Real-Time Stereo Mosaicing using Feature Tracking

Marc Vivet
Computer Science Department
Universitat Autònoma de Barcelona
Barcelona, Spain

Shmuel Peleg
School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel

Xavier Binefa
ITT Deptartment
Universitat Pompeu Fabra
Barcelona, Spain

***Abstract*—Real-time creation of video mosaics needs fast and accurate motion computation. While most mosaicing methods can use 2D image motion, the creation of multi view stereo mosaics needs more accurate 3D motion computation. Fast and accurate computation of 3D motion is challenging in the case of unstabilized cameras moving in 3D scenes, which is always the case when stereo mosaics are used. Efficient blending of the mosaic strip is also essential.**

Most cases of stereo mosaicing satisfy the assumption of limited camera motion, with no forward motion and no change in internal parameters. Under these assumptions uniform sideways motion creates straight epipolar lines. When the 3D motion is computed correctly, images can be aligned in space-time volume to give straight epipolar lines, a method which is depth invariant.

We propose to align the video sequence in a space-time volume based on efficient feature tracking, and in this paper we used Kernel Tracking. Computation is fast as the motion in computed only for a few regions of the image, yet giving accurate 3D motion. This computation is faster and more accurate than the previously used direct approach.

We also present "Barcode Blending", a new approach for using pyramid blending in video mosaics, which is very efficient. Barcode Blending overcomes the complexity of building pyramids for multiple narrow strips, combining all strips in a single blending step.

The entire stereo mosaicing process is highly efficient in computation and in memory, and can be performed on mobile devices.

***Keywords*-Video Mosaics; Stereo Mosaics; Stereo Panorama; Parallax; Pyramid Blending;**

I. Introduction

Stereo mosaics (or stereo panoramas) from a single video are created by stitching the video frames into at least two mosaic images that can create a 3D effect for the viewer. The 3D effect can be generated only when the captured scene has depth differences, and when the motion includes translation to create motion parallax. Stereo mosaics from a single camera were first introduced in [1], [2], and were generated in two steps. The first step computed image translation and rotation between video frames, and the second step was mosaic stitching. Left eye mosaics were stitched from strips taken from the right side of the frames, and right-eye mosaics were generated from strips taken from the left side of the video frames. Generation of any number of views has been described in [3].

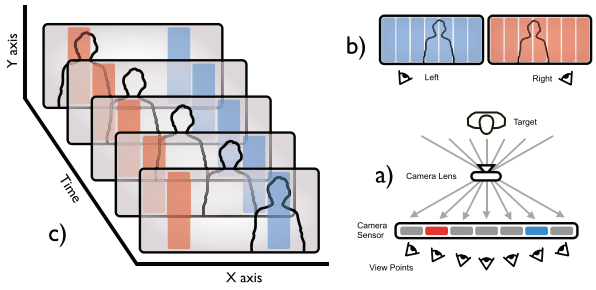

Figure 1. (a) Every region on the image sensor corresponds to a different viewing direction. (b) Stereo mosaics are generated by creating the left-eye mosaic from strips taken from the right side of the video frames, and the right-eye mosaic is created from strips taken from the left side of the video frames. (c) Stacking the video frames in an $x-y-t$ space time volume. Creating a stereo mosaic can be seen as "slicing" the space-time volume by two $y-t$ planes. One on the left side and one on the right side.

Computing a global image motion, e.g. a global rotation and translation between frames, results in distorted stereo mosaics, as relevant scenes have 3D parallax that can not be described by global parametric motion. The accuracy of motion computation can be increased when observing that the motion used for stereo mosaicing is limited: there is no forward motion, and there is no change in internal parameters. This observation was used in [4], [3] to propose depth invariant image alignment of the video sequence in an $x-y-t$ space time volume, see Fig. 1. The concept of Time Warping has been introduced based on the EPI (epipolar) planes described in [5]. Paper [5] observed that when the camera motion is constant the lines generated by trajectory of features in the EPI planes (epipolar lines) are straight lines. Time Warping was a resampling of the t axis in order to make the epipolar lines straight, simulating a constant velocity. The method proposed in [4], [3] uses Lucas Kanade [6] over the entire image, making it computationally expensive.

