A Comparative Study of Algorithms for Realtime Panoramic Video Blending

Zhe Zhu, Jiaming Lu, Minxuan Wang, Songhai Zhang, Ralph R. Martin, Hantao Liu, and Shi-Min Hu, *Member, IEEE*

***Abstract*—Unlike image blending algorithms, video blending algorithms have been little studied. In this paper, we investigate 6 popular blending algorithms—feather blending, multi-band blending, modified Poisson blending, mean value coordinate blending, multi-spline blending and convolution pyramid blending. We consider their application to blending realtime panoramic videos, a key problem in various virtual reality tasks. To evaluate the performances and suitabilities of the 6 algorithms for this problem, we have created a video benchmark with several videos captured under various conditions. We analyze the time and memory needed by the above 6 algorithms, for both CPU and GPU implementations (where readily parallelizable). The visual quality provided by these algorithms is also evaluated both objectively and subjectively. The video benchmark and algorithm implementations are publicly available[1].**

***Index Terms*—Video blending, video stitching, video quality assessment, panoramic video.**

I. Introduction

Many image editing [1] tasks involve blending images, e.g. panorama stitching, or copying-and-pasting of objects into images. As human eyes are sensitive to color and lighting inconsistencies within images, image blending is used to provide smooth transitions between image parts from different sources. Image blending is now a standard part of modern image editing tools such as Adobe Photoshop.

While state-of-the-art image blending algorithms [2], [3], [4], [5], [6], [7] can achieve good results, it is difficult to find evaluations of the trade-off between their speed and quality of results. This is mainly because these algorithms can provide high quality results in a short time: e.g. mean value coordinate blending [5] can blend a region with 1 million pixels in about 1 s.

For video blending, especially at high resolution, the situation changes. The quantity of data is much higher, so efficiency becomes a major concern. For example, virtual reality applications, e.g. involving live sports, can demand real-time content creation based on 360° panoramic video blending; these panoramic videos are much larger than ordinary videos. In a typical 4k 360° 30fps panoramic video, blending must be done in under 30 ms (and often significantly less to allow time for other processing tasks on each frame). Thus, real-time high resolution video blending is much more challenging than image blending, and indeed, parallelization is often needed.

Recently, many works have considered generating high-quality panoramic videos by stitching multiple videos together. The input videos may be captured from structured [8], [9] or unstructured camera arrays [10], or even multiple moving cameras [11], [12]. Early work [13] addressed the parallax issue by recovering depth information in the overlap region followed by new view synthesis. Later, various energy functions [14], [15] were proposed to calculate a warp to alleviate issues due to parallax while preserving spatial-temporal coherence. While such approaches can often generate high-quality panoramic videos, they reply on sophisticated video content analysis. This makes such methods unsuitable for realtime processing, and at times, they may lack robustness.

As noted, then, the aim of this paper is to compare the suitability of various image blending algorithms for real-time usage for video blending in high resolution panoramic video stitching. We first briefly describe each algorithm, as well as analysing the relationships between them. Then, we conduct experiments on a benchmark dataset, evaluating both their performance on different kinds of scenes, considering both time and memory costs, and the quality of the blended results, using both objective and subjective assessments. Unlike image blending, which is a one-shot operation, video blending involves a sequence of frames, which may share a common fixed camera position. Some algorithms take advantage of this by performing a (possibly lengthy) precomputation. For short video clips, when using methods like mean value coordinate blending, the precomputation may even comprise the majority of the computation.

We do not include content-aware blending algorithms [16], [17] in our comparison as they are unsuited to real-time video blending, for two reasons: (i) these methods are relatively slow due to the need to analyze content, and (ii) they cannot readily provide interframe coherence.

We have captured a set of benchmark videos including various types of scenes, to enable evaluation of different aspects of the blending algorithms. Each video has 6 separate overlapping streams; we also provide a stitching template which defines where the pixels in each stream are to go in the final panorama. A blending algorithm under test uses this information to produce the panoramic result.

Z. Zhu, J.-M. Lu, M.-X. Wang and S.-H. Zhang are with the TNList, Tsinghua University, Beijing 100084, China. E-mail: ajex1988@gmail.com,loyavaforever@gmail.com, hitminxuanwang@gmail.com,shz@tsinghua.edu.cn

R.-R. Martin and H.-T. Liu are with the School of Computer Science & Informatics Cardiff University, Cardiff, Wales, CF24 3AA, UK. Email:ralph@cs.cf.ac.uk,LiuH35@cardiff.ac.uk

S.-M. Hu is with the TNList, Tsinghua University, Beijing 100084, China, and also with the School of Computer Science and Informatics, Cardiff University, Cardiff CF24 3AA, U.K. (e-mail: shimin@tsinghua.edu.cn; hus3@cardiff.ac.uk)

[1]http://cg.cs.tsinghua.edu.cn/blending/

Fig. 1. Capture device. Left: camera rig holding multiple cameras. Center: rig mounted on a car. Right: rig mounted on a tripod.

The contributions of this paper are:

- A comparative study of the suitability of several state-of-the-art image blending algorithms for panoramic video blending, making clear the advantages and disadvantages of each algorithm, as well as the relationships between them. Our implementations of these algorithms are publicly available.
- A benchmark set of videos for evaluation of panoramic video blending methods, again publicly available.

In Section II we describe our benchmark. We describe the different blending algorithms tested and their relationships in Section III. The behaviour of these algorithms on our benchmark is examined in Section IV, and we give our conclusions in Section V.

II. Benchmark and Experimental Setting

A. Hardware

To evaluate the performance of blending algorithms when used for panoramic video blending, we captured videos from various indoor and outdoor scenes to make a benchmark. Our camera mount was based on a six GoPro camera rig—see Figure 1. Five cameras were arranged symmetrically in a plane around a vertical axis, while the last camera pointed vertically upwards. A GoPro Smart Remote was used to synchronize video capture from all cameras. Each video has the same resolution (1920×1440, at 30 fps).

B. Formulation

The 6 video streams S_t, $t = 1, \ldots, 6$, are recorded simultaneously using the rig. Before blending, these must be transferred to a single frame of reference in which the panorama is described. Given our fixed camera rig, and known camera intrinsic parameters, we first perform radial and tangential distortion correction for each stream, then match keypoints [18] between adjacent streams. We pick one frame as a reference for each stream and apply its correction to all remaining frames, to ensure coherence between frames. We then select one stream as a base, and rotate other streams in the viewing sphere according to the yaw, pitch and roll which best match the keypoints. Spherical projection is then used to map the rotated content in the viewing sphere to the planar panoramic output video. We finally perform a local varying warp following [19] to further align details. For example, the stream of the top camera (see Figure 1) is mapped to the top region in the output panorama (see Figure 2). The resulting

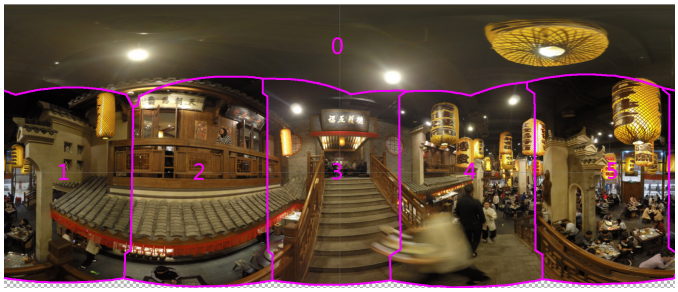


Fig. 2. A typical stitched panorama. Region 0 is captured by the upwards-pointing camera. Regions 1–5 are captured by the other cameras. Purple lines indicate the boundary seams between overlapping adjacent video streams.

pixels in the panorama corresponding to each initial stream are determined by a mapping function:

$$P_t = \varphi_t(S_t). \tag{1}$$

The final panorama has resolution 4000×2000.

We associate a mask M_t with each mapped stream P_t, which contains 1 for panorama pixels covered by that stream, and 0 for pixels not covered. Each mapped stream overlaps adjacent streams by about 20% of its total area, an overlap being necessary for certain blending algorithms. Other blending algorithms require boundaries between streams, which we determine in the overlap region using distance transforms [20] on the first frame to place seams at locations equidistant to each pair of adjacent streams. This boundary is used to trim the original mask M_t to a new mask M'_t. See Figure 2.

