Fast Generation of Cylindrical Panoramic Views From Free-Hand Video Sequences

Kai Ide, Matthias Kunter, Thomas Sikora
Communication Systems Group
Technische Universität Berlin
Einsteinufer 17
10587 Berlin, Germany
{ide, kunter, sikora}@nue.tu-berlin.de

ABSTRACT

We report on a fast algorithm for the generation of cylindrical panoramic views from hand-held video sequences. Due to its high processing speed the algorithm is suited for hardware implementation into next generation video- and photo cameras. This enables the user to easily create immersive views from simple pan shots of variable quality. The individual processing steps within the algorithm are described in detail. Final results of the video to panorama conversion process along with an an outlook on how to further improve the method when implemented in consumer grade video- and photo cameras are given at the end of this paper.

1. INTRODUCTION

When attempting to capture scenes with an extremely wide field of view, such as inside stadiums or on top of a mountain, one encounters the task of creating panoramic views. A standard image captures a field of view of roughly 60°, the use of extreme wide angle lenses can give fields of view of up to 180°. Our algorithm however can easily yield full 360° panoramas, thereby creating an immersive visual of the captured scenery. The advances in processing capabilities of mobile devices now allow the computation of panoramic views within the device. In general, the resulting image quality is of course limited by the captured video resolution, and thus, unable to compete with offline computation of panoramic views made from still images. However, using video sequences enables the use of feature tracking for finding point correspondences of adjacent frames. This approach proves to be much faster than conventional full search methods that have to be applied for finding point correspondences in arbitrary images. We tested the algorithm with a dataset containing over 30 different video sequences comprised of both pan shots and free hand videos. Video resolution ranges from standard VGA to 720p. It has to be mentioned that the algorithm runs fully automatically requiring no user defined parameters for the computation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IMMERSCOM '09 Berkeley, California USA
Copyright 2009 ACM ICST 978-963-9799-39-4 ...$5.00.

1.1 System Overview

The algorithm is comprised of five functional blocks, which are illustrated below in fig. 1. These block are described in more detail later in this paper.

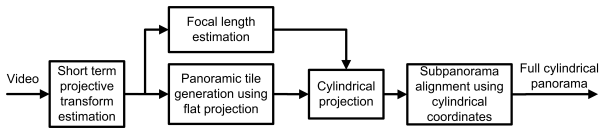


Figure 1: Schematic block diagram illustrating the functional overview of the presented system.

1.2 Related Work

A number of previous publications deal with the general problem of stitching together several still images in order to form a panoramic view [12], [16], [13], [15]. There exist however only few publications that focus on the computation of panoramic views directly from videos [14]. An impressive example is given by [2] in which the dynamic nature of video is integrated into the final panorama. While the use of dynamic video elements within a panoramic view further enhances the immersive viewing experience, the process requires manual user input and is therefore not suitable for the fully automatic implementation into consumer grade cameras.

2. THE WORKING PRINCIPLE

As described above the algorithm aims at combining the individual frames of a given video sequence in a way that results in a high quality panoramic view. In order to achieve this in minimal time the following steps are taken in the order given here. The implementation of this work was done in C++, extensively utilizing the popular OpenCV [4] library.

2.1 Feature Tracking

In order to find a hopefully large number of point correspondences between two neighboring frames the Kanade-Lucas-Tomasi (KLT) feature tracker (FT) [3] is utilized. Given a small interframe displacement with respect to the optical flow of adjacent frames the FT is able to produce satisfying results which predicate the horizontal and vertical displacement of a small 5×5 pixel window I around a given feature point, thus minimizing the sum of squared

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6212
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6212