As in [4], [3] we propose to align the video sequence using the $x-y-t$ space time volume, but using efficient feature tracking instead of using Lucas Kanade. In [7] Kernel Tracking has been demonstrated as a very efficient and accurate technique for moasicing purposes. With our novel approach only a few regions of each frame are needed in

978-0-7695-4589-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ISM.2011.102

order to obtain the epipolar lines, resulting in Real-Time 3D alignment. All the alignment parameters can be computed in a single step using a newton style approximation, taking advantage of the character of the Kernel Tracking.

Another contribution of this paper is an introduction of a new pyramid blending scheme [8] for video mosaicing, overcoming the difficulty in blending together multiple narrow strips. The new blending, "Barcode Blending", blends all strips in a single blending step.

The remainder of the paper is structured as follows: Sec. II provides a detailed description of our image alignment approach, and Sec. III presents mosaic generation and Barcode Blending. Sec. IV describes the experiments conducted. Final remarks and future work can be found in Sec V.

II. 3D Motion Computation

A video sequence is a sequence of frames $\{f_i\}_{i=1:N}$. The first task in any video mosaicing is to compute the alignment between the frames, alignment that will allow stitching the frames into a coherent mosaic image. In stereo mosaicing, creating multiple views of a 3D scene, 2D image alignment is not appropriate. Computation of the 3D camera motion is needed for creation of undistorted stereo mosaics. Following [4], [3] we align frames in the space-time volume so that EPI lines [5] become straight.

The alignment parameters of each frame f_i are $\Omega_i = \{t_i, y_i, \alpha_i\}$, placing the frame f_i in the $x-y-t$ space time volume such that the epipolar lines become straight. Parameters $\{t_i, y_i, \alpha_i\}$ corresponds to the time, the vertical displacement, and rotation angle of frame f_i. We refer to this alignment as Time Warping motion model. We will first make a brief review to Kernel Tracking. Afterwards, our Time Warping approach will be described in detail. A summary of our algorithm will than be provided.

A. Kernel Tracking

Kernel Tracking (KT) and Multiple Kernel Tracking (MKT) with Sum of the Squared Differences (SSD) were presented on [9], and we adopt a similar notation. Kernel-based objective functions have been demonstrated as an effective means of tracking in video sequences. In these functions the target model q is a spatially weighted histogram as defined in a formal way in (Eq. 1).

$$q = U^t K(c), \tag{1}$$

where $K(c)$ is a the kernel weight function giving a lower weight to pixels further away from $c = (x, y)$, the center of the target region, and U is the sifting matrix. $K(c)$ is a vector of n elements (n is the number of pixels of the target region), and U is an n by m matrix, where m is the number of bins in the histogram. U is defined by $U = [u_1, u_2, ..., u_m]$ where, $u_i = u_i(t) = \delta(b(x_i, t), i)$, x_i is the intensity value of the pixel at position i, and δ is the Kronecker delta function.

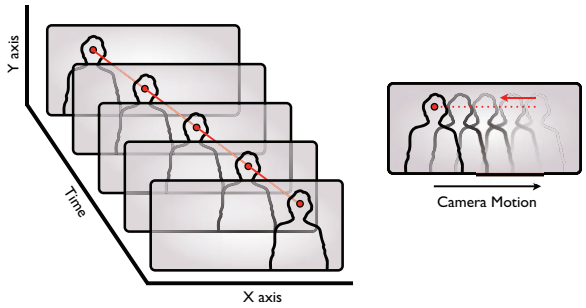

Figure 2. Scheme of how tracking a target can describe an EPI line in the $x-y-t$ space time volume. The red line represents an EPI line, and the rectangles frames of a video sequence panning from left to right. The red circle represents the target.