We may divide blending algorithms into two categories: those that calculate the blended pixels directly, and those that first compute a color offset map, and then add the offset map's colour values to each original video. The offset map is an image with the same resolution as each video frame, in which the pixel value at each position is the difference between the desired blended value and the original value.

Algorithms in the first category obtain the final panoramic video P by computing:

$$P = f(P_1, M_1, \ldots, P_z, M_z) \tag{2}$$

where f is some function that performs operations on the mapped streams $P_1, \ldots, P_z$.

Algorithms in the second category produce the final panoramic video by computing:

$$P = P' + P^* \tag{3}$$

where P' is a video obtained by directly trimming and compositing the mapped input streams along precomputed boundaries, and P^* is a combined offset map formed from the offset maps of each mapped input stream, using the same boundaries. Thus P' is computed using:

$$P' = \sum_{t=1}^{6} M'_t P_t{}' \tag{4}$$

where P'_t is the mapped t^{th} stream. P^* is defined in a similar way:

$$P^* = \sum_{t=1}^{6} M'_t P_t{}^*. \tag{5}$$

The way in which the individual offset maps P_t^* are computed varies according to the blending algorithm.

C. Benchmark

To produce the benchmark, we captured videos with variation in three key properties—illuminance conditions, camera motion, and object distance to the camera—as the quality of blending results can be significantly effected by changes in these properties. In the benchmark, illuminance variations cover both indoor and outdoor scenes, with adequate and poor lighting. While video cameras often automatically determine exposure, changes in illuminance conditions may have a strong effect on the brightness of the video. We provide videos from both static and moving camera setups. The latter cause content to change along the boundary seams, sometimes substantially, e.g. if the camera rig is mounted on a moving vehicle. As noted, for video blending, we manually pick one frame as a reference and compute a stitching template, and apply this template to all other frames, to ensure coherence in the blended video. However, even if we perfectly stitch the reference frame, when the template is applied to other frames, this can result in misalignments. Thus, distances of key objects from the video cameras can also affect blending results; objects with varying distances can cause bleeding artefacts near seams. As human eyes find larger objects more salient, artefacts in objects closer to the cameras are often more obvious. We thus captured scenes with moving objects at near, intermediate and far distances. In total, we have 4 illumination conditions, 2 motion types, and 3 distance types, giving 24 types of video; our benchmark provides 2 of each type, giving 48 video scenes altogether, each lasting from a few to tens of minutes.

III. Blending Algorithms

A. Overview

Image blending is well studied. Perhaps the most widely used approach is multi-band blending [3]. It is easy to implement and provides stable blending results. It blends the images at each level of a Laplacian pyramid, and merges them to give the result.

Perez et al [6] formulate image blending via a Poisson equation whose solution can be obtained by solving a large sparse linear system. Although this is mathematically elegant and provides perfect results when the intensity changes across the boundary is smooth, it is time consuming, especially for large images. It also suffers from bleeding artefacts when the intensity changes across the blending boundary is insufficiently smooth.

Agarwala [2] observes that the color offset between the original content and the blended content in the target region is piecewise smooth, allowing ready approximation of the whole offset field by a quadtree. This significantly reduces the number of variables in the linear system, accelerating blending.

Szeliski et al [21] further observe that if each image has a separate offset field represented by a low-dimensional spline, each offset field is everywhere smooth, not just piecewise smooth. As the spline has low dimensionality, the number of variables is further reduced.

To avoid solving linear equations, Farbman et al [5] instead use mean-value coordinates (MVC) to interpolate the smooth offset field from boundary differences. For a target region of fixed shape, these coordinates can be precomputed and reused for all frames of a video. Furthermore, this method is readily parallelizable, but since it approximates the Poisson formulation, it too suffers from bleeding artefacts.

In [4] Farbman et al observe that the key operations in MVC interpolation are convolution operations with large kernels; these can be approximated by several small kernels to further reduce computation.

Poisson blending can also be improved by adding an intensity constraint [7], as explained later (and henceforth referred to as the *modified Poisson* approach).

We analyze six representative blending algorithms, chosen for the following reasons. Feather blending has the lowest computational expense (apart from trivially clipping images at the seams), and provides a baseline of visual quality. Multi-band blending is the most widely used approach in the open source community [22], and is relatively insensitive to misalignment. MVC blending can be readily parallelized, and avoids large linear equations, while providing almost visually identical results to standard Poisson blending. Using a convolution pyramid approximates the MVC approach and further speeds it up. Multi-spline blending offers another strategy to approximate the original Poisson equation, resulting in a significantly smaller linear system. The differences in formulation of modified Poisson blending lead to visually different blending results.

We do not consider the original Poisson blending method, which is both slow and memory hungry, so unsuited to high resolution realtime video blending. We also do not consider the quadtree approximation to Poisson blending as it uses the smoothness of the offset map in a similar way to multi-spline blending, but the latter solves a smaller linear system.

B. Intensity Changes

Since different blending algorithms have different formulations, they affect the pixel intensities in the result in different ways. We briefly analyse their effects here, and schematically show the trends of pixel intensity changes produced by different blending algorithms in Figure 3 (real examples are given later).

Feather blending linearly blends the images in the overlapped regions, and other regions remain unchanged.

Multi-band blending blends the images everywhere at different frequencies, causing intensities to be averaged across the whole image.

Since MVC blending approximates Poisson blending, and convolution pyramid blending further approximates MVC blending, in these two algorithms, the regions to be blended change in intensity to fit the anchor region (the region whose pixel intensities remain unchanged).

Multi-spline blending uses splines to approximate the offset map, so lighting inconsistency is obvious along the boundary seams, especially if the input scenes are poorly aligned (see Figure 13, row 5).

Modified Poisson blending tries to preserve the original intensities as well as the gradient field of the blended region, so it produces rather different results to all the other algorithms.

In summary, MVC blending, modified Poisson blending and convolution pyramid blending are sensitive to choice of anchor stream, while feather blending, multi-band blending and multi-spline blending produce the same blending results given an arbitrary blending order. This implies that MVC blending, convolution pyramid blending and modified Poisson blending are not symmetric, treating their two input images differently, while feather blending, multi-band blending and multi-spline blending are symmetric. Lack of symmetry is acceptable for cut and paste applications, but is inappropriate when the two input images have the same status, as in panorama blending. They require a blending order to be chosen, and there is no direct way to choose a good blending order automatically. In our implementation, we manually choose a blending order.

C. Algorithms

We now detail the algorithms tested.

1) Feather Blending (FB): Feather blending simply linearly combines the regions to be blended, using:

$$P = \sum_{i=1}^{6} \omega_i P_i \tag{6}$$

where ω_i is a per pixel weight for each input stream. At each pixel, the weights of all streams sum to 1, so FB only affects areas where streams overlap. The simplest approach uses weights of 0.5 everywhere in the overlap. A better approach uses a weight of 0.5 at the seam, with the weight falling as we go nearer to the edge of the stream, until it becomes zero. As each pixel is processed independently, FB is fully parallelizable.

2) Multi-band Blending (MBB): In essence, multi-band blending combines feather blending results from versions of the images containing different frequencies. A Laplacian pyramid is built, and the regions to be blended are linearly combined at each level. The final result is obtained by merging the blended images from the different levels. The Laplacian pyramids can be constructed in parallel using equivalent weighting functions [3]. As each level of the pyramid can be regarded as a function of the original image, it is possible to precompute the function mapping between the input image and the other levels, allowing computation of each level of the pyramid simultaneously. Combination of the Laplacian images using a Gaussian weight image is also fully parallelizable. Multi-band blending can be defined as:

$$P = \sum_{j=1}^{l} U(Q_j), \tag{7}$$

where l is the number of layers in the pyramid, $U()$ up-samples an image to the original resolution, and Q_j is defined as:

$$Q_j = \sum_{i=1}^{6} G_i^j L_i^j. \tag{8}$$

Here, G_i^j is the i^{th} stream's Gaussian pyramid at level j, obtained by convolving a small weighting function (for example, a 5×5 filter) with the image, and L_i^j is the i^{th} stream's Laplacian pyramid at level l. We use 8 levels in our implementation.

