$\begin{aligned} y(x) &= a_y \cdot x + b_y, \\ z(x) &= a_z \cdot x + b_z. \end{aligned} \tag{3}$$

After we estimate linear regression, we calculate translation and rotation that transform camera 0 coordinate system to the new system. The y- and z-intercepts (b_y, b_z) of linear regression are used to create translation correction ($\mathbf{T}_c$) vector, which corresponds to the origin points difference of the old and new system:

$$\mathbf{T}_c = \begin{bmatrix} 0 \\ -b_y \\ -b_z \end{bmatrix}. \tag{4}$$

where $\mathbf{T}_c$ is translation correction vector. Next step is to estimate rotation between camera 0 coordinate system and the new coordinate system. We use slopes of the lines (a_y, a_z) to calculate rotation correction matrix ($\mathbf{R}_c$). Angle of rotation around z axis (α_z) is calculated using a_y coefficient and rotation around y axis (α_y) is calculated using az coefficients:

$$\begin{aligned} \alpha_z &= \tan^{-1}(a_y), \\ \alpha_y &= \tan^{-1}(a_z). \end{aligned} \tag{5}$$

Previously obtained rotation angles are then converted into rotation matrices. The z-axis rotation matrix ($\mathbf{R}_z$) is calculated using α_z rotation angle, and y-axis matrix ($\mathbf{R}_y$) is calculated using α_y rotation angle (see Fig. 1):

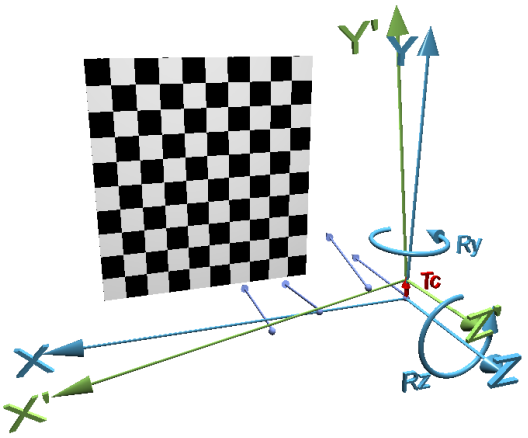

Fig. 1. Transformation between coordinate systems. X,Y,Z is camera 0 coordinate system. X',Y',Z' is independent coordinate system.

$$\mathbf{R}_y = \begin{bmatrix} \cos(\alpha_y) & 0 & \sin(\alpha_y) \\ 0 & 1 & 0 \\ -\sin(\alpha_y) & 0 & \cos(\alpha_y) \end{bmatrix}, \mathbf{R}_z = \begin{bmatrix} \cos(\alpha_z) & -\sin(\alpha_z) & 0 \\ \sin(\alpha_z) & \cos(\alpha_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6)$$

Rotation between camera 0 coordinate system and new coordinate system is calculated as a rotation correction matrix ($\mathbf{R}_c$) as a product of $\mathbf{R}_y$ and $\mathbf{R}_z$ matrices:

$$\mathbf{R}_c = \mathbf{R}_z \cdot \mathbf{R}_y. \quad (7)$$

Previously calculated relative translation vectors have to be converted to the new coordinate system. This allows us to find cameras coordinates in the new coordinate system:

$$\mathbf{T}_n(i) = \mathbf{R}_c \cdot [T_{rel}(0 \rightarrow i) + \mathbf{T}_c]. \quad (8)$$

where $\mathbf{T}_n(i)$ is camera i coordinate in new coordinate system, $T_{rel}(0 \rightarrow i)$ is translation matrix between camera 0 and camera i, R_c is rotation correction matrix, T_c is translation correction matrix, i is camera number.

The last estimated parameter is an average rotation matrix $\mathbf{R}_a$, calculated by averaging relative rotation matrices $R_{rel}(0 \rightarrow i)$ for all cameras. This parameter is used to find the difference between camera 0 rotation and average rotation of all cameras. Ra parameter will be used to eliminate camera 0 rotation influence in target camera parameters calculation.

3.3 Final camera parameters calculation

Previously calculated real cameras parameters and cameras relations are now used to find target cameras parameters. These parameters correspond to parameters of ideal multi-view system. Target intrinsic matrix $\mathbf{A}_t$ is calculated

by averaging all cameras intrinsic matrices. This forces all cameras to have the same focal length and the same position of the principal point. We also assume that target cameras are distortion free, so set the distortion coefficients in target system to zero. Target translation and rotation matrices were calculated using known translation and rotation matrices and correction matrices ($\mathbf{R}_c$, $\mathbf{T}_c$, and $\mathbf{R}_a$). Target parameters for camera 0 are calculated as follows:

$$\begin{aligned} \mathbf{T}_t(0) &= \mathbf{R}_c \cdot [\mathbf{T}(0) + \mathbf{T}_c] \,, \\ \mathbf{R}_t(0) &= \mathbf{R}_a^{-1} \cdot [\mathbf{R}_c \cdot \mathbf{R}(0)] \,. \end{aligned} \tag{9}$$

where $\mathbf{T}_t(0)$ is target translation matrix for first camera, $\mathbf{T}(0)$ is first camera translation matrix, $\mathbf{R}_t(0)$ is target rotation matrix for first camera, $\mathbf{R}(0)$ is first camera rotation matrix, $\mathbf{T}_c$ is translation correction matrix, $\mathbf{R}_c$ is rotation correction matrix, $\mathbf{R}_a$ is average rotation matrix. Target rotation matrices for all cameras are the same as for the camera 0. This makes cameras optical axes parallel. Target translation matrices for other cameras are calculated by separating their new optical centres positions along x axis by a defined step, calculated as an average distance between neighbouring cameras (d_a). This operation was performed by modifying x-axis value in target translation matrix:

$$\mathbf{T}_t(i) = \mathbf{T}_t(0) + \begin{bmatrix} d_a \cdot i \\ 0 \\ 0 \end{bmatrix} , \tag{10}$$

where $\mathbf{T}_t(i)$ is target translation matrix for i camera, $\mathbf{T}_t(0)$ is target translation matrix for first camera, d_a is average distance between cameras, i is camera number.

