Panoramic Stereo Video Textures

Vincent Couture
Université de Montréal
chapdelv@iro.umontreal.ca

Michael S. Langer
McGill University
langer@cim.mcgill.ca

Sébastien Roy
Université de Montréal
roys@iro.umontreal.ca

Abstract

A panoramic stereo (or omnistereo) pair of images provides depth information from stereo up to 360 degrees around a central observer. Because omnistereo lenses or mirrors do not yet exist, synthesizing omnistereo images requires multiple stereo camera positions and baseline orientations. Recent omnistereo methods stitch together many small field of view images called slits which are captured by one or two cameras following a circular motion. However, these methods produce omnistereo images for static scenes only. The situation is much more challenging for dynamic scenes since stitching needs to occur over both space and time and should synchronize the motion between left and right views as much as possible. This paper presents the first ever method for synthesizing panoramic stereo video textures. The method uses full frames rather than slits and uses blending across seams rather than smoothing or matching based on graph cuts. The method produces loopable panoramic stereo videos that can be displayed up to 360 degrees around a viewer.

1. Introduction

Stereo cameras capture two images of a scene from slightly different viewpoints, and when the stereo pair is displayed to a human viewer, one image to each eye, the images are fused and the disparities provide strong cues to scene depth, thereby enhancing the immersion experience. Stereo capture and display has a long and exciting history and we are now seeing a resurgence in popularity, especially in digital 3D cinema. The resurgence is, to a large extent, the result of recent advances made in computer vision and computer graphics which have been incorporated into many steps of the production pipeline [7].

Traditional stereo video uses two roughly parallel cameras, which maximizes the available stereo information near the optical axes. This paper addresses the more challenging problem of capturing stereo video over a much wider field of view, up to 360 degrees, and synthesizing the videos into a stereo panorama. One application of such *omnistereo* videos is for display screens with a very wide field of view. In the extreme case of a 360 degree cylindrical screen, observers would be be able to turn their gaze in any orientation and there could be more than one observer present, with different observers looking in different directions at the same time. A second and more "every day" application of a 360 degree stereo video panorama would be to use a standard display such as a stereo computer monitor, and to allow the user to pan over the 360 degree view. An example would be a stereo-video extension of Google Street View, where the motion could be texture such as waves on a river or lake, trees blowing in the wind, or a flag waving.

To capture stereo video in a wide range of directions, one could extend multi-camera systems. For example, the commercially available Ladybug [12] has five cameras to cover 360 degrees. One could extend such systems to stereo by doubling the number of cameras. Alternatively, one could attempt to combine previous computational approaches for static omnistereo and dynamic panoramas. We argue in Section 2, however, that one faces fundamental difficulties in doing so, related to stereo-motion synchronization. This leads us to take a different approach.

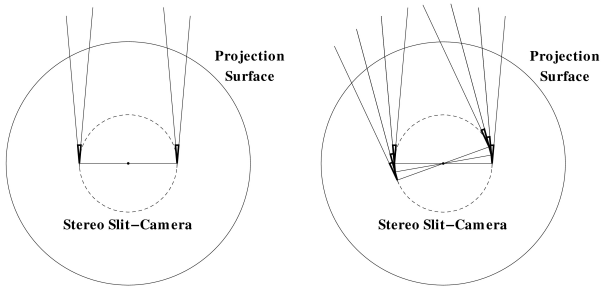
The approach we introduce in this paper uses a conventional stereo video rig where the two cameras each follow a circular motion and capture a space-time volume of images. We show how to combine the full frames of these videos into left and right panoramic video textures. The method is most effective with localized stochastic motions such as leaves moving in the wind or waves on a lake. Because our method uses full frames, most of the scene is viewed by both cameras simultaneously which guarantees stereo synchronization for these points. The method produces omnistereo video textures that are several seconds long and are loopable in time [14].

The paper is organized as follows. Section 2 gives a brief overview of previous work on omnistereo and dynamic panoramas. Section 3 gives the main details of our approach and outlines key differences from previous approaches. Section 4 presents example results, using our method. We conclude in Section 5.

Figure 1. Half a frame (180^o field of view out of 360^o) of an omnistereo video of a field, shown in red/cyan anaglyph format.

2. Previous Work

Traditional omnistereo methods are designed to give equal stereo resolution for all visual directions in a *static* scene [5, 6, 8, 9, 10, 15]. These methods typically gather several small field of view images called *slits* from one or two cameras rotating off-axis, and then make a mosaic of these slits. The slits typically cover one or two degrees, so 200-400 slits are used to capture 360 degrees. Figure 2 illustrates an omnistereo system that uses two cameras. An alternative omnistereo configuration uses a single camera with two slits, one to the left of center and one to the right [9]. This corresponds to two virtual cameras following a circle.