3) MVC Blending (MVCB): MVC blending approximates the Laplacian membrane used in Poisson blending, constructing a harmonic interpolant from the boundary intensity differences. Unlike Poisson blending, which finds the final pixel values directly, MVC blending computes a color offset map; the final blended result is obtained by adding this offset map to the region to be blended: see Equation 3. For each pixel in the output region, the offset value is a weighted linear combination of all the boundary differences; the weights are derived from each pixel's mean value coordinate. The boundary is the outer boundary of the region to be blended. In detail, given a point x in the region to be blended, $P^*(x)$ is calculated by:

$$P^*(x) = \sum_{i=0}^{m-1} \lambda_i(x)\Delta(p_i), \tag{9}$$

where p_i is a pixel along the boundary of the region to be blended, $\Delta()$ is the pixel-wise difference operation between the two images (to blend P_a and P_b, changing P_b to fit P_a, $\Delta()$ computes $P_a - P_b$ along the blending boundary), and m is the number of boundary points. λ_i is the mean value coordinate of x with respect to the current boundary points—see [5]. As the boundary seams have fixed locations, the mean value coordinates and weights can be precomputed once, for all frames, saving effort in video blending. Since the value at each position of the offset map only depends on the boundary differences, MVCB is parallelizable.

4) Convolution Pyramid Blending (CPB): In MVCB, the final membrane (i.e. offset map) can be written as:

$$P^*(x) = \frac{\sum\limits_k w_k(x) b(x_k)}{\sum\limits_k w_k(x)}, \tag{10}$$

where x_k are boundary points, $b(x)$ are boundary values and $w_k(x)$ are corresponding mean value coordinates. As shown in [4], Equation 10 can be rewritten as a ratio of convolutions by incorporating a characteristic function $\chi_{\hat{P}}$ which is 1 where $\hat{P}$ is non-zero and 0 otherwise:

$$P^*(x_i) = \frac{\sum\limits_{j=1}^{6} w(x_i, x_j)\hat{P}(x_j)}{\sum\limits_{k=1}^{6} w(x_i, x_j)\chi_{\hat{P}}(x_j)} = \frac{w * \hat{P}}{w * \chi_{\hat{P}}}, \tag{11}$$

and $\hat{P}$ is an extension of the boundary b to the entire domain:

$$\hat{P}(x_i) = \begin{cases} b(x_k), & \text{if } x_i = x_k \\ 0, & \text{otherwise} \end{cases}. \tag{12}$$

Calculation of the offset map now involves convolutions with large filters. *Multiscale transforms* [4] allow these to be approximated by a set of smaller filters in linear time.

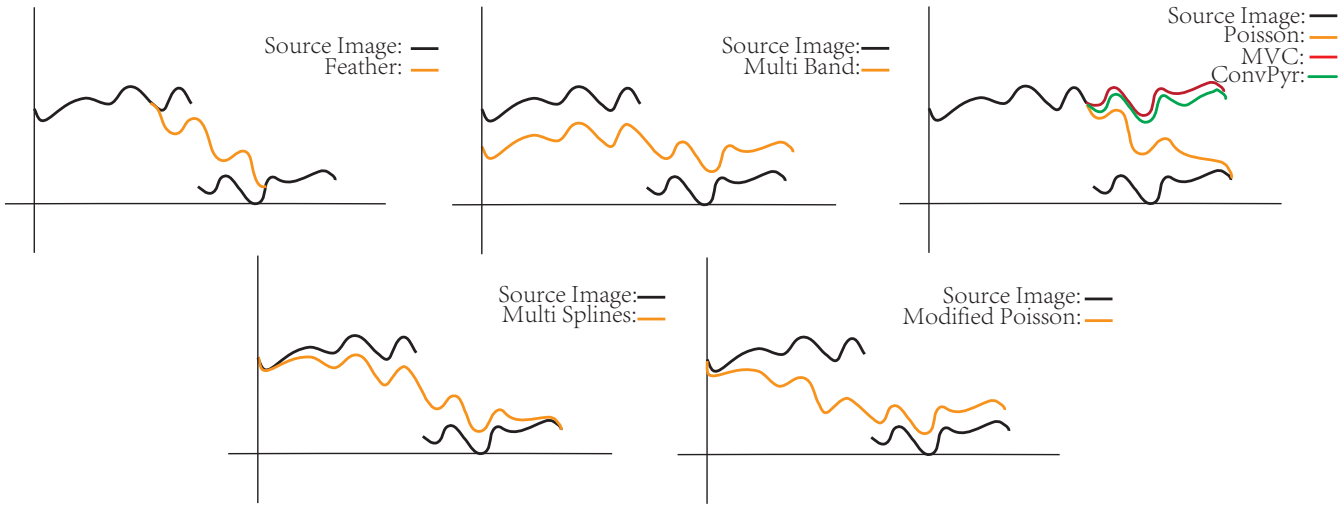


Fig. 3. One dimensional schematic illustration of results produced by different blending algorithms.

TABLE I
COMPUTATION TIMES (PER FRAME) AND MEMORY USAGE FOR 4000×2000 RESOLUTION VIDEO, FOR EACH OF THE SIX ALGORITHMS.

	FB	FB (GPU)	MBB	MBB (GPU)	MVCB	MVCB (GPU)	CPB	CPB (GPU)	MSB	MPB
Memory (MB)	498	428	841	2274	3303	3982	942	2380	2225	1295
Time (ms)	642	7	3029	25	2535	31	4782	63	7940	11322

5) Multi-Spline Blending (MSB): Poisson blending can be expressed in an energy minimization formulation [6]. The energy may be written in offset map form as:

$$E = \sum_{i,j} \left(P^{*l_{i+1,j}}_{i+1,j} - P^{*l_{i,j}}_{i,j} - \hat{g}^x_{i,j}\right)^2 + \left(P^{*l_{i,j+1}}_{i,j+1} - P^{*l_{i,j}}_{i,j} - \hat{g}^y_{i,j}\right)^2, \tag{13}$$

where $l_{i,j}$ indicates which stream each pixel comes from (as given by the mask information), (i,j) indicates the location in the image plane, and the (modified) gradient $\hat{g}^x_{i,j}$ is defined by:

$$\hat{g}^x_{i,j} = P'^{l_{i,j}}_{i,j} - P'^{l_{i+1,j}}_{i,j} + P'^{l_{i,j}}_{i+1,j} - P'^{l_{i+1,j}}_{i+1,j}. \tag{14}$$

Here, $P'^{l_{i,j}}_{i,j}$ is the pixel intensity at location (i,j) in the lth stream. The modified y gradient $\hat{g}^y_{i,j}$ is defined similarly. The energy E can be minimized by solving a linear system $Az = b$ where z represents the unknown pixel values in the offset map. By using spline cells to approximate the assumed-smooth offset map, each pixel in the final offset map can be represented as:

$$P^{*l}_{i,j} = \sum_{k,m} c^l_{k,m} B(i - kR, j - mR), \tag{15}$$

where R is the pixel spacing (we choose 64 in our experiment) of the spline cells, $B(i-kR, j-mR)$ is the spline basis, and the $c_{k,m}$ are the spline control points. In this way, the size of the linear system is reduced significantly.