[9] suggested to compute the location of the original region in a new video frame f_i by building a weighted histogram $p(c)$ for a candidate target region centered around an initial location c,

$$p(c) = U^t(i)K(c). \tag{2}$$

To improve the initial location c we need to find $\triangle c$ minimizing the Matusita metric,

$$||\sqrt{q} - \sqrt{p(c+\triangle c)}||^2 = 0. \tag{3}$$

Eq. 3 is optimized by approximating $\sqrt{p(c+\triangle c)}$ using Taylor series,

$$\sqrt{p(c+\triangle c)} = \sqrt{p(c)} + \frac{1}{2} d(p(c))^{-\frac{1}{2}} U^t J_K(c) \triangle c \tag{4}$$

where $d(p(c))$ is the diagonal matrix with values $p(c)$ and J_K is the Jacobian matrix of the Kernel function and is defined as,

$$J_K = \left[\frac{\partial K}{\partial x}, \frac{\partial K}{\partial y}\right] \tag{5}$$

Representing $\frac{1}{2} d(p(c))^{-\frac{1}{2}} U^t J_K(c)$ in Eq. 4 by M we obtain the linearized objective function, $||\sqrt{q} - \sqrt{p(c)} - M\triangle c||^2$, leading to a standard least square solution:

$$\triangle c = (M^t M)^{-1} M^t (\sqrt{q} - \sqrt{p(c)}) \tag{6}$$

In MKT the same procedure is used to compute the new position of a region, but the model histogram is generated differently. Instead of using a single kernel to represent the target region, it uses several kernels placed over this region. The MKT model is created by stacking in a single vector Q all the individual histograms q. MKT allows to track larger regions without examining all pixels in those regions, and can compute higher order motion models.

B. Time Warping Motion Model

Following [5], EPI lines in a video sequence are trajectories of feature points in (x, y, t) 3D space, where t is the time of each frame. When the camera translates in a constant velocity in a static scene all EPI lines are straight lines, see Fig. 2. The slope of each line represents the depth of the corresponding feature point, where very far feature points will create lines almost perpendicular to the $x-y$ planes, while closer feature points will create a smaller angle with the $x-y$ planes. If we assume that the camera motion is a purely constant horizontal translation, the location of the feature point moves by a constant x displacement m between successive frames, where m corresponds to the depth of the point (the smaller m the more depth point). Formally, the EPI lines in the $x-y-t$ space-time volume can be recursively defined in a matrix form as,

$$\begin{pmatrix} x_i \\ y_i \\ t_i \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & m \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{i-1} \\ y_{i-1} \\ t_{i-1} \\ 1 \end{pmatrix}, \quad (7)$$

where m is the slope of the EPI line in the (x, t) axis and i is the frame number. We use homogeneous coordinates since the transformation involves displacements. In this case, if we "slice" the space-time volume by a $y-t$ plane we get a reasonable mosaic image of the scene. But in most cases the motion of a hand-held camera is not constant, and has rotations and vertical displacements. In such case EPI lines can be defined recursively in a matrix form as,

$$\begin{pmatrix} x_i \\ y_i \\ t_i \\ 1 \end{pmatrix} = \begin{pmatrix} cos(\triangle\alpha_i) & -sin(\triangle\alpha_i) & 0 & \frac{m}{\triangle t_i} \\ sin(\triangle\alpha_i) & cos(\triangle\alpha_i) & 0 & \triangle y_i \\ 0 & 0 & 1 & \triangle t_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{i-1} \\ y_{i-1} \\ t_{i-1} \\ 1 \end{pmatrix}, \quad (8)$$

where $\triangle t_i$, $\triangle y_i$ and $\triangle\alpha_i$ represent the motion parameters of frame f_i: the horizontal translation of the camera, the vertical image displacement, and the image rotation about the optical axis. These parameters are identical for all tracked feature points in image f_i, and represents the increments of these parameters respect to the previous image. Our goal is to compute the parameters $\triangle t_i$, $\triangle y_i$ and $\triangle\alpha_i$ (three parameters for each image) such that the EPI lines become straight, and Eq. 8 is satisfied. This is equivalent to compensate for the vertical displacements and rotations of the image, and compensating for the horizontal translation of the camera by time warping.