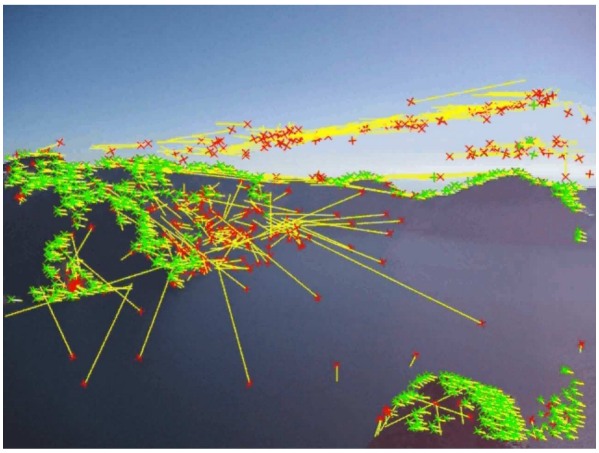


Figure 2: The above image shows detected point correspondences within one of two adjacent frames. Correct matches are marked in green, whereas outliers are shown in red. The yellow trajectory indicates the optical flow from the previous frame to the current one.

differences:

$$\epsilon = \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x}) \right]^2 , \tag{1}$$

where $\mathbf{W}$ denotes a warping function, $\mathbf{x}$ are the pixels in homogenic coordinates, $\mathbf{p}$ is the set of parameters constituting the actual warp, and T is a template window within the next frame. The iterative search algorithm is strongly related to a gradient descent search and well described in [3].

The answer to which features to select is given by Shi and Tomasi [11]. They derive the feature selection process from the actual task of feature tracking and, in a nutshell, conclude that features in forms of corners and single points make good features to track. Both of these functions are built into the OpenCV library and are therefore readily available without an otherwise necessary need to re-implement.

The tracking algorithm, however, sometimes gives erroneous results, namely wrong point correspondences or outliers that have been marked in red in fig. 2. This is fatal for the algorithm's next processing step and has to be compensated for by removing outliers from the feature set with RANSAC[1] [6]. Implementation of the latter corresponds with the description in [7].

2.2 Creating Cylindrical Subpanoramas

With the optical flow between the two frames a homography matrix $\mathbf{H}$ can be computed by applying a singular value decomposition to the set of correct point correspondences as described in [7] and [10]. $\mathbf{H}$ is then applied to warp each pixel $\tilde{\mathbf{x}}$ of successive frames i onto the initial frame 1.

$$\tilde{\mathbf{x}}_1 = \mathbf{H}_{1i}\tilde{\mathbf{x}}_i . \tag{2}$$

[1]RANSAC: Random Sample Consensus

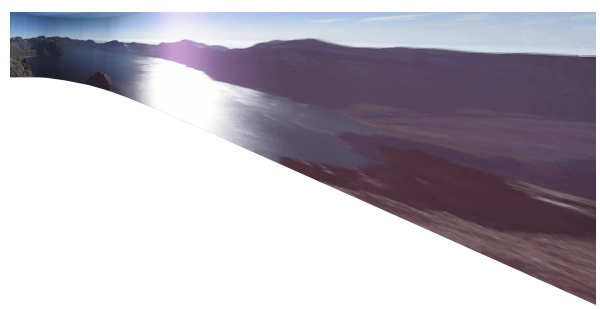
2.2.1 Perspective Projection

Up to this point the algorithm is able to calculate simple planar panoramic views which can have a field of view no more than 180 °. As mentioned above outliers in the calculated optical flow can have a dramatic effect on the image quality of the panoramic views due to malformed homography matrices. This effect is illustrated in fig. 3. Keeping in mind that not all tracked features within the inliner set are perfect, further attention must be payed when warping multiple frames onto a single planar panoramic view. Each succesive homography matrix is partially calculated from previous homography matrices such that the N-th matrix warping onto the first frame is given by:

$$\mathbf{H}_{1,N} = \mathbf{H}_{1,N-1} \cdot \mathbf{H}_{N-1,N} . \tag{3}$$

Since it is, due to imperfect point correspondences, possible to have error propagation from matrix to matrix we introduce keyframe panoramas after a certain horizontal image size, relative to the video frame size we are creating the panoramic view from, has been reached. This size threshold limits the negative effects on error propagation within the planar panoramic views and also prevents singularities at the image boundaries, which occur reaching a camera rotation of ±90 ° in the horizontal direction.