(a) (b)

Figure 2. An omnistereo method that uses a rotating stereo pair of parallel slit-cameras. (a) a pair of slits (b) slits of each camera are stitched together into a mosaic that covers 360° (only three slits are shown).

It is not clear how/if one could generalize these static omnistereo methods to dynamic scenes. In particular, one has to deal with a well-known problem in stereo imaging that objects at the edge of one camera's field of view often are not seen in the other camera [7]. This problem is especially difficult to solve if one uses slits, since every point is near a slit boundary. Using a two-camera omnistereo system (with one slit per camera), the only way that a point could be visible in both slits *at the same time* is if the point happened to have a similar disparity as the slit. For one-camera omnistereo, the slits cannot be synchronized anywhere since at any time the left and right slits are capturing different parts of the scene.

We next turn to dynamic monocular panoramas. An example is the *dynamosaicing* method [13] which makes a video mosaic by using graph cuts to compute a time evolving surface in a video's space-time volume. The surfaces are then stitched together to yield a video. A second graph-cut based approach [2] is *panoramic video textures*. This method renders a video seen by a rotating camera. Rather than selecting and stitching together slices in the space-time volume, it selects and stitches small space-time blocks. In addition to using graph cut matching to make the seams less visible, it also uses gradient smoothing. This method has been shown to be effective for dynamic panoramas that contain waves on a lake, a flag in the wind, etc.

Regardless of whether one stitches together surfaces in XYT or small blocks, the key problem remains of how to synchronize the left and right views. If one were to compute dynamic monocular panoramic textures for the left and right eye independently, using the above methods, there is no reason why the resulting panoramas would be synchronized, that is, there is no reason why the same scene events (*e.g.* a leaf blowing to the right) would appear at the same time in the left and right views and would have the correct disparity. One might try to extend the dynamic panorama methods to also enforce a constraint on stereo motion consistency, but such an extension is not obvious and would significantly increase the (already large) computational complexity of these methods.

The approach that we take differs from previous approaches in two fundamental ways. First, we use full video frames rather than slits or small blocks. Using full frames reduces the stereo-motion synchronization problem, which arises only near the boundaries of the regions being stitched together. That is, using full frames rather than slits or small blocks reduces the percentage of pixels that lie near the boundaries. The second difference is that, rather than using graph-cut based stitching or gradient smoothing to reduce the visual seam boundaries between regions, our method simply blends neighbouring regions together. many cases, namely the blended regions are not visually salient in practice for motions that are stochastic and localized, such as water flows or leaves in the wind. Although the blending is visible in some cases if one scrutinizes the video, it is typically not visible in casual viewing.

3. Our approach

The panoramic stereo video problem begins with a stereo video pair which has been captured by rotating a stereo rig around a vertical axis. Each camera follows a circular path similar to Figure 2. In each of the examples we present in Section 4, we capture a full 360 degrees. The videos are about 2 minutes each, *i.e.* a few thousand stereo frames.

3.1. Camera calibration and video registration

Given the left and right videos, we first calibrate the stereo camera rig, both for the camera internals (focal length) and externals (position and orientation in each frame). This calibration allows us to map pixels in the two cameras in each frame to pixels on a cylindrical projection surface (one for each camera). This yields left and right XYT volumes, composed of frames that shift over time as the cameras move. The camera calibration method and the mapping from camera pixels to the cylindrical projection pixels use standard computer vision techniques. Details are briefly summarized as follows.

As a first approximation, we estimate camera parameters by ignoring camera translation and sub-sampling the frame sequence in time. We compute SIFT features in these frames and compute homographies between frames using RANSAC for robustness. We then estimate camera parameters (rotation, focal length) and perform bundle adjustment, taking radial distortions into account [4]. The next step is to improve and complete the previous estimates by considering all the frames. We track features between frames, allowing for small camera translation, and perform another bundle adjustment that triangulates features in 3D [16].

The next section describes a few theoretical observations underlying our method. We then present the method itself. For the sake of clarity, we begin with a simple case in which the stereo rig rotates at constant angular velocity. This would be the case if the camera motion were driven by a motor, for example. With uniform rotation, the boundary directions of the calibrated space-time volume are two diagonal planes of constant x-slope, namely lines in Figure 3. In Section 3.5 we will return to the more general case that the camera rotation may be non-uniform.