As we have seen in the previous section, the center c of the Kernel function was defined in terms of x and y coordinates and was optimized to satisfy Eq. 3. We would like now to optimize Eq. 3 in terms of t, y and α. We use Eq. 8 to find the center of the Kernel function c into the image frame and

Figure 3. Evolution of some tracked regions in a Video Sequence. Each circle represents a target and the color its state. From green to red are represented the SSD error (green less error). The targets detected on flat regions are pink, targets out of bounds are black and new targets white.

the Jacobian of the Kernel Function, which is computed from the gradients of the time warping parameters as follows,

$$J_K = \left[\frac{\partial K}{\partial t}, \frac{\partial K}{\partial y}, \frac{\partial K}{\partial \alpha}\right], \quad (9)$$

where $\frac{\partial K}{\partial t}$ and $\frac{\partial K}{\partial \alpha}$ can be computed using the chain rule as,

$$\begin{aligned} \frac{\partial K}{\partial t} &= \frac{\partial K}{\partial xy}\frac{\partial xy}{\partial t} = \frac{\partial K}{\partial x} m \\ \frac{\partial K}{\partial \alpha} &= \frac{\partial K}{\partial xy}\frac{\partial xy}{\partial \alpha} = \frac{\partial K}{\partial x}(y - C_y) - \frac{\partial K}{\partial y}(x - C_x), \end{aligned} \quad (10)$$

where (C_x, C_y) is the frame center. In this case the result of Eq. 6 becomes $\triangle c = (\triangle t, \triangle y, \triangle\alpha)$.

C. 3D Alignment by Time Warping

Our algorithm is based on two alternating steps. In the first step we use Kernel Tracking to track L target regions which are distributed over the frame as described in Sec. II-A. The tracks of these tracked regions, $\{k_j\}_{j=1:L}$, are used to estimate straight EPI lines by computing the global m_j slope for each track. The global slopes m_j are computed using an iterative mean of the target velocity in the $x-y-t$ space time volume, $(\frac{\triangle x_j}{\triangle t_i})$.

The second step (Sec. II-B) consists of computing the time warping motion parameters using the estimated slopes of the EPI lines as computed in the previous step.

It is important to maintain a uniform distribution of tracked target regions during the alignment process in order to get accurate motion estimation. We do this by dividing the frame in sections, requiring that one target will be selected from each section. If a section contains two or more targets,

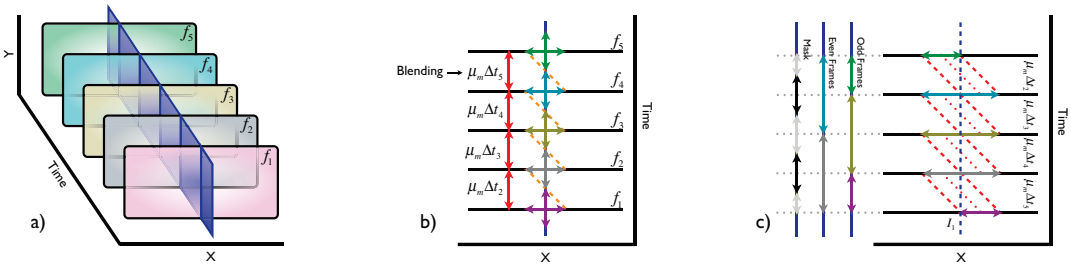

Figure 4. (a) Mosaic construction as slicing the $x-y-t$ space time volume. Five frames of a video sequence are shown in an aligned space time volume. The generated mosaic is the blue plane P_{y-t}. Each frame is represented by a different color. (b) Image strips around the intersection of P with each image are pasted onto P. In order to simplify the scheme we assume that the slope of the epipolar lines $\mu_m = 1$. (c) Odd, even and mask mosaics generation, in order to compute Barcode Blending.

only a single target is used. In our implementation we used the target region belonging to the longest track. In addition, new targets are selected for those sections having no targets.