6) Modified Poisson Blending (MPB): Tanaka et al [7] modified the original Poisson energy function by adding an intensity constraint:

$$E' = \sum_{i,j} \varepsilon(I_{i,j} - P_{i,j})^2 + (g_{i,j} - \nabla P_{i,j})^2, \tag{16}$$

where $I_{i,j}$ is the original pixel intensity at location (i,j), $P_{i,j}$ is the intensity of the final panorama at that location, ε is a weight, $\nabla P_{i,j}$ is the gradient of the final panorama, and $g = g_{i,j}$ is a gradient map computed by summing the gradients of each stream g^i:

$$g = \sum_{i=1}^{6} g^i M'_i \tag{17}$$

Unlike in the original Poisson blending approach, the coordinates (i,j) now range over the whole image, so that *all* streams change the pixel value. Tanaka et al [7] solve this equation in the frequency domain:

$$P^T_{i,j} = \frac{v^T_{i,j} - \varepsilon u^T_{i,j}}{d^T_{i,j} - \varepsilon}, \tag{18}$$

where $P^T_{i,j}$ is the discrete cosine transform (DCT) of each pixel in the final panorama, $v^T_{i,j}$ is the DCT of the Laplacian of the image (found by combining the Laplacians of each stream using Equation 17), $u^T_{i,j}$ is the DCT of the original intensity image, and $d^T_{i,j}$ is the DCT of the Laplacian operator. The final panorama is obtained by computing the inverse DCT of $P^T_{i,j}$. In our implementation ε was set to 1e-8.

D. Requirements for Overlapping Areas

The above blending algorithms have different requirements for overlapping regions. FB only blends pixels in overlapping regions. Thus, no blending is performed if there is no overlap. MBB requires the images to be blended to completely overlap if we are to use a complete pyramid. In our implementation we achieve this by mirroring each input stream to fill the whole panorama. Since MVCB and CPB are approximations of Poisson blending, these two algorithms require pixel intensity

differences along the blending boundary, so there must be at least a one-pixel wide overlap along the boundary. The only change MPB makes is to add a color constraint to the original Poisson blending, so it also only requires a one-pixel wide overlap along the boundary. MSB requires a two pixel-wide overlap along the boundary since it diffuses boundary differences in two opposite directions.

E. GPU Acceleration

As already noted, FB, MBB, MVCB and CPB can be easily parallelized. As MSB needs to solve an optimization problem represented by a linear system for each high resolution video frame, even a parallel solver [23] is still time-consuming. For MBB and CPB, convolution is the most time-consuming operation. In our implementation, accessing GPU memory takes most of the time, as each pixel is accessed c^2 times, where c is the kernel size. We take full advantage of GPU shared memory, which is much faster than ordinary GPU memory, by decomposing the 2D convolution into 2 1D convolutions, vertically then horizontally. Using this strategy, each pixel is accessed only c times. In our implementation, due to a limited amount of shared memory for each thread, we perform this computation 256 pixels at a time.

IV. Experiments

Our experiments were performed on a PC with an Intel Xeon E5-2620 2.0GHz CPU with 32GB memory, and an NVIDIA GTX 970 GPU with 4GB memory; the bandwidth between PC memory and GPU memory was 4GB/s. The blending algorithms were implemented in C++, while GPU implementations used CUDA.

A. Performance

1) Theory: We initially considered the theoretical time complexity of these 6 representative algorithms. Since feather blending only computes a linear combination for each pixel, its complexity is $O(n)$ where n is the number of pixels. Multi-band blending also has complexity $O(n)$, as the extra levels only multiply the number of pixels to process by a constant factor. MVC blending requires target region triangulation and adaptive boundary sampling, with $O(m)$ cost for evaluating the membrane, where m is the number of pixels along the boundary; this is typically $O(\sqrt{n})$. Since the last step interpolates the membrane values to all n pixels, the total cost is again $O(n)$. Convolution pyramid blending uses small kernels to approximate a large kernel, so its complexity is again $O(n)$. Multi-spline blending needs to solve an $O(n/s^2)$ linear system where s is the sampling space of the spline, which also has complexity asymptotically $O(n)$. Modified Poisson blending finds pixels in the frequency domain with complexity $O(n\log(n))$.

2) Practice: We experimentally measured the time required by each blending algorithm, per output frame, as well as the memory used, for output videos of size 4000×2000 pixels. Note that for all algorithms, the time and memory costs only depend on the resolution of the input videos and the shape of the mask, and not on the video content, so one scene sufficed for this experiment. I/O times as well as precomputation times were not considered, as we are interested in how suitable each method is for continuous realtime operation. Table I gives the results, both for CPU implementation, and where appropriate, GPU implementation. They show that, when using a GPU with sufficient memory, multi-band blending, MVC blending, feather blending and convolution pyramid blending are fast enough for realtime performance.

B. Visual quality

We also both objectively and subjectively evaluated the blended videos produced by these algorithms.

1) Objective evaluation: Image and video quality assessment methods can be classified into full-reference and no-reference approaches [24], [25]. Full-reference approaches such as PSNR (peak signal-to-noise ratio) and SSIM (structural similarity index) require an original image or video as a reference, but in video blending there is no such ground truth. Thus, our objective evaluation for blended videos relies on no-reference approaches. We used 6 state of the art no-reference video quality metrics: BIQI [26], BRISQUE [27], FRIQUEE [28], NIQE [29], SSEQ [30] and VIIDEO [31]. Each metric was applied to assess the quality of the 42 blended videos produced by 7 different algorithms (the 6 video blending algorithms tested plus simple stitching without blending). The resulting objective scores were statistically analysed. For each metric, an analysis of variance (ANOVA) was performed by selecting the predicted quality as the dependent variable, and the blending algorithm as the independent variable. ANOVA aims to analyse the differences among group means, providing a statistical test of whether or not the means of several groups are equal. The ANOVA results show that, for each case, the algorithm has no statistically significant effect on video quality ($P>0.05$ at the 95% confidence level). This means that combining videos with a sophisticated blending algorithm is no better than stitching without blending. These findings are clearly contradictory to a simple visual assessment which makes it obvious that the stitched videos have lower quality. Unfortunately, this implies that the video quality metrics are not adequate for the kinds of artefacts arising in blending—indeed, they were mainly devised to assess defects due to compression and transmission of video. This finding is in itself an interesting topic for future research. However, for the topic investigated here, problem-specific objective evaluations as well as subjective quality assessment were required to determine the effect of different blending algorithms on video quality.

2) Temporal coherence: Blending video is unlike blending still images, in that temporal coherence largely contributes to the overall blended video quality. Inspired by [32], we calculate the temporal coherence score of the blended video as:

$$\sum_n \sum_x \left\| P^n(x) - \mathrm{warp}(P^{n-1}(x)) \right\|^2. \tag{19}$$

Here x represents the spatial location of the frame, P^n is the current frame and P^{n-1} is the previous frame. $\mathrm{warp}()$ is a

TABLE II
TEMPORAL COHERENCE SCORES FOR 6 BLENDING ALGORITHMS.

	MVCB	FB	CPB	MSB	MBB	MPB
Mean	552	320	566	362	316	373
Std. Dev.	265	402	242	409	378	387

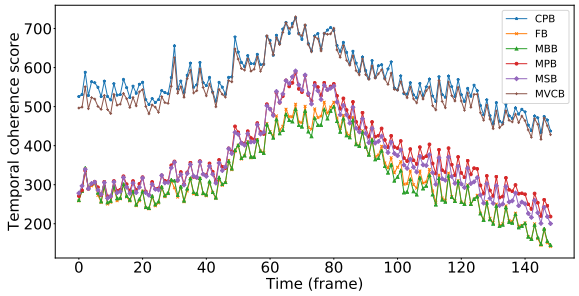


Fig. 4. temporal coherence score over time for each blending algorithm over all the scenes.

mapping given by the optical flow [33] between the previous frame and the current frame. For each blending algorithm, we calculate the mean and standard deviation of the temporal coherence score for all scenes. The temporal coherence score for each scene is divided by the number of frames. We show the results in Table II. Lower temporal score indicates better temporal coherence. Not surprisingly, MVCB and CPB are worst at providing temporal coherence. Since the blended result of these two Poisson blending approximations largely depends on boundary differences, even a subtle change at the boundary will propagate to the entire blended region. In practise, misalignment and noise are always present in the blending boundary, resulting in boundary differences which temporally vary in an unsmooth manner. MSB and MPB provide better temporal coherence than MVCB and CPB but worse than FB and MBB. In MSB only vertex intensities are calculated from the boundary differences, and the intensity of the pixels inside each cell are obtained by interpolation of the corresponding vertices. Thus, MSB has a smoother offset map than MVCB and CPB. Compared to the original Poisson blending formulation, the added color constraint helps to ensure temporal coherence. Among the 6 blending algorithms tested, MBB and FB have best temporal coherence. FB computes a linear blend in the overlapping region while MBB can be regarded as a multi-frequency linear blending algorithm. Since MBB blends more regions as well as at more frequencies, the temporal coherence of its is slightly better than that of FB. We also demonstrate the temporal coherence score per frame over time for each blending algorithm, for all scenes in Figure 4. We picked 150 continuous frames and calculated the temporal coherence scores, plotting the 149 scores in different colours for different algorithms.