3.2. Motion paths and parallax

As the stereo rig rotates, the projection of a fixed point in the scene moves across the image. This image motion is a combination of the motion of the scene point and the camera's rotation and translation. To understand this image motion, first consider two *static* scene points $P_{Z_{min}}$ and P_∞ that enter the field of view at the same time and the same position, namely at one edge of the frame. See Figure 4. As the rig rotates, *motion parallax* occurs and the x-pixel positions of these two points diverge slightly as they cross the frame. The x positions converge and meet again when the points leave the field of view at the opposite edge, namely

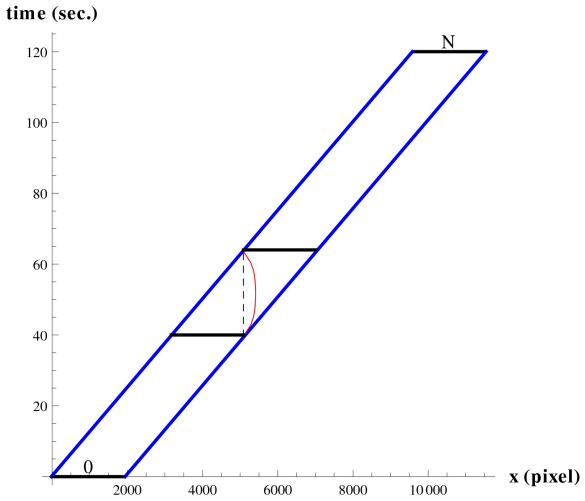
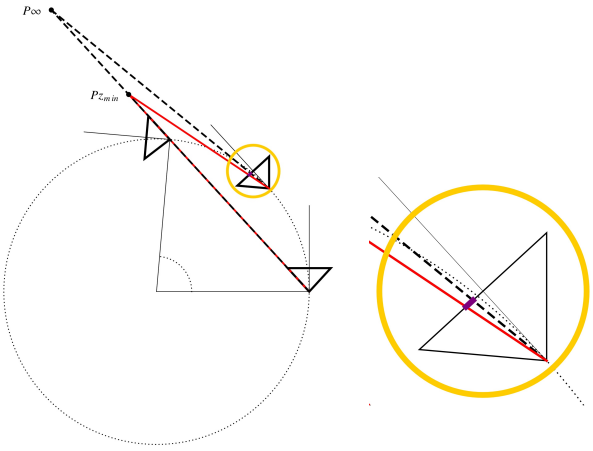


Figure 3. A sequence of N frames captured by the left or right camera performing a full turnaround, after calibration and registration. For simplicity, we assume in this figure that the rig rotates at constant speed. The dashed and red lines represent the path followed by two static points, one far and one close, respectively (see Fig. 4), that enter and exit the field of view at the same time. The two thick black lines represent the entry and exit frames [3].



(a) (b)

Figure 4. The right camera only is shown. Let the camera rig rotate clockwise. (a) Two points $P_{Z_{min}}$ and P_∞ are shown that enter the camera frame at the right edge at the same time (top). These points also exit the frame at the same time (right). As the camera moves, the positions of the points drift across the frames. The depth difference of the two points leads to motion parallax. See expansion of the yellow circle in (b). The figure is not to scale, namely the camera baseline is typically much smaller than the distance to scene points and so parallax is typically very small. In this example, the camera field of view is 90^o, but the argument about coincidence at the left and right edge holds for any field of view size.

when the camera center again lies on the line connecting the two points. In practice, the separation that is due to this motion parallax is maximum at the center of the frame and is a few pixels only [3]. This parallax magnitude is comparable to that seen by a person who translates his head slightly, as in normal posture adjustment.

The space-time paths of the two points are sketched in Figure 3. The dashed vertical line traces the constant visual direction of the point at infinity P_∞. The red curve traces the changing visual direction of the point $P_{Z_{min}}$ which is at a finite distance.