Targets giving inaccurate trajectories are also removed. This includes targets placed on uniform regions, targets with a high SSD after alignment, or targets that overlap the image boundaries. Targets with a high SSD error or targets overlapping the image boundaries are trivial to detect. Targets placed on flat regions are detected using $\kappa_S(M_T M)$ (see Eq. 6), where in [10] it is demonstrated that the Schatten norm of the matrix $M_T M$ is related to the stability of tracking. When $\kappa_S(M_T M)$ is above a threshold it means that the tracker is placed on a flat region. Note that this process is performed in the first step of the proposed algorithm where only x and y parameters are estimated, therefore the matrix $M_T M$ is a 2×2 symmetric and positive definite. In such case $\kappa_S(A)$ can be easily computed as,

$$\kappa_S(A) = \frac{(a_{11}+a_{22})^2}{a_{11}a_{22}-a_{12}a_{21}}. \tag{11}$$

A summary of the proposed algorithm is shown in algorithm 1 and an example of the evolution of the tracks in the fist step is shown in Fig. 3.

III. Mosaic Construction

Once the video sequence is aligned, the mosaic image is generated by concatenating strips from different frames. Each strip $s_i \in f_i$ is generated by cutting a region surrounding the intersection $I_i = f_i \bigcap P_{y-t}$ between a frame f_i and a plane P_{y-t} in the $x-y-t$ space time volume. See Fig. 4(a). The width of strip s_i depend on the image alignment, s_i including image columns having the following x coordinates:

$$s_i = \{x \in f_i | [I_i - \frac{d_i}{2}, I_i + \frac{d_{i+1}}{2}]\}, \tag{12}$$

where $d_i = \mu_m \Delta t_i$, and μ_m is the mean of all EPI lines slopes m_j. See Fig. 4(b). Simple stitching of the strips produces visible misalignments caused by the parallax effect, and strip blending is needed.

Algorithm 1: Stereo Mosaic construction using Feature Tracking

Data:
Video Sequence $F = \{f_i\}_{i=1:N}$
Trackers $K = \{k_j\}_{j=1:L}$
Alignment Parameters $\Omega_0 = \{0,0,0\}$
Result:
Stereo Mosaic image M

```
begin
    for i ← 1 to N do
        Step 1. Track K;
        Step 2. Compute time Warping Ω_i;
        ∀k_j ∈ K update m_j;
        ∀k_j ∈ K replace if is necessary;
        Update the mosaic M using f_{i-1} and Ω_{i-1};
```

In this work we present the **Barcode Blending**, a novel blending method which is based on Pyramid Blending, but only a single blending step is required for blending all strips. This process is much faster and gives better results compared to the traditional approach of blending each strip separately.

Barcode Blending starts with the generation of two different mosaics without blending, one from the odd input frames and another for the even input frames. Strips in the Odd Mosaic are defined as,

$$s_{2k+1}^{Odd} = \{x \in f_{2k+1} | [I_{2k+1} - d_{2k+1}, I_{2k+1} + d_{2k+2}]\}, \tag{13}$$

and strips in the even mosaic are defined as,

$$s_{2k}^{Even} = \{x \in f_{2k} | [I_{2k} - d_{2k}, I_{2k} + d_{2k+1}]\}, \tag{14}$$

where $k \in [0, N-1]$. The number of strips in each of these two mosaics is half of the number of strips in the final mosaic image, and each strip has a double width. The final mosaic will be generated by pyramid blending from these two mosaics using a binary mask which defines which part