(a) without RANSAC



(b) with RANSAC

Figure 3: The figure above shows two partial panoramic views of the test sequence Manchuria. Image (a) was computed neglecting the effect of outliers. Image (b) shows the partial panoramic view after outliers have been removed from the correspondence set.

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6212
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6212

2.2.2 Focal Length Estimation

The problem of preventing singularities in size when generating planar panoramic views will be addressed in the cylindrical projection stage of the algorithm. Prior to computing the cylindrical projection it is necessary to compute the focal length of the video sequence in units of pixels. The focal length will represent the cylinder's radius and is therefore a crucial parameter for the entire process. Two approaches for solving this task apply: First, when handling video sequences with an arbitrary camera path standard camera calibration or tracking techniques can be utilized. Second, when dealing with a pan shot, i.e. a video where the camera center is static through the entire sequence, we can use the pre-computed homography matrices to solve for the focal length as described in [8] and [16]. A homography between two frames can be separated into a linear combination of the intrinsic camera matrices of both views $\mathbf{K}_i$ and $\mathbf{K}_j$ and a rotation matrix $\mathbf{R}$. When assuming a constant focal length, which is the case in the examples presented in this paper, we can write this as:

$$\mathbf{K}_i = \mathbf{K}_j = \mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

Except for a scaling factor the corresponding homography matrix $\mathbf{H}_{ij}$ can therefore be written as:

$$\mathbf{H}_{ij} = \mathbf{K}\mathbf{R}_{ij}\mathbf{K}^{-1} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{R}_{ij} \cdot \begin{bmatrix} \frac{1}{f} & 0 & 0 \\ 0 & \frac{1}{f} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

Considering the matrix's individual parameters and simplifying our notation to $\mathbf{R}_{ij} = \mathbf{R}$, with

$$\mathbf{R} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \tag{6}$$

and $\mathbf{H}_{ij} = \mathbf{H}$, we can write the homography as:

$$\mathbf{H} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \sim \begin{bmatrix} r_{00} & r_{01} & f \cdot r_{02} \\ r_{10} & r_{11} & f \cdot r_{12} \\ r_{20}/f & r_{21}/f & r_{22} \end{bmatrix}. \tag{7}$$

Since the rows and columns of $\mathbf{R}$ have to be orthogonal to one another, we can set up the following system of equations:

$$\begin{aligned} 0 &= h_{00}h_{10} + h_{01}h_{11} + h_{02}h_{12}/f^2 \\ &= h_{00}h_{20}f + h_{01}h_{21}f + h_{02}h_{22}/f \\ &= h_{10}h_{20}f + h_{11}h_{21}f + h_{12}h_{22}/f \\ &= h_{00}h_{01} + h_{10}h_{11} + h_{20}h_{21}f^2 \\ &= h_{00}h_{02}/f + h_{10}h_{12}/f + h_{20}h_{22}f \\ &= h_{01}h_{02}/f + h_{11}h_{12}/f + h_{21}h_{22}f \end{aligned}. \tag{8}$$

In addition to this the first two columns and the first two rows of $\mathbf{R}$ have identical norms, which results in the second system of equations:

$$\begin{aligned} h_{00}^2 + h_{01}^2 + h_{02}^2/f^2 &= h_{10}^2 + h_{11}^2 + h_{12}^2/f^2 \\ &= h_{20}^2 f^2 + h_{21}^2 f^2 + h_{22}^2 \\ h_{00}^2 + h_{10}^2 + h_{20}^2/f^2 &= h_{01}^2 + h_{11}^2 + h_{21}^2 f^2 \\ &= h_{02}^2/f^2 + h_{12}^2/f^2 + h_{22}^2 \end{aligned} \tag{9}$$

Solving both systems for the focal length f yields the cylinder's radius in units of pixels, which will be needed in the following processing step.