In addition to the horizontal parallax just discussed, there can also be a slight vertical parallax. For example, a point at infinity that enters the field of view at the top right corner of a frame will leave the frame at the top left corner, but if a point that is a finite distance were to enter at the top right corner at the same time then it would leave the frame earlier, namely at the vertical edge (before it reaches the top left corner). This vertical parallax is zero for points on the horizontal mid-line and increases to a few pixels high toward the upper and lower corners. [1]

A few other observations about motion paths are worth mentioning. First, our arguments above follow from geometry illustrated in Figure 4 and do *not* require that the camera rotation speed is uniform, *e.g.* the paths of the two points in Figure 3 meet at the boundaries, regardless of whether the boundaries are straight or curved. Second, the above argument considers static scene points only. For scene points that are moving, the image paths will depend on the parallax just discussed and on the scene motion. Only the latter creates synchronization problems at frame boundaries, as we discuss in the next section. Third, the motion paths discussed above were for one camera only. How are the motion paths for the two cameras related? Points that are a finite distance away will enter each camera's field of view at slightly different frames. This is the well-known monocular problem at frame boundaries where some points are only seen by one of the two cameras because of image disparities. In addition, the shape of the corresponding red curves will be slightly different for the two eyes, which causes disparities to vary slightly over time. The parallax effects are very small and in our examples there are visible only under careful scrutiny.

3.3. Partition and alignment of space-time blocks

Suppose at this point that calibration and registration has been done, so that the pixels in each frame have been remapped to the cylindrical projection surface. Let the two videos have N frames each. In the case that the camera turns 360 degrees, frame N is where the camera completes the full turnaround. Thus, frame N would be registered with frame 0.

At this stage, if we were to display the image sequence in stereo on a cylindrical projection screen, we would see the scene through a window translating slowly over time, namely we would see the scene in stereo as captured by the rotating camera and projected in the correct direction. At any time, we would see only the field of view of the stereo camera, however. The problem that we are solving is to take this stereo video and make a panorama stereo video from it, which is defined over the entire cylinder and at every time.

The main idea of our method is to partition each of the stereo XYT volumes into blocks (parallelipeds), and then to stitch the blocks together to form left and right video textures. To explain how this partitioning and stitching works, we continue for now with the simplified case that the camera rotation is uniform.

Suppose that it takes T frames for any point to enter the field of view at the right edge and exit the field of view at the left edge. (This time is constant when the camera rotation speed is constant, and the scene point is static.) We partition the entire image sequence into a set of consecutive blocks, each of T frames. We then re-align the blocks so that they all start at the same frame in the video texture. See Figure 5. In this example, the entire video is 120 seconds and is partitioned into five blocks that are 24 seconds each.

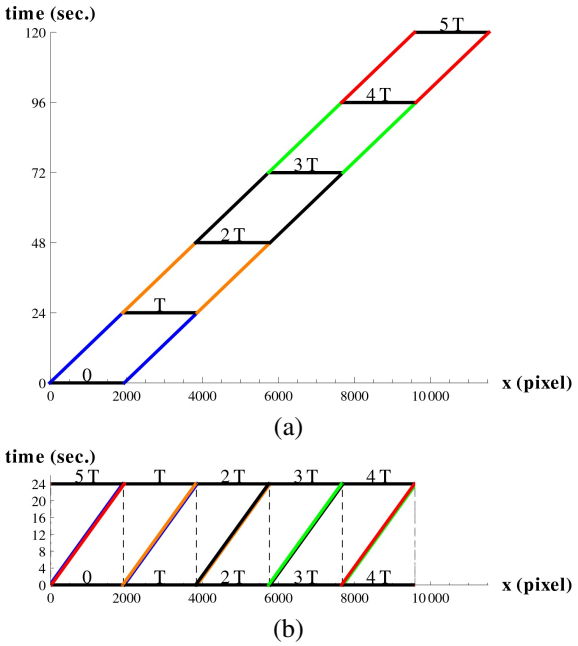


Figure 5. For a frame sequence captured by a camera performing a full turnaround in $N = 5T$ seconds at constant speed. (a) The full original space-time volume divided in five non-overlapping blocks. (b) The blocks are aligned to start at the same time.

[1] The amount of horizontal and vertical parallax depends on camera resolution, field of view and the range of scene depths. For HD cameras having a 60 degree field of view and scene depths ranging for 2m to infinity, it can be shown that maximum parallax is about 5 pixels wide and 7 pixels high.

Consider a scene point at infinity that enters the field of view somewhere on the right diagonal of the first block. Since this point is within the field of view for T frames, its path extends beyond the first block. When the blocks are aligned so that they all start at the same frame, the vertical path followed by this point wraps around from frame T to frame 0 and again forms a vertical line. See Figure 6.

Recalling the arguments of Section 3.2, if a static scene point at a finite depth were to enter the space-time volume at the same frame, then it would take a curved path instead of a vertical path. The curved path would also wrap around from frame T to frame 0, and rise again to meet the vertical dashed line at the diagonal boundary. Thus, the paths of static points in the scene would be continuous both at their temporal boundaries (allowing the video to loop with a period of T frames) and also at the seams that define the frame's spatial boundaries, *i.e.* the diagonals.