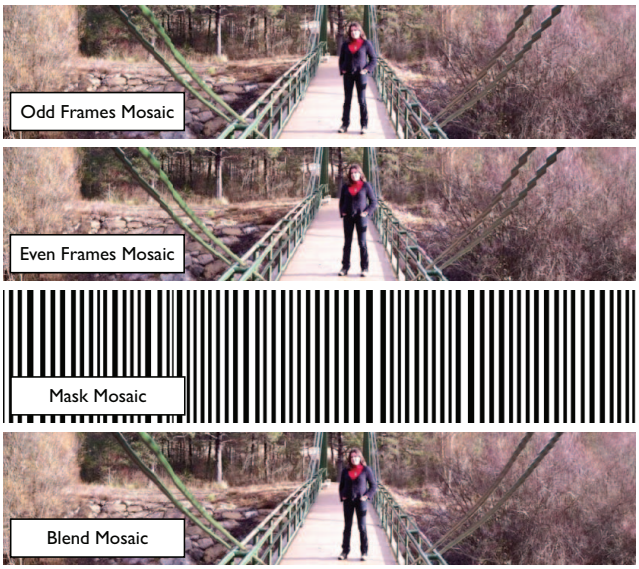

Figure 5. Example of Odd, Even and Mask mosaic and the resultant mosaic when Barcode Blending is applied.

of the final mosaic is taken from the odd mosaic and which is taken from the even mosaic. The mask therefore looks like a barcode whose strips have the same widths as the strips in the final mosaic, and is defined as,

$$s_i^{mask} = \begin{cases} [1, \frac{d_i}{2}] & = \begin{cases} 0 & \Leftarrow i \quad Odd \\ 1 & \Leftarrow i \quad Even \end{cases} \\ [\frac{d_i}{2}, d_i] & = \begin{cases} 1 & \Leftarrow i \quad Odd \\ 0 & \Leftarrow i \quad Even \end{cases} \end{cases} , \quad (15)$$

Fig. 4(c) shows the barcode blending process.

Once we have generated the odd mosaic, the even mosaic, and the mask, we blend the two mosaics according to the mask using a simple pyramid blending. Some results are shown in Fig. 5 and Fig. 6. It is important to note that generating the two mosaics and the mask is a trivial and fast process, and that pyramid blending is used only once.

IV. Experimental Results

In all the examples we used a uniform lattice of 12×12 tracked features. The use of anaglyph glasses will enable the reader to appreciate the 3D effect.

Fig. 7 used a video captured from a river boat. A small area is shown from different mosaics to emphasize the parallax effect between the different mosaic images.

The mosaics in Fig. 8 where made from video recorded by a hand held iPhone 4. The resolution of the captured sequences are 1280×720.

We have implemented the proposed method in C++, and tested it by tracking 144 features (on a 12×12 lattice) on three HD video sequences (1280×720) running on a Mac Book Pro 15' 2.66GHz Core i7. Alignment frame rate

Figure 6. Comparison between mosaic regions with and without Barcode Blending. The top images belong to a mosaic image generated without any blending and the bottom images are generated using Barcode Blending.

was ≈18 fps, with implementation not fully optimized, no hardware acceleration, and using a single thread. The most costly step is the tracking of 144 features, each of size 31×31 pixels, running at ≈30 fps. The 3D alignment is computed in a frame rate of ≈45 fps.

V. Concluding Remarks

The presented generation of stereo mosaics is faster and more accurate than previous approaches. The 3D alignment is based on feature tracking, which allows to process HD video sequences with low computational cost and memory requirements. While we used Kernel Tracking, any other feature tracking can be used.

The new Barcode Blending also presents a substantial improvement in speed and memory for mosaic blending. It is trivial to implement and results are better than traditional mosaic blending.

Acknowledgment

This work was funded by Universitat Autonoma de Barcelona (UAB) and Universistat Pompeu Fabra (UPF).