2.2.3 Cylindrical Projection

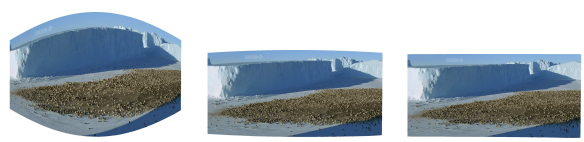
The various planar panoramic views can now be projected onto a cylinder, where the transformation from Cartesian image coordinates $\mathbf{x} = [x, y]^T$ to cylindrical image coordinates $\mathbf{x}_{zyl} = [\theta, h]^T$ is given as follows:

First the planar image pixels have to be mapped into the 3D space in front of a cylinder:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \frac{1}{\sqrt{(x - x_d)^2 + f^2}} \begin{bmatrix} x - x_d \\ y - y_d \\ f \end{bmatrix}, \tag{10}$$

where x_d is half of the video frame width and y_d is half of the video frame height in pixels. The focal length is again given by f. The cylinder coordinates θ and h can then be calculated with:

$$\begin{bmatrix} \theta \\ h \end{bmatrix} = \begin{bmatrix} \arctan\left(\hat{x}/\hat{z}\right) \\ \hat{y} \end{bmatrix} \tag{11}$$



(a) $f = 500$ (b) $f = 2000$ (c) $f = 8000$

Figure 4: Image distortion with respect to the focal length f, increased by a factor of four in each step. As can be observed a value of $f = 500$ strongly distorts the given image and renders it unuseful for generating a full $360\,^\circ$ panorama.

As a third step we re-project the cylinder's surface onto a plane with:

$$\begin{bmatrix} x_{zyl} \\ y_{zyl} \end{bmatrix} = \begin{bmatrix} f \cdot \theta + x_d \\ f \cdot h + y_d \end{bmatrix}. \tag{12}$$

Fig. 4 shows the resulting image deformations with respect to the focal length. A transformation with $f = \infty$ would leave the input image, a partial planar panoramic view, unchanged.

2.3 Full cylindrical panorama generation

Once multiple cylindrical panoramic views have been computed the algorithm will attempt to stitch all of them together to form a preliminary cylindrical panoramic view. In theory, having computed the correct focal length of the camera with which the underlying video sequence had been

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6212
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6212

recorded, the individual cylindrical panoramic views will easily align. This is because image alignment on a cylindrical surface becomes a pure translative problem [12]. A rotation of the camera is a translation of the cylindrical panorama. *If* the focal length has been computed correctly. Since this can not be always assumed the algorithm has to compensate in a way that preserves high quality partial panoramic views that have been mapped onto a cylindrical surface with correct focal length and at the same time minimizing computational cost and maximizing subjective image quality when partial panoramic views generated from substantially wrong focal lengths are stitched together.

2.3.1 Alignment of partial panoramic views

A rather large 200×150 pixel template is taken from a given partial cylindrical panorama, which we from now on refer to as tile, and compared to a window of the same size within the subsequent tile. A similar approach is for instance taken in [18]. To get a rough initial estimate for the translation vector between the two windows, we remember the position of the center of the last frame n that was added to the preceding tile. In the current tile we already know where the center of the next frame $n+1$ will be, since it forms the seed image for the panoramic view to grow on. The translation between the two points is our initial guess which is then forwarded to a gradient descent search that tries to iteratively match the template with the current tile. As mentioned above, given a perfect focal length, this task is a solely translative problem and will converge quickly. If however cylindrical distortions, due to an erroneously computed focal length, come into play, alignment will *not* necessarily converge assuming a pure translation of the search window. For this reason an iteratively updated window warp matrix $\mathbf{W}$ has to contain translative components t_x and t_y in the x- and y-direction, respectively as well as a rotation component w_z. $\mathbf{W}$ can then be written as:

$$\mathbf{W} = \begin{pmatrix} 1 & w_z & t_x \\ -w_z & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{13}$$

.

In case the gradient descent search gets stuck in a local minimum, thereby making it in some cases impossible to converge, we limit the maximum number of iterations to 1000. The preliminary results after this processing step can be observed in fig. 6.