3) Bleeding: Poisson image blending and its variants have elegant mathematical definitions but suffer from bleeding artefacts when the blending boundary is not smooth. This artefacts is manifest as a particular color leaking into its surroundings, as illustrated in Figure 5(a). Since MVCB, CPB, MSB and MPB are all variants of Poisson image blending, we assess its significance using the bleeding degree proposed in [34] to evaluate results for all scenes in our dataset. To calculate the bleeding map we first calculate an offset map by subtracting the original image from the blended image. An energy map (illustrated in Figure 5(b)) is given by the absolute values of the offset map, and the bleeding map is computed from the energy map by:

$$B(x) = \max(0, x - \alpha \frac{E_h}{A_h + \delta}), \tag{20}$$

where A_h is the number of non-zero values in the binarized energy map (binarized by Otsu's method [35]), and E_h is the sum of the energies at the non-zero positions in the binarized map. α is a weight, set to 2 in our evaluation, used to truncate high peak values, and δ is set to 1e-8. An example of bleeding map is shown in Figure 5(c). Given the bleeding map, the bleeding degree, i.e. the total amount of bleeding per frame is given by

$$V^n = \sum B(x)^2. \tag{21}$$

This quantity is averaged over all frames, for all scenes in our dataset. The results for MVCB, CPB, MSB and MPB are shown in Table III. Clearly, MSB and MPB suffer much less from bleeding artefacts than MVCB and CPB. MVCB and CPB are particularly affected when there the blending boundary is not smooth. For MSB, spline cells are used to approximate the offset map, which largely alleviates the bleeding artefacts. MPB not only considers gradients, but also the colors, so it too suffers less from bleeding artefacts.

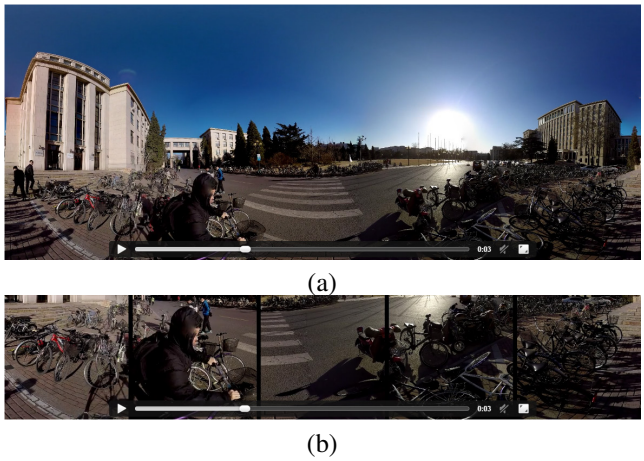
4) Subjective evaluation: We did not use all scenes in subjective testing (given that there are 6 algorithms for comparison): to avoid fatigue, we limited the assessment performed by each observer to last under 20 minutes. We thus used

TABLE III
BLEEDING DEGREES FOR DIFFERENT ALGORITHMS.

	MVCB	CPB	MSB	MPB
Mean	76.0	68.1	18.5	19.2
Std. Dev.	68.1	78.7	32.4	23.6

(a) (b) (c)

Fig. 5. (a) A blended result using MVCB (cropped from the panorama). (b) Energy map. (c) Bleeding map.



(a)

(b)

Fig. 6. (a): Output blended video. (b): Local seam regions.

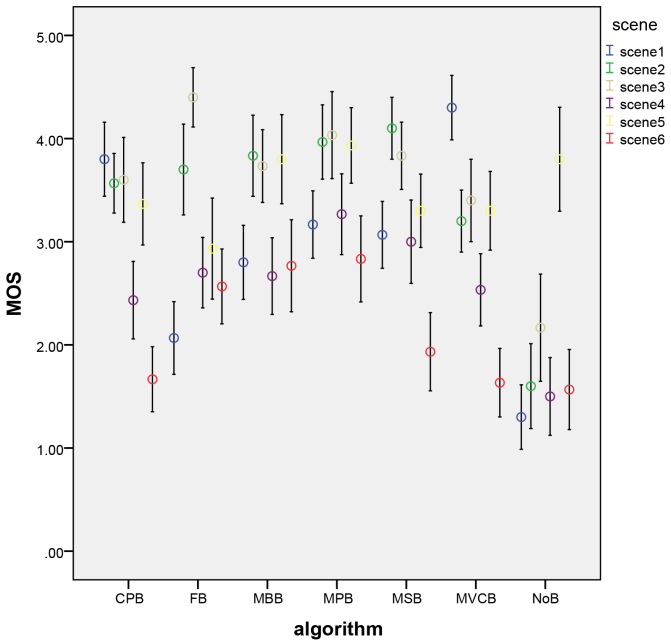


Fig. 9. Mean opinion score (MOS) averaged over all participants, for each blended video and each algorithm, in our subjective quality experiment. The vertical axis gives the MOS, with error bars indicating a 95% confidence interval.

6 representative scenes from our benchmark for subjective evaluation.

The subjective video quality assessment followed the guidelines described in [36]. The experiments were carried out in a standard office environment. The output videos (stimuli) were displayed on a 24-inch LCD with a native resolution of 3840×2160 pixels. The viewing distance was approximately 50 cm. A single-stimulus method was adopted, so that subjects had to score the overall quality for each stimulus in the absence of any reference. The ITU-R absolute category rating (ACR) 5-point scale (i.e., 1 = Bad, 2 = Poor, 3 = Fair, 4 = Good, 5 = Excellent) was used for quality scoring. For the 6 test videos, we picked 10 s from each scene and used the results of 7 different algorithms (the above 6, plus simple stitching without blending) as stimuli, giving each participant 420 seconds of video to view. Since artefacts mainly appear near seams, we also cropped 5 square regions around the seams between different streams for each scene. We manually picked five 300×300 local regions containing the most moving objects; the areas of the subregions divided by the seam are approximately in correspondence. During the subjective assessment, the whole blended video was presented to subjects to evaluate the overall blending quality, while the local seam regions were presented to evaluate the artefacts caused by the blending algorithms. Thus, the total time of videos to be viewed was 840 seconds, so an entire subjective assessment could be done in under 20 minutes. The interface for subjective assessment is illustrated in Figure 6. The participants in the study consisted of two groups of people. The first group comprised university students, 18 male and 12 females who were inexperienced with video quality assessment. The second group comprised 3 experts in image processing: one university faculty member, one postdoctoral associate and one Ph.D. student.

Before the start of each experiment, written instructions concerning the experimental procedure, including the scoring scale and timing, were given to each subject. A training set of several typical scenes and their blending results was presented to the participants in order to familiarise them with the issues in visual blending quality and with use of the scoring scale. These were annotated with an expert's remarks on the video such as "there is an obvious seam and the color is not very consistent near it" or "there is flickering around the moving object". Several kinds of artefact were also described. The stimuli used in training were different from those used for assessment. After training, the scenes showing the results of different blending algorithms were presented to each participant in a random order. Each stimulus was shown once, and the participants were requested to assess its quality immediately after viewing it. Figure 7 illustrates scenes 1–6 used in our experiment. Figures 11–13 show some scenes and the results of the 6 video blending algorithms applied to each scene. Blending result on local seam regions of different algorithms are illustrated in Figure 8.

Results of the experiment were filtered [37] to reject outlier evaluations and individuals. Results more than two standard deviations from the mean score for a test were considered to be outliers; an individual was an outlier if 1/3 of scores submitted were outliers. This caused one participant to be rejected. After data filtering, the remaining scores were analysed. Figure 9 gives the mean opinion score (MOS) in our subjective experiment, averaged over all participants, for each blended video. It shows that both video content and blending algorithm affect the overall perceived blending quality.