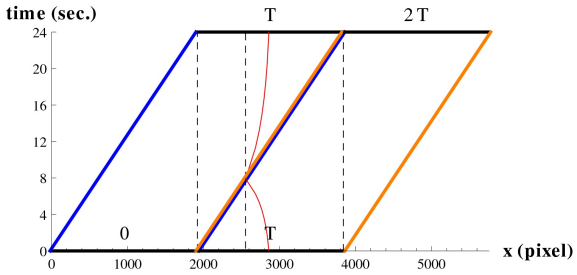


Figure 6. Motion paths of two static objects, one far away (central dashed vertical line) and one close-by (red curve). In both cases, the motion paths are continuous and loop. The video goes from frame 0 to T-1 and then loops so that frame T equals frame 0.

The continuity at the temporal boundary (looping) does *not* depend on any assumptions about the points being static in the scene, nor does it depend on the camera rotation speed being constant. *This looping property at the temporal boundaries always holds.*[2] The continuity at the frame boundary, though, often does not always hold exactly. Discontinuities can occur when there is scene motion (see Fig. 7) as just discussed, and also when there is vertical parallax, or lighting changes over time, or exposure changes due to a camera aperture change e.g. if one is in in shutter priority mode.

How can one avoid such visual seams? Existing monocular methods that render dynamic mosaics [2, 13] attempt to minimize seams both in space and time by using graph cut matching and/or gradient smoothing. As we discussed

[2] The only exception occurs when the frame at 360^o loops to the frame at 0^o. In this case, if there are moving scene points at these limit frames and/or the lighting changes, then the video will not be loopable at these points. The problem could be lessened by starting the capture in a direction in which the scene is static, or using a blending technique similar to what we discuss next. Similarly, if the panorama is less than 360 degrees and the scene has motion at frames 0 (or N), our method will not produce looping there.

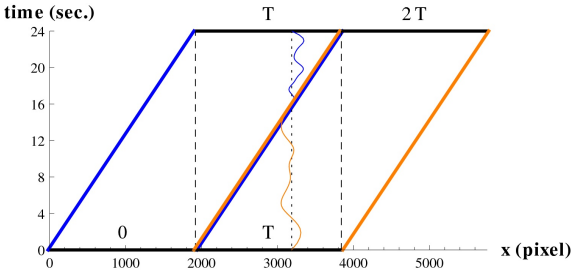


Figure 7. For an object moving in time in a small area (a leaf for instance), the motion is continuous at the temporal boundary (horizontal edge), but there will be a motion discontinuity at the spatial boundary (seam), namely the diagonal edge.

in Section 2, however, it is unclear whether such methods could be extended to dynamic stereo since such methods use thin slits or small blocks, and there are fundamental difficulties in stereo motion synchronization in these cases. Our approach is to avoid boundaries as much as possible, by using full frames rather than slits or small blocks. We still need to stitch boundaries together, however, and for this we use blending as we describe next.

3.4. Blending adjacent blocks

To blend adjacent blocks, we decrease the duration T of each block and shift the block by the number of pixels covered during that decrease in duration. See Figure 8 for an illustration of what happens for static scene points, and see Figure 9 for the case of a moving scene point. In these figures, T has been decreased from 24 to 20 seconds. In the overlap region, we blend the frames together using a simple linear ramp function (see Figure 10).

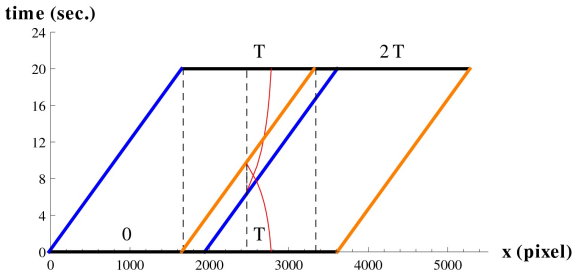


Figure 8. Images are blended near the boundaries of two blocks.