2.3.2 Blending with Multi-Resolution Splines

The basic principle behind this technique is to subdivide the images which are to be blended together into several subbands with respect to the image's frequency components. Low frequencies, such as the mean gray value, will receive a wide transition zone throughout both images, so that they can be blended smoothly. High frequencies, such as small rocks, along the principle transition line will receive a narrow transition zone, thereby preserving detail from both images. The implementation presented in this paper is largely based on [5], where image blending with multi-resolution splines was first described. In order to find a non necessarily vertical transition line, we apply a technique from the open source project *enblend* [1].

First the required target space for the resulting image after the blending process is calculated. Then each input tile

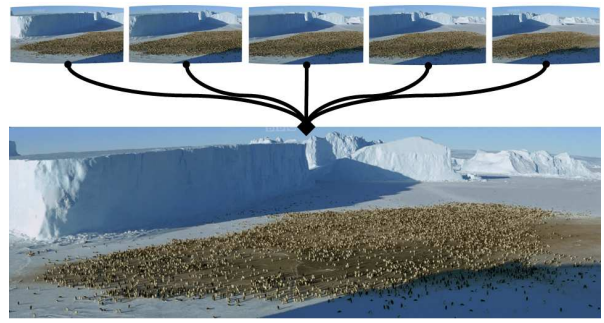


Figure 5: Combination of subpanoramas to form the resulting panorama showing a group of penguins in Antarctica. This sequence is taken from the documentary BBC: Planet Earth – Ice Worlds

is warped into this target space with $\mathbf{W}$. We refer to the target space as overlay. In order to create a pyramid containing non-overlapping subbands from the complete image spectrum, we first make sure the overlay image precondition, where l is the overlay's length in x- and y-direction holds:

$$\left(\frac{l}{2^{N-1}}\right) \bmod 2 = 0 \quad \forall x \in [1, \ldots, M]\,. \tag{14}$$

The total number of subbands is given by $N-1$, M describes the maximum input size in pixels. The images will both be put into the target overlay image, where zero color values are appended around them for the next processing step. This step is the creation of subbands. This is done with image pyramids, which are for instance described in [17]. Four Gaussian pyramids GA, GB, GR and GRN are created by subsampling the input images by a factor of two and applying the convolution kernel $\mathbf{\Gamma}$ in eq. 16, with

$$\mathbf{\Gamma} = \begin{bmatrix} 0.0025 & 0.0125 & 0.0200 & 0.0125 & 0.0025 \\ 0.0125 & 0.0625 & 0.1000 & 0.0625 & 0.0125 \\ 0.0200 & 0.1000 & 0.1600 & 0.1000 & 0.0200 \\ 0.0125 & 0.0625 & 0.1000 & 0.0625 & 0.0125 \\ 0.0025 & 0.0125 & 0.0200 & 0.0125 & 0.0025 \end{bmatrix}. \tag{15}$$

The Gaussian pyramid of the first input image is given by GA, the Gaussian pyramid of the second input image is given by GB. GR and GRN represent the Gaussian pyramids of the non-inverted and inverted binary blending mask, respectively. Each layer of the pyramids is created according to eq. 16, as described above, by subsampling and convolving the layer with $\mathbf{\Gamma}$. Formally this can be described as follows:

$$G_l(i,j) = \sum \sum_{m,n=1}^{5} \mathbf{\Gamma}(m,n) G_{l-1}(2i+m, 2j+n), \tag{16}$$

where i and j as well as m and n denote pixel coordinates. The pyramidal layer index is given by l, where $l = 0$ represents the original image. Bandpass images are computed by subtracting layer l from layer $l-1$. Since higher layers contain a lower spatial resolution, they have to be expanded by interpolation to match the preceding layers resolution. This step can be described as:

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6212
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6212

(a) before optimization

(b) after optimization

Figure 6: Two examples from the 180° Bonn-Rhein testsequence. (a) shows the panorama before the multiresolution spline blending step, (b) shows the final panorama. We point out the good overall result despite dynamic objects within the scene.

$$G_{l,k} = 4 \sum \sum_{m,n=-2}^{2} G_{l,k-1} \left(\frac{2i+m}{2}, \frac{2j+n}{2} \right). \quad (17)$$