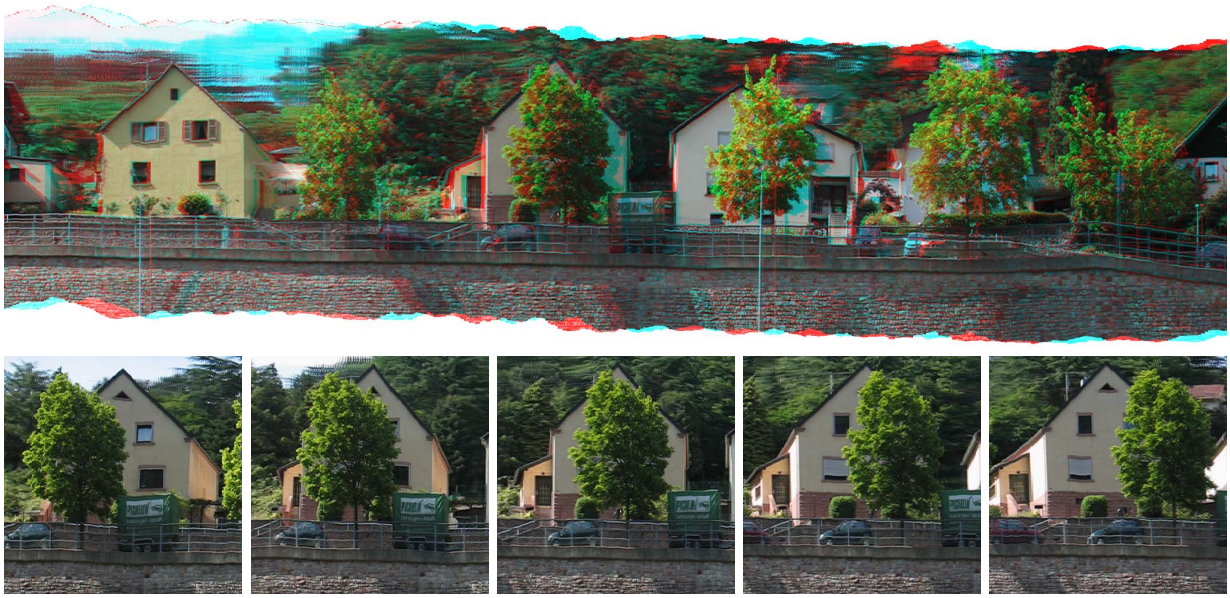

Figure 7. An anaglyph stereo mosaic generated from a video sequence of 450 frames captured from a river boat. The bottom images show the same regions as they appear in the different mosaics, showing the parallax effect.

Figure 8. Anaglyph stereo mosaics generated from videos recorded by a hand held iPhone 4, with resolution of 1280×720 pixels.

REFERENCES

[1] S. Peleg and M. Ben-Ezra, "Stereo panorama with a single camera," in *CVPR*, 1999, pp. 1395–1401.

[2] S. Peleg, M. Ben-Ezra, and Y. Pritch, "Omnistereo: Panoramic stereo imaging," *TPAMI*, pp. 279–290, 2001.

[3] A. Rav-Acha and S. Peleg, "A unified approach for motion analysis and view synthesis," in *3DPVT*, 2004, pp. 717–724.

[4] A. Rav-Acha, Y. Shor, and S. Peleg, "Mosaicing with parallax using time warping," in *CVPRW*, 2004.

[5] R. Bolles, H. Baker, and D. Marimont, "Epipolar-plane image analysis: An approach to determining structure from motion," *IJCV*, pp. 7–55, 1987.

[6] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI81*, 1981, pp. 674–679.

[7] M. Vivet, B. Martinez, and X. Binefa, "Dlig: Direct local indirect global alignment for video mosaicing," *TCSVT*, 2011.

[8] P. Burt and E. Adelson, "A multiresolution spline with application to image mosaics," *ACMTG*, vol. 2, pp. 217–236, 1983.

[9] G. Hager, M. Dewan, and C. Stewart, "Multiple kernel tracking with ssd," in *CVPR*, 2004, pp. 790–797.

[10] Z. Fan, M. Yang, Y. Wu, G. Hua, and T. Yu, "Efficient optimal kernel placement for reliable visual tracking," in *CVPR*, 2006, pp. 658–665.