The observed tendencies were statistically analysed with a full factorial ANOVA using subjective quality as the dependent variable, the video content and blending algorithm as fixed independent variables, and the participant as the random independent variable. Two-way interactions of the fixed variables were included in the analysis. The results are summarised in Table IV, where the F-statistic and its associated degrees of freedom and p-value are reported for each variable. These

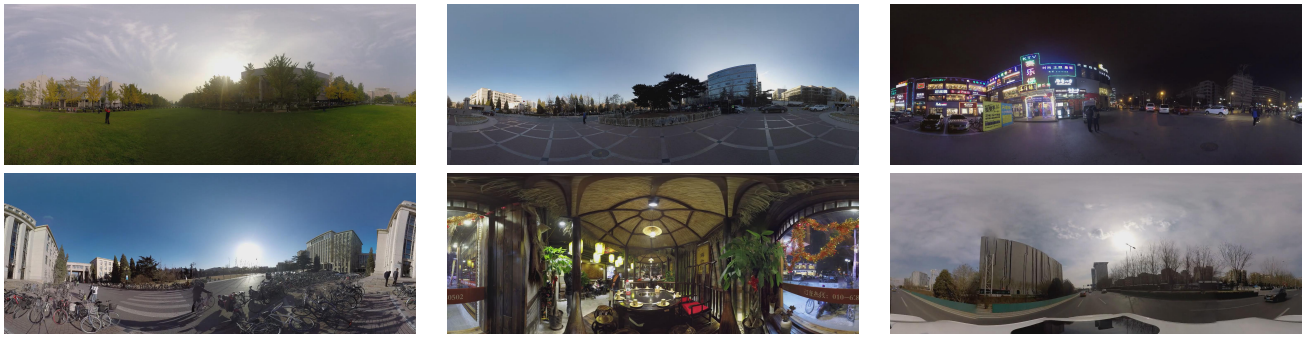


Fig. 7. From left to right, top to bottom: scenes 1–6 used in our evaluation.

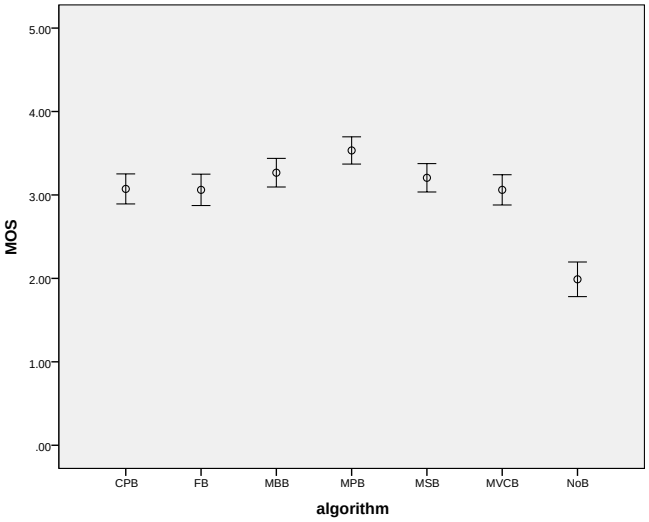


Fig. 10. Mean opinion score (MOS) averaged over all participants and all blended videos for each algorithm, in our subjective quality experiment. The vertical axis gives the MOS, with error bars indicating a 95% confidence interval.

results indicate that video content and blending algorithm have a statistically significant effect on perceived quality. Not all participants have the same average rating of quality, which is normal in a typical video quality assessment experiment, as human observers can have different internal scales.

The interaction between video content and blending algorithm is significant, which implies that the impact the different blending algorithms have on video quality depends on the video content. Not all output scenes have the same overall quality. The test reveals the output scenes have the following order of quality (qualities of jointly underlined scenes do not significantly differ from each other): scene6 $<$ scene4 $<$ scene1 $<$ scene2 $<$ scene5 $<$ scene3. These differences may be because different scenes have different levels of interactions between blending artefacts and visual content, consequently affecting the visibility of artefacts in different scenes.

There is clearly a significant difference in quality between the 6 blending algorithms. The analysis reveals the following order in quality (jointly underlined algorithms do not significantly differ): No-blending $<$ MVCB $<$ FB $<$ CPB $<$ MSB $<$ MBB $<$ MPB. Figure 10

TABLE IV
ANOVA TEST RESULTS EVALUATING THE EFFECT OF VIDEO CONTENT AND BLENDING ALGORITHMS ON THE QUALITY OF VIDEO BLENDING.

	F-value	DoF	Significance
Participant	2.524	29	< 0.01
Scene content	71.534	5	< 0.01
Blending algorithm	43.174	6	< 0.01
Scene content * Blending algorithm	9.577	30	< 0.01

and Table V summarises the impact of the 6 different algorithms as well as no blending on video quality, averaged over all scenes. The subjective evaluation shows that using a blending algorithm improves output video quality relative to stitching videos without blending, and that MPB outperforms other algorithms in terms of final video quality. Considering the standard deviations in Table V, MPB also provides the most consistent quality of video results across all test scenes and all participants.

When considering local seam regions, stitching again has the lowest quality while CPB, MSB and MPB have the highest quality. Comparing the mean scores for the whole video and the local seam regions allows us to draw several conclusions. MPB provides the highest quality in both global and local evaluation. Stitching has the lowest quality globally, and second lowest quality in local evaluation, for which FB is worst. This is because several scenes in the evaluation have not been well aligned. Such misalignments lead to ghosting artefacts, which leads to a very poor viewing experience when concentrating on local windows. MVCB and CPB do poorly in the global evaluation but rather higher in a local evaluation, as the bleeding artefacts of these Poisson approximations have greater effects on a large scale than locally.

V. Conclusions

We have compared the performance and visual quality of 6 blending algorithms when used for realtime 4K video blending for a variety of scenes. Simple approaches such as FB and MBB are fast when implemented on a GPU, but do not produce high quality blending results. The main problem with MVCB and CPB is that they are too sensitive to boundary conditions, and suffer from bleeding artefacts. MSB suffers less from bleeding than MVCB and CPB, but obvious lighting inconsistencies are visible in the output when the input video

TABLE V
MEAN AND STANDARD DEVIATION OF THE QUALITY SCORE AVERAGED OVER ALL SCENES AND ALL PARTICIPANTS FOR EACH BLENDING ALGORITHM.

		No-blending	MVCB	FB	CPB	MSB	MBB	MPB
Whole video	Mean	1.97	3.04	3.05	3.07	3.19	3.28	3.52
	Std. Dev.	1.40	1.23	1.29	1.21	1.16	1.17	1.12
Local seam region	Mean	1.94	3.08	2.82	3.21	3.21	3.07	3.21
	Std. Dev.	1.38	1.08	1.19	1.25	1.12	1.23	1.14

streams are not well aligned. Our experiments show that modified Poisson blending performs surprising well on various scenes, but it is not as efficient as some other approaches. This suggests that further work to improve the efficiency of modified Poisson blending would be useful, making it more practical in real world applications.

Hopefully, further efficient blending algorithms will become available in future, with improved capabilities and speed. We have made our video benchmark and code implementing each method publicly available to facilitate further evaluation of new algorithms in this field.

A side-result of this work is that current objective quality assessment algorithms and metrics are unsuitable for application to the results of video blending, having been devised to evaluate the results of compression and transmission errors, and work is needed to devise new metrics suited to assessing results of blending.

Although the objective evaluation results are not exactly the same as subjective evaluation results, they are highly consistent. In future work we hope to combine several objective evaluation results so as to approximate human judgement. A potential way forward is to explicitly incorporate brightness consistency between the blended regions as well as several other kinds of artefacts as evaluation terms in a new metric.

ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation of China (Project Number 61561146393, 61521002), Research Grant of Beijing Higher Institution Engineering Research Center and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

REFERENCES

[1] Z. Zhu, H. Z. Huang, Z. P. Tan, K. Xu, and S. M. Hu, "Faithful completion of images of scenic landmarks using internet images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 8, pp. 1945–1958, Aug 2016.