The reason we use blending, rather than a more sophisticated technique such as gradient based smoothing [11], is that it seemed to be sufficient. Although blending does leads to a duplication of the points – or "ghosting" – the duplication is typically not perceived, unless one is looking for it. There are several reasons for this. First, the blending is continuous over time, with one copy fading out over time and the other copy fading in, and so there are no salient low level features that grab the eye such as block edges or temporal popping. Second, in the case of motion texture such as

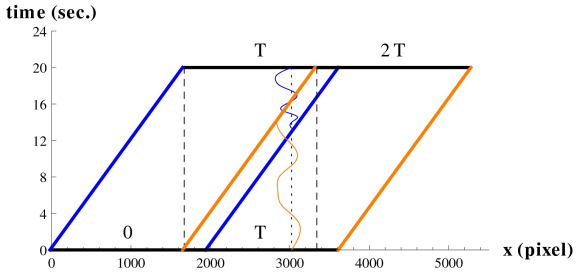


Figure 9. For an object moving in time (a leaf, for instance), motion is blended over the overlap between the two blocks.

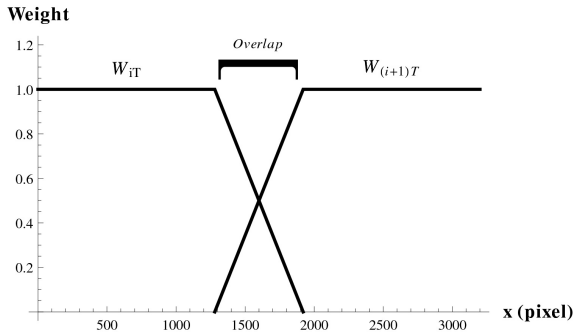


Figure 10. Blending function for overlapping frames.

leaves or grass blowing, or waves evolving, the "texels" often change over time or undergo occlusions and so are not visually isolated and trackable entities. Third, the blending window translates which results in further variation that may mask the duplication of texels.

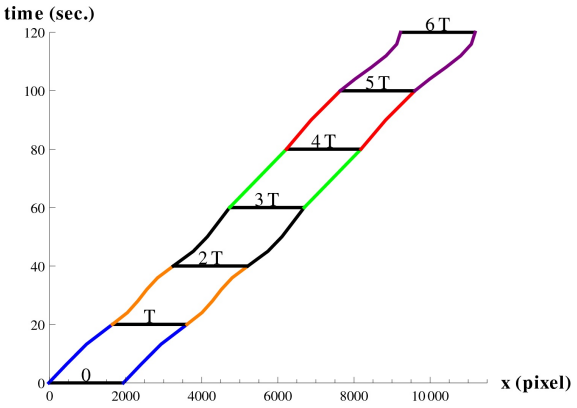
3.5. Non-uniform camera rotation and blending

We next turn to the more general case that the camera rig is rotating at a non-uniform speed, and so the boundaries of the space-time volume are not straight (see Fig. 11). To handle the non-uniform speed, we continue to use a constant duration T for all blocks, but now we vary the blending overlap. The blending width depends on the frame-to-frame overlap at each boundary, which corresponds to the distance between two adjacent diagonal curves in Figure 11(b).

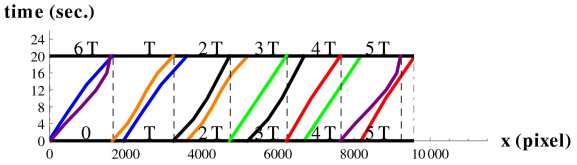
To ensure there is some overlap between each pair of adjacent frames, T must be chosen carefully. Let $d(i, i')$ be the angular distance (in units of pixels on the cylindrical projection surface) travelled by the camera between frames i and i'. To be conservative, we require that, for all frames j, the distance $d(j, j+T)$ is less than or equal to some chosen fraction α of the width W of the original frame. This ensures a blending overlap of at least $(1-\alpha)W$ pixels between frames. Given α and W, a sufficient condition on T that ensures some overlap is that, for all frames j,

$$\sum_{i=j}^{j+T-1} d(i, i+1) \;\leq\; \alpha W \;.$$

In our experiments, we chose $\alpha = 0.8$ which ensures a minimum overlap of 20%. The result is that the overlap is smaller when the stereo rig rotates faster and the overlap is larger when the rig rotates slower.



(a)



(b)

Figure 11. Similar to Figure 5 except that here the camera rig rotational velocity is not constant. The blocks are no longer aligned. Blending over frame boundaries is used to reduce the visibility of seams.

3.6. Stereo motion synchronization

We have discussed how we partition and blend the blocks of each camera's video. But to what extent are the resulting left and right panoramic videos synchronized ? For scene points that are imaged simultaneously by the left and right cameras, stereo-motion is automatically synchronized since we are using the full frames. The asynchronization occurs only near the frame boundaries, namely for points visible to one camera but not the other at a given time. Note however that not all points near the frame boundaries are monocular and asynchronous. For example, if the cameras are exactly parallel, then points at infinity always will be binocular. It follows that the vast majority of visible points in each frame will be synchronized between the left and right cameras.