3.4 Rectifying transform calculation

Previously calculated ideal multi-view system cameras parameters are now used to estimate rectification parameters. We select four points from corners of calibration pattern, and correct their coordinates to remove distortion influences. We also use the target cameras parameters and the equation (1) to calculate projection of points that correspond to the corners of calibration pattern. This allows us to obtain corners coordinates in ideal multi-view system. Having two groups of four points (real and target), we are able to find a perspective transformation $\mathbf{P}_t$ that performs mapping between these points. Example of two groups of points, real and target, are shown in Fig. 2. The final part of the described solution is image transformation. First, the distortions are removed from the image with the use of known intrinsic matrix $\mathbf{A}$ and distortion vector $\mathbf{D}$. Then, a perspective transformation $\mathbf{P}_t$ is applied to all images.

3.5 Rectification results

Quality of rectification can by judged by calculating depth maps using rectified images. Depth maps calculation requires that all scene points can be found in

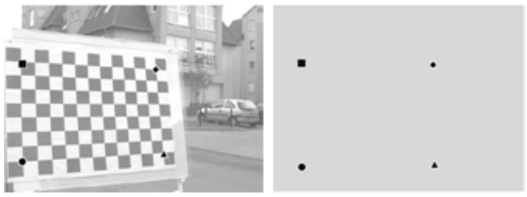

Fig. 2. Image of four real corners of calibration pattern (left), and image of four target corners corresponding to real ones (right).

Fig. 3. Image of the scene (top left), depth map estimated for test sequence without rectification (top right) where significant artefacts can be seen and after rectification with proposed algorithm (bottom) with very limited amount of artefacts.

the same line of images from all cameras. Resulting depth map can be compared with the one calculated with original images. Test result of a depth map estimation before and after performing rectification using algorithm proposed can be seen on Fig. 3.

4 Conclusions

The algorithm is proved to be an effective way of performing rectification in multi-camera system. Its main application is to produce images that satisfy the requirements of depth estimation algorithms and simplified view synthesis algorithms. The algorithm described here was used to prepare 4 test sequences contributed to MPEG works [15]. These sequences are regarded as very good and are used by many research groups in the world for experiments in 3D video field.

The use of proposed algorithm increases overall quality of generated depth map compared to the case with not rectified images. It is an universal and straightforward solution for parallel multi-camera setups.

Acknowledgment

This work was supported by the public funds as a research project.

References

1. Description of Exploration Experiments in 3D Video Coding (2010), ISO/IEC JTC1/SC29/WG11, MPEG 2010/W11274, Dresden, Germany
2. Fusiello A., Trucco E., Verri A. (2000) A compact algorithm for rectification of stereo pairs. Machine Vision and Applications vol. 12, no. 1, Springer-Verlag
3. OpenCV homepage http://sourceforge.net/projects/opencvlibrary/
4. Zhang Z. (2000) A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11:1330–1334
5. Cyganek B., Siebert J. P. (2009) An Introduction to 3D Computer Vision Techniques and Algorithms. John Wiley and Sons
6. Yun-Suk Kang, Yo-Sung Ho (2008) Geometrical Compensation for Multi-view Video in Multiple Camera Array. 50th International Symposium ELMAR-2008, 10-12 September 2008, Zadar, Croatia
7. Yun-Suk Kang, Cheon Lee, Yo-Sung Ho (2008) An Efficient Rectification Algorithm for Multi-view Images in Parallel Camera Array. 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 28-30 May 2008, Istanbul
8. Zhang Z. (1999) Flexible Camera Calibration By Viewing a Plane From Unknown Orientations. The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 1:666–673
9. Tsai R. (1987) A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE Journal of Robotics and Automation, vol. 3, no. 4:323–344
10. Hartley R. (1999) Theory and Practice of Projective Rectification. International Journal of Computer Vision, vol. 35, no. 2:115–127
11. Loop C., Zhang Z. (1999) Computing rectifying homographies for stereo vision. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1999, vol. 1:125–131
12. Nowakowski A., Tomaszewski M., Skarbek W. (2006) Modelowanie Nieliniowoci Optycznych w Obrazie Cyfrowym Technik Iteracji Punktw Centralnych. V Sympozjum Naukowe "Techniki Przetwarzania Obrazu", Serock (in Polish)
13. Domański M., Klimaszewski K., Konieczny J., Kurc M., Łuczak A., Stankiewicz O., Wegner K.(2009) An Experimental Free-view Television System. Image Processing and Communications Challenges, R. Choraś, A. Zabłudowski (editors), pp. 169–176, Academy Publishing House EXIT, Warsaw
14. Domański M., Klimaszewski K., Konieczny J., Kurc M., Łuczak A., Stankiewicz O., Wegner K.(2009) An Experimental Free-view Television System. Image Processing and Communications Challenges, R. Choraś, A. Zabłudowski (editors), pp. 169–176, Academy Publishing House EXIT, Warsaw
15. Domański M., Grajek T., Klimaszewski K., Kurc M., Stankiewicz O., Stankowski J., Wegner K., (2009) Poznan Multiview Video Test Sequences and Camera Parameters. ISO/IEC JTC1/SC29/WG11, MPEG 2009/M17050, Xian, China