[2] A. Agarwala, "Efficient gradient-domain compositing using quadtrees," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 94, 2007.

[3] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics*, vol. 2, no. 4, pp. 217–236, 1983.

[4] Z. Farbman, R. Fattal, and D. Lischinski, "Convolution pyramids," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 175:1–175:8, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2070781.2024209

[5] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski, "Coordinates for instant image cloning," *ACM Transactions on Graphics*, vol. 28, no. 3, p. 67, 2009.

[6] P. Perez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics (SIGGRAPH'03)*, vol. 22, no. 3, pp. 313–318, 2003.

[7] M. Tanaka, R. Kamio, and M. Okutomi, "Seamless image cloning by a closed form solution of a modified poisson problem," in *SIGGRAPH Asia 2012 Posters*, ser. SA '12. New York, NY, USA: ACM, 2012, pp. 15:1–15:1. [Online]. Available: http://doi.acm.org/10.1145/2407156.2407173

[8] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz, "Jump: Virtual reality video," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 198:1–198:13, Nov. 2016. [Online]. Available: http://doi.acm.org/10.1145/2980179.2980257

[9] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, and R. Szeliski, "Low-cost 360 stereo photography and video capture," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 148:1–148:12, Jul. 2017.

[10] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang, S. Watson, and M. Gross, "Panoramic video from unstructured camera arrays," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 57–68, May 2015. [Online]. Available: http://dx.doi.org/10.1111/cgf.12541

[11] H. Guo, S. Liu, T. He, S. Zhu, B. Zeng, and M. Gabbouj, "Joint video stitching and stabilization from moving cameras," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5491–5503, Nov 2016.

[12] M. Wang, J. B. Liang, S. H. Zhang, S. P. Lu, A. Shamir, and S. M. Hu, "Hyper-lapse from multiple spatially-overlapping videos," *IEEE Transactions on Image Processing*, vol. PP, no. 99, pp. 1–1, 2017.

[13] Q. Zhi and J. R. Cooperstock, "Toward dynamic image mosaic generation with robustness to parallax," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 366–378, Jan 2012.

[14] K. Lin, S. Liu, L.-F. Cheong, and B. Zeng, "Seamless video stitching from hand-held camera inputs," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 479–487, May 2016. [Online]. Available: https://doi.org/10.1111/cgf.12848

[15] J. Lee, B. Kim, K. Kim, Y. Kim, and J. Noh, "Rich360: Optimized spherical representation from structured panoramic camera arrays," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 63:1–63:11, Jul. 2016. [Online]. Available: http://doi.acm.org/10.1145/2897824.2925983

[16] J. Jia, J. Sun, C.-K. Tang, and H.-Y. Shum, "Drag-and-drop pasting," *ACM Transactions on Graphics*, 2006.

[17] M. W. Tao, M. K. Johnson, and S. Paris, "Error-tolerant image compositing," *International Journal of Computer Vision*, vol. 103, no. 2, pp. 178–189, 2013.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000029664.99615.94

[19] J. Zaragoza, T.-J. Chin, Q.-H. Tran, M. S. Brown, and D. Suter, "As-projective-as-possible image stitching with moving dlt," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1285–1298, Jul. 2014. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2013.247

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, vol. 8, no. 19, pp. 415–428, 2012.

[21] R. Szeliski, M. Uyttendaele, and D. Steedly, "Fast poisson blending using multi-splines," in *International Conference on Computational Photography (ICCP 11)*. IEEE, April 2011.

[22] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, Aug 2007. [Online]. Available: https://doi.org/10.1007/s11263-006-0002-3

[23] C. Wu, M. Zollhöfer, M. Nießner, M. Stamminger, S. Izadi, and C. Theobalt, "Real-time shading-based refinement for consumer depth cameras," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 200:1–200:10, Nov. 2014. [Online]. Available: http://doi.acm.org/10.1145/2661229.2661232

[24] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, Jan 2009.

[25] "Perceptual visual quality metrics: A survey," *Journal of Visual Communication and Image Representation*, vol. 22, no. 4, pp. 297 – 312, 2011.

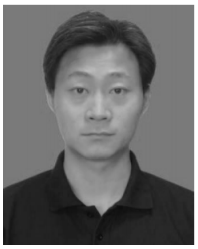
[26] A. K. Moorthy and A. C. Bovik, "A two-step framework for constructing blind image quality indices," *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 513–516, May 2010.

[27] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference image quality assessment in the spatial domain," *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4695–4708, Dec 2012.
[28] D. Ghadiyaram and A. C. Bovik, "Perceptual quality prediction on authentically distorted images using a bag of features approach," *CoRR*, vol. abs/1609.04757, 2016. [Online]. Available: http://arxiv.org/abs/1609.04757
[29] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a completely blind image quality analyzer," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209–212, March 2013.
[30] L. Liu, B. Liu, H. Huang, and A. C. Bovik, "No-reference image quality assessment based on spatial and spectral entropies," *Signal Processing: Image Communication*, vol. 29, no. 8, pp. 856 – 863, 2014.
[31] A. Mittal, M. A. Saad, and A. C. Bovik, "A completely blind video integrity oracle," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 289–300, Jan 2016.
[32] N. Bonneel, J. Tompkin, K. Sunkavalli, D. Sun, S. Paris, and H. Pfister, "Blind video temporal consistency," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 196:1–196:9, Oct. 2015. [Online]. Available: http://doi.acm.org/10.1145/2816795.2818107
[33] D. Sun, S. Roth, and M. J. Black, "A quantitative analysis of current practices in optical flow estimation and the principles behind them," *Int. J. Comput. Vision*, vol. 106, no. 2, pp. 115–137, Jan. 2014. [Online]. Available: http://dx.doi.org/10.1007/s11263-013-0644-x
[34] M. Wang, Z. Zhu, S. Zhang, R. Martin, and S. Hu, "Avoiding bleeding in image blending," in *IEEE International Conference on Image Processing (ICIP)*, Sept 2017.
[35] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan 1979.
[36] "Methodology for the subjective assessment of the quality of television pictures," *ITU-R Recommendation BT.500-13.*, Jan. 2012.
[37] H. Liu, N. Klomp, and I. Heynderickx, "A no-reference metric for perceived ringing artifacts in images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 4, pp. 529 – 539, 2010.

Zhe Zhu is currently a Postdoc Associate in Duke University. He got his Ph.D. in the Department of Computer Science and Technology, Tsinghua University in 2017. He received his bachelor's degree in Wuhan University in 2011. His research interests are in computer vision and computer graphics.

Jiaming Lu is a Ph.D. student in the Department of Computer Science and Technology, Tsinghua University. His research interests are in computer vision and fluid simulation.

Minxuan Wang is a master student in the Department of Computer Science and Technology, Tsinghua University. His research interests are in computer vision and deep learning.



Songhai Zhang obtained his Ph.D. in 2007 from Tsinghua University. He is currently an associate professor of computer science at Tsinghua University, China. His research interests include image and video processing, geometric computing.

Ralph Martin is an Emeritus Professor of Cardiff University.

Hantao Liu received the Ph.D. degree from the Delft University of Technology, Delft, The Netherlands in 2011. He is currently an Assistant Professor with the School of Computer Science and Informatics, Cardiff University, Cardiff, U.K. He is serving for the IEEE MMTC, as the Chair of the Interest Group on Quality of Experience for Multimedia Communications, and he is an Associate Editor of the IEEE Transactions on Human-Machine Systems and the IEEE Transactions on Multimedia.

Shi-Min Hu received a Ph.D. degree from Zhejiang University in 1996. He is currently a professor in the Department of Computer Science and Technology, Tsinghua University, China. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He is Associate Editor-in-Chief of The Visual Computer, and an Associate Editor of IEEE Transactions on Visualization and Computer Graphics, Computers & Graphics and Computer Aided Design.