Finally, for those points that are imaged near a frame boundary at some given time, it is important to distinguish synchronization issues from blending issues. Blending can introduce duplicated scene points in either camera's video. But if these duplicated points are seen simultaneously in the left and right views then they will be synchronized, namely there will be two blended but distinct stereo copies of the scene points and the disparities of each copy will be correct. We have found that this ghosting is visible mainly when the

duplicated object is a visually isolated and trackable scene feature such as a single tree branch. But even then, it is often only noticeable when one is looking for it.

4. Results

In our experiments, we used two Canon HFS11 cameras on a fixed tripod. This allowed camera rotation around an almost vertical axis (y axis). The distance between the centers of both lens was about 6.5 cm, similar to the typical distance between human eyes. To synchronize frame capture as well as zoom, both cameras were controlled through the LANC protocol. (A LANC Shepherd was connected to the cameras by RA-V1 remote control adaptors.)

To speed up the experiments, we down-sampled the HD original content, from 1920×1080 resolution to 960×540. The final panoramic video is high-resolution at about 6500×540 pixels per eye. A GPU was used for the dense frame calibration and the blending. Each example took about an hour to render, separated about evenly between calibration and blending, on a laptop with an NVidia GeForce 8400M graphics card and an Intel dual core T7500 2.2 Ghz CPU and 2GB of RAM. Both steps could be accelerated by having a separate thread handling disk operations (loading and saving frames). Moreover, both the calibration and the blending steps have low memory requirements. Every output frame of the video texture can be blended in parallel, which allows the method to render very high-resolution 360 degree textures. This contrasts with other approaches [2, 13] that require solving a large minimization over the whole space-time volume.

We present two examples: one containing a river and another containing a field with blowing tall grass. See Figs. 1 and 12 for half of a single frame of each video (full videos are available online at [1]). The reason we show half a frame only is that the 12:1 aspect ratio (horizontal:vertical) of the entire frame is very large and the vertical dimension would be excessively squeezed.

Figure 12 shows a single frame from the left camera's panoramic video texture and compares (a) no blending, versus (b) blending. At first glance, the seams in (b) are slightly visible when seen below (a). However, this is an illusory contour effect. The reader should cover up (a) when examining (b).

To fully appreciate the stereo effects, the videos should be displayed with correct perspective. We have projected them on a cylindrical screen made of a silver fabric that maintains light polarization. The screen is about $1.5m$ high with a $4.5m$ diameter. A multiprojection system [18, 17] was setup with half the projectors polarized horizontally and the other half polarized vertically, and viewed with glasses for polarized projection. To our knowledge, this is the first time that a 360 panoramic stereo video texture has been captured, computed and displayed.

Finally, although our method is motivated by the problem of stereo video panoramas, it also applies to the more specific problems of static omnistero and to dynamic monocular panoramas. For example, we tested our method on monocular input videos from [2], which were shot by a single camera rotating on a tripod. (These sequences do not have parallax since the camera undergoes pure rotation.) The result for the *Y*achts sequence is available online at [1]. In this example, camera rotation stops at a few discrete positions which causes blending overlaps to increase considerably. Nonetheless, our method produces very good results.

5. Conclusion

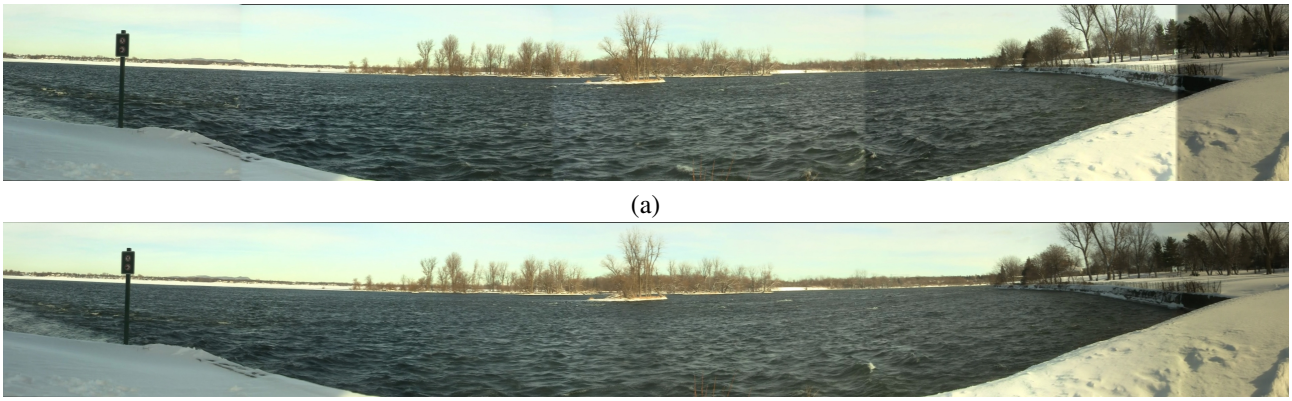
This paper has introduced a method for computing panoramic stereo video textures using a pair of off-the-shelf consumer video cameras. There are several key ideas to the method. First, it uses the full frame video which gives automatic stereo motion synchronization for the vast majority of the visible points. This synchronization issue would be problematic if one were to use slits or small blocks as in previous stereo or motion panorama methods. Second, rather than using graph cuts and/or smoothing to stitch together seams from different parts of each camera's XYT volume, we use simple blending. While blending can create duplicates of points, this duplication is typically not visible since the blending is continuous and the blended points are part of a texture.