The pyramid of bandpass filtered images is then formed by

$$L_l = G_l - G_{l+1,l}, \quad (18)$$

where the highest layer is equivalent to the corresponding Gaussian layer $L_N = G_N$. The subtraction above is related to Laplace operators, which are commonly used in image processing. The resulting pyramid L is therefore called a Laplacian pyramid. The final image is reconstructed by simply adding the individual layers of L together.

$$G_0 = \sum_{l=0}^{N} L_{l,l} \quad (19)$$

Fig. 6 compares results of simply stitch overlay tiles together versus the multi-resolution splining technique. As can be observed, the individual video frame boundaries disappear in the latter panorama and the mean brightness of for instance the sky is homogeneous throughout the entire image.

3. RESULTS

Examples of the algorithm's output are given in fig. 5 and 6. The first sequence yields a high quality panoramic view which can be mainly contributed to two aspects. First, the input video has a relatively high resolution, namely HD 720p. Second, the sequence is made up from a smooth pan shot filmed using a tripod. This means that the camera center remains static throughout the entire sequence and therefore makes it perfectly suitable for video to panorama conversion. The second example shows the Rhine river running through the city of Bonn, Germany. Although filmed with a handheld camera not mounted onto a tripod and despite the fact that the boat and the waves on the river itself form dynamic objects, which are critical in the feature tracking process, the result is satisfying. We notice however some artifacts when closely observing the boat and the mountain range above it. This effect occurs due to the fact that the boat has moved between two tiles used during tile stitching.

Figure 7: The image above shows a cylindrical panoramic view of the volcanic lake at Changbai shan, China.

Another example can be seen in fig. 7. The camera used here is a Canon Ixus 50, which gives a low quality video in 640×480 pixel resolution at 30 frames per second. The camera center is not static throughout the sequence which is problematic when attempting to map the video onto a cylinder's surface. However, due to blending with multi-resolution splines the subjective panorama quality is satisfying. Distortions at the right hand side of the image are caused by imperfect computation of the video sequence's focal length, which is a problem that could be neglected if the algorithm received this information directly from a given camera. The system we used for creating the panoramic views in this paper is an off the shelf P4 3 GHz Windows XP computer with 1.5 GByte of RAM. The processing time for each panorama was roughly in the 2 minute range for sequence comprised of 360 frames in average. The algorithms computational cost is linear with $O(n)$, but the gradient descent tile stitching processing time depends on the overall video quality. Utilizing the well know 300 frame long *Stefan* video sequence at a resolution of 352×240 non-cylindrical panoramic views are synthesized in [9]. Albeit these sprites are close to perfect in a sense that makes them suitable for sprite-based video coding and, for instance, segmentation of dynamic objects, the processing time required to compute them lies around 10 hours. When comparing this number with the 2 minutes it takes to create cylindrical panoramic views with our algorithm it becomes evident why we refer to our system as *fast*.

4. LIMITATIONS OF THE ALGORITHM

Despite of good overall results the algorithm presented here has some limitations. Video sequences with dominant dynamic foreground objects are problematic during the feature tracking process and should best be avoided. Also, despite the algorithm being able to handle translations of the camera center, optimal results can only be obtained when the camera is rotated as steadily as possible. It is, for instance, not possible to create a panoramic view, by translating the camera horizontally, due to physical limitations of optical geometry when very different perspectives of the same basic scenery are introduced into the process. An example illustrating this problem is given in fig. 8.

5. SUMMARY

We have successfully created a solution for the automatic conversion of suitable video sequences into subjectively high

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6212
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6212

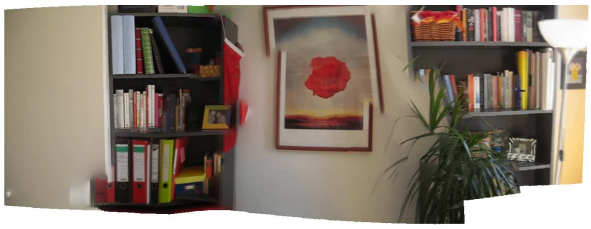


Figure 8: The distorted panoramic view above illustrates the limitations of the algorithm. It is impossible to align individual video frames when given a pure horizontal translation of the camera.