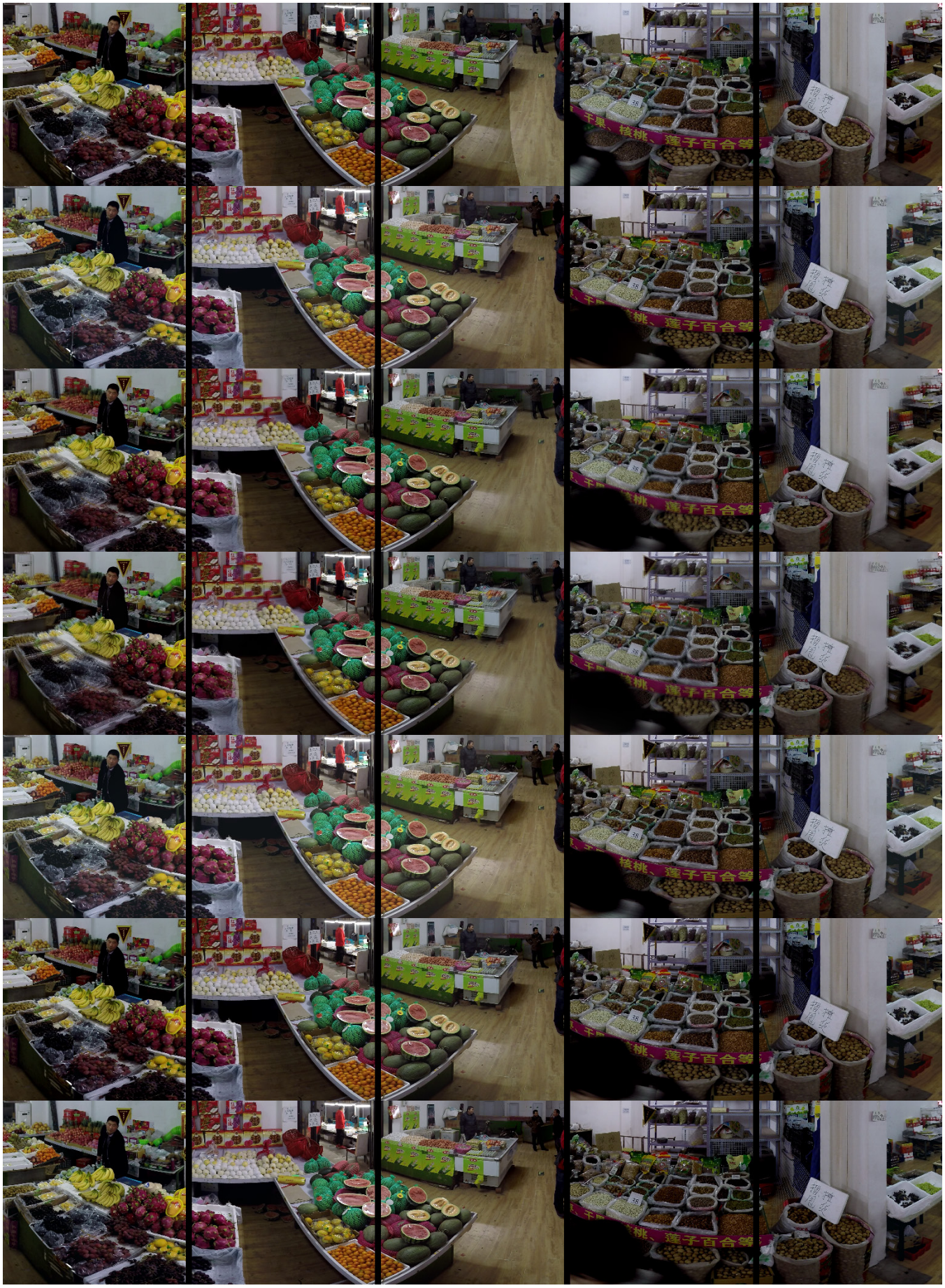


Fig. 8. Blending result on local seam regions. From top to bottom: blending results using no-blending, MVCB,FB, MBB, CPB, MSB and MPB.

Fig. 11. An outdoor scene with obvious illuminance differences for different streams. Top row: results of feather blending and multi-band blending; middle row: results of MVC blending and convolution pyramid blending; bottom row: results of multi-spline blending and modified Poisson blending.

Fig. 12. An indoor scene. Top row: results of feather blending and multi-band blending; middle row: results of MVC blending and convolution pyramid blending; bottom row: results of multi-spline blending and modified Poisson blending.

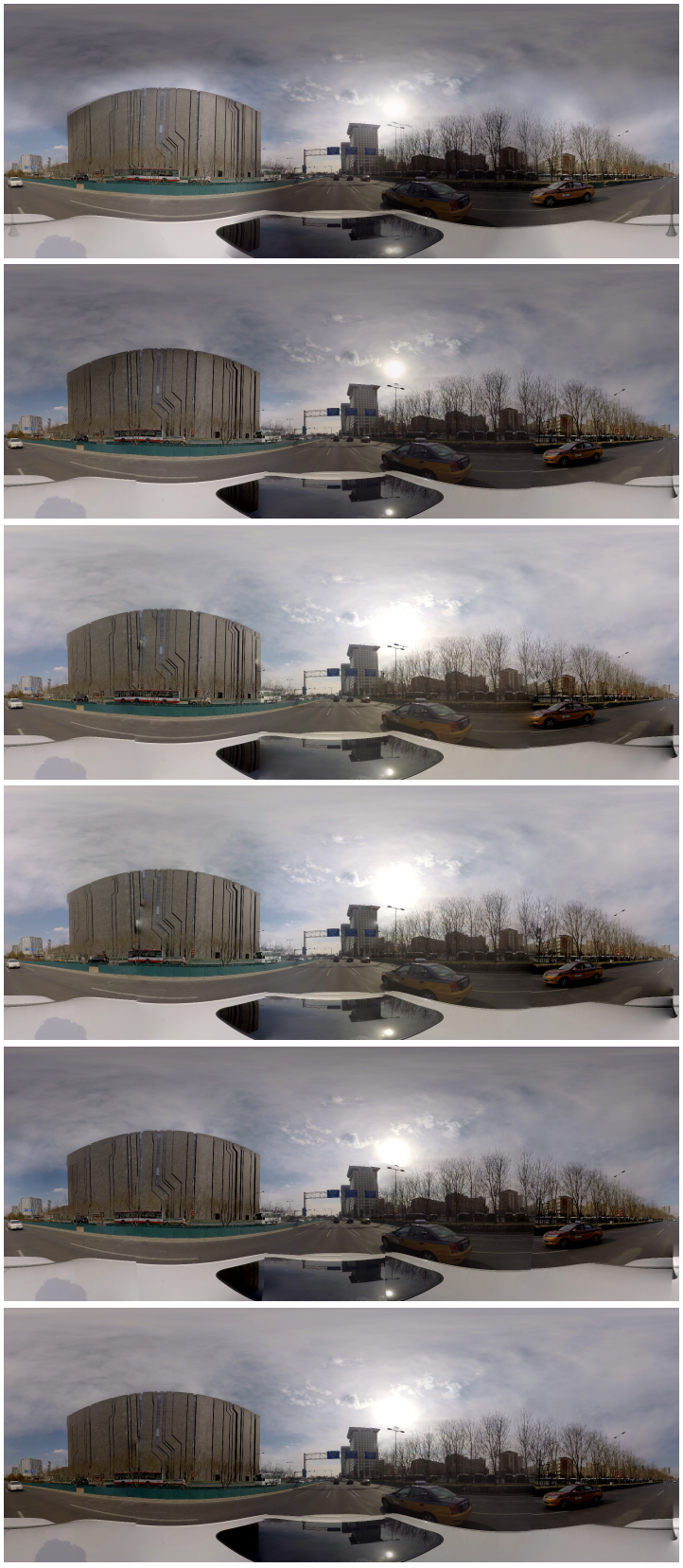


Fig. 13. A challenging scene taken with a moving camera rig. Top to bottom: results of feather blending, multi-band blending, MVC blending, convolution pyramid blending, multi-spline blending and modified Poisson blending.