The main limitations of our method are similar to those faced by other stereo capture methods and mosaicing methods. For stereo capture, there is always a problem of what to do with monocular points at the frame boundary. For mosaicing, there is always the problem of how to stitch boundaries together so that the image is smooth. For the latter problem, one might try to improve our method using a more sophisticated smoothing technique, rather than blending, though in many cases this is unnecessary since blending works well. Finally, like other dynamic mosaicing methods, we assume motion texture (defined loosely), rather than structured aperiodic motion. When the latter is present – for example, an isolated moving object – it would be smoothly blended in or out as the frame boundary passes over it.

References

[1] http://vision3d.iro.umontreal.ca/en/projects/omnistereo/. 7

[2] A. Agarwala, K. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic video textures. *ACM Transactions on Graphics*, 24(3):821–827, 2005. 2, 5, 7



(a)

(b)

Figure 12. Half a frame (180^o field of view out of 360^o) of one camera's panoramic video with (a) no overlap between the blocks. (b) a minimum overlap of 20% between blocks. The overlap blends motion discontinuities as well as lighting changes.

[3] V. Couture, M. S. Langer, and S. Roy. Capturing non-periodic omnistereo motions. In *10th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*, Zaragoza, Spain, 2010. 3, 4

[4] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 3

[5] H.-C. Huang and Y.-P. Hung. Panoramic stereo imaging system with automatic disparity warping and seaming. *Graphical Models and Image Processing*, 60(3):196–208, 1998. 2

[6] H. Ishiguro, M. Yamamoto, and S. Tsuji. Omni-directional stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):257–262, 1992. 2

[7] B. Mendiburu. *3D Movie Making: Stereoscopic Digital Cinema from Script to Screen*. Focal Press, 2009. 1, 2

[8] T. Naemura, M. Kaneko, and H. Harashima. Multi-user immersive stereo. *IEEE International Conference on Image Processing*, 1:903, 1998. 2

[9] S. Peleg and M. Ben-Ezra. Stereo panorama with a single camera. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:1395, 1999. 2

[10] S. Peleg, M. Ben-Ezra, and Y. Pritch. Omnistereo: Panoramic stereo imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):279–290, 2001. 2

[11] P. Perez, M. Gangnet, and A. Blake. Poisson image editing. In *SIGGRAPH Conference Proceedings*, pages 313–318, New York, NY, USA, 2003. ACM Press. 5

[12] Point Grey Research. *Ladybug3*, 2008. 1

[13] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg. Dynamosaicing: Mosaicing of dynamic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1789–1801, 2007. 2, 5, 7

[14] A. Schödl, R. Szeliski, D. Salesin, and I. Essa. Video textures. In *SIGGRAPH Conference Proceedings*, pages 489–498, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 1

[15] S.M. Seitz, A. Kalai, and H.Y. Shum. Omnivergent stereo. *International Journal of Computer Vision*, 48(3):159–172, July 2002. 2

[16] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press. 3

[17] J.-P. Tardif and S. Roy. A MRF formulation for coded structured light. *International Conference on 3-D Digital Imaging and Modeling*, pages 22–29, 2005. 7

[18] J.-P. Tardif, S. Roy, and M. Trudeau. Multi-projectors for arbitrary surfaces without explicit calibration nor reconstruction. *International Conference on 3-D Digital Imaging and Modeling*, pages 217–224, 2003. 7