quality cylindrical panoramic views. The algorithm runs fast and on standard office computers. An intuitive graphical user interface (GUI), which is shown in fig. 9 enables easy use of the software we call **Vi2Pa**, making it suitable for offline creation of panoramic views from videos filmed during i.e. vacations. We are currently investigating possible applications of the algorithm with respect to sprite based video coding and hope to present some achievements in this field in the near future. Implementation of the algorithm into next generation camcorders or photo cameras would allow for easy creation of high resolution panoramic views for end customers.

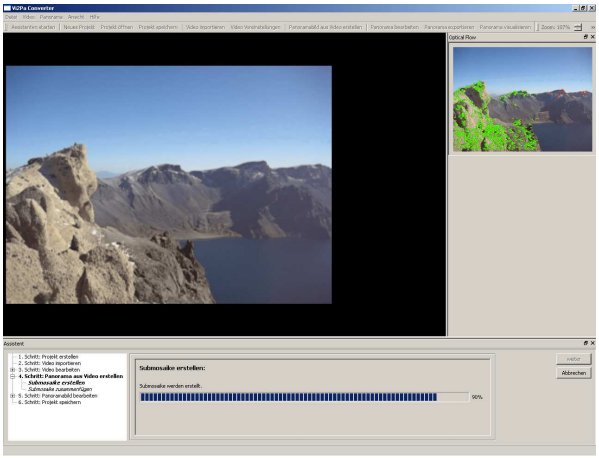


Figure 9: Screenshot showing the GUI of the algorithm in the form of a stand-alone software solution called Vi2Pa.

6. ACKNOWLEDGMENTS

This work has been supported by the Integrated Graduate Program on Human-Centric Communication at Technische Universität Berlin. We would like to thank Hannah Rehders for providing some of the sequences filmed in the People's Republic of China. We also are grateful for the development of the GUI, which was done by Moritz Wendt.

7. REFERENCES

[1] http://enblend.sourceforge.net/.

[2] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic video textures. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 821–827, New York, NY, USA, 2005. ACM.

[3] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2002.

[4] G. Bradski. The OpenCV library. 25(11):120, 122–125, Nov. 2000.

[5] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. volume 2, pages 217–236, New York, NY, USA, 1983. ACM.

[6] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. volume 24, pages 381–395, New York, NY, USA, June 1981. ACM Press.

[7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[8] M. Kunter. *Advances in Sprite-based Video Coding Towards Universal Usability*. Dissertation, Technische Universität Berlin, Germany, 2007.

[9] 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, VCIP, IS&T/SPIE's Electronic Imaging 2007*, San José, CA, USA, Jan. 2007. IS&T/SPIE.

[10] O. Schreer. *Stereoanalyse und Bildsynthese*. Springer, 2005.

[11] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[12] H.-Y. Shum and R. Szeliski. Panoramic image mosaics. Technical Report MSR-TR-97-23, Microsoft Research, Redmond, WA, January 1997.

[13] A. Smolic and T. Wiegand. High-resolution video mosaicing. In *ICIP (3)*, pages 872–875, 2001.

[14] R. Szeliski. Video mosaics for virtual environments. volume 16, pages 22–30, Los Alamitos, CA, USA, 1996. IEEE Computer Society.

[15] R. Szeliski. Image alignment and stitching: a tutorial. volume 2, pages 1–104, Hanover, MA, USA, 2006. Now Publishers Inc.

[16] 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*, pages 251–258, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[17] S. L. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, 4(2):104–119, June 1975.

[18] Z. Xiao-chun, Z. Xin-bo, and F. Yan. An efficient medical image registration algorithm based on gradient descent. *Complex Medical Engineering, 2007. CME 2007. IEEE/ICME International Conference on*, pages 636–639, May 2007.

Digital Object Identifier: 10.4108/ICST.IMMERSCOM2009.6212
http://dx.doi.org/10.4108/ICST.IMMERSCOM2009.6212