Computational Zoom: A Framework for Post-Capture Image Composition

ABHISHEK BADKI, University of California, Santa Barbara and NVIDIA
ORAZIO GALLO, NVIDIA
JAN KAUTZ, NVIDIA
PRADEEP SEN, University of California, Santa Barbara

(a) $f = 16mm$ (b) $f = 35mm$ (c) $f = 105mm$ (d) **Computational Zoom**

Fig. 1. The interplay of camera position and focal length allows a photographer to achieve different compositions of the same scene. Moving the camera away from the scene while increasing the focal length f affects the sense of depth of the scene, as well as the relative magnification of objects at different depths, see (a) through (c). Note that the woman did not move while these three pictures were being taken. Given a stack of images captured with a fixed focal length at different distances from the scene, our framework allows us to modify the composition of the scene in *post-capture*, and leverages multi-perspective cameras for added flexibility. This is shown in the **animation** in (d), which can be viewed by clicking on it in a media-enabled PDF viewer, such as Adobe Reader. The colors overlaying the key-frames of the animation indicate regions imaged with different focal lengths, shown in the visualization on the right. The static frame shows the final composition and the multi-perspective camera used to render it.

Capturing a picture that "tells a story" requires the ability to create the right composition. The two most important parameters controlling composition are the camera position and the focal length of the lens. The traditional paradigm is for a photographer to mentally visualize the desired picture, select the capture parameters to produce it, and finally take the photograph, thus committing to a particular composition. We propose to change this paradigm. To do this, we introduce *computational zoom*, a framework that allows a photographer to manipulate several aspects of composition in post-processing from a stack of pictures captured at different distances from the scene. We further define a multi-perspective camera model that can generate compositions that are not physically attainable, thus extending the photographer's control over factors such as the relative size of objects at different depths and the sense of depth of the picture. We show several applications and results of the proposed *computational zoom* framework.

CCS Concepts: • **Computing methodologies** → **Reconstruction**; **Computational photography**; **Image-based rendering**;

This work was supported in part by the National Science Foundation under grants #13-21168 and #16-19376.

DOI: http://dx.doi.org/10.1145/3072959.3073687

ACM Reference format:
Abhishek Badki, Orazio Gallo, Jan Kautz, and Pradeep Sen. 2017. Computational Zoom: A Framework for Post-Capture Image Composition. *ACM Trans. Graph.* 36, 4, Article 46 (July 2017), 14 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073687

1 INTRODUCTION

Pictures can be powerful devices of communication when their composition allows the photographer to tell the right story. By *composition*, we refer to the process of adjusting the relative positions and sizes of foreground and background objects in the image. Photographers traditionally accomplish this by varying capture parameters such as the camera position and the focal length of the lens. When skillfully employed, the interplay of these parameters affects the resulting image in fundamental ways. One example of this is shown in Fig. 1(a-c), where the foreground subject is standing in the exact same position, but the composition (and therefore the "story") of each image is fundamentally different. In this case, this was done by increasing the focal length and moving back the camera.

Figure 1(a) shows that a short focal length provides a large field of view (FOV), thereby capturing a large portion of the background. In this case, the image provides a better context of the scene in which the person is standing. Furthermore, because the camera is close to the subject, there is more perspective distortion (also

known as *extension distortion*[1]) on the foreground, making it look more three-dimensional. Depending on the situation, this may or may not be a desirable quality.

On the other hand, switching to a longer focal length magnifies the objects in the scene. If the camera is then moved backwards to compensate for the magnification of the foreground subject, the scene appears more shallow, as in Fig. 1(c). Here, the rusty bridge in the background appears closer to the subject, affecting the story the picture tells. Also, because the camera is farther away, the image of the woman is not distorted and appears flatter.

The effect of the interplay of camera position and focal length has long been exploited in still photography and film. For example, a popular cinematographic technique that leverages this is the *dolly zoom*, also known as the "Vertigo" effect. A dolly zoom is a video sequence that is made more dramatic by collapsing or expanding the apparent depth of the scene around the main subject, often during an important, perspective-altering event. This effect is achieved by moving the camera in or out of the scene (a motion called *dollying*), while the focal length is progressively changed so as to maintain the magnification of the main subject.

However, adjusting the composition of the image by manually changing the camera position and focal length appropriately presents several difficulties. First of all, varying these parameters as needed may be impractical or demand advanced skills. For instance, dolly zoom sequences require either specialized devices that change the focal length to exactly compensate for the camera motion, or a skilled camera crew of several people to execute the maneuver. Additionally, the range of focal lengths covered by the lenses at hand may be limited, thereby reducing the range of compositions that can be captured. Furthermore, the photographer is limited to compositions that are physically possible. Finally, but perhaps most importantly, any adjustments to the composition using these capture parameters must be decided by the photographer *before* capturing the sequence.

To address these problems, we present *computational zoom*, a framework that allows a photographer to adjust composition as a post-process. At capture time, we simply require the photographer to take a dolly-in video (or a stack of pictures) without changing the focal length of the lens—thus supporting widespread consumer devices such as camera phones. After capture, our computational zoom framework allows photographers to modify the FOV, the extension distortion, and the perspective of the image. They can also play with the sense of depth of the scene and the relative magnification of objects at different depths. Moreover, our framework allows compositions that would not be possible with a physical camera system: it can simulate multi-perspective cameras that can image different depth ranges in the scene with different focal lengths.

Figure 1(d), shows an animation of the type of post-capture control our method offers. Each frame in the sequence is rendered by our framework and results from the camera projection model shown on the right. Initially, the FOV (red cone) is decreased and the camera is moved back to preserve the magnification of the woman, reducing the perspective distortion. Subsequently, the appearance of the foreground subject is preserved and the background (briefly marked in green in the animation) is imaged with a shorter focal length (green cone). Finally, the very back of the scene is "pulled closer" by increasing the focal length for the part of the scene behind the first green column in the scene (shown in blue).

This paper makes several contributions to the filed of computational photography. The first is the idea of computational zoom itself, a framework that allows a photographer to change the sense of depth of a scene, the relative magnification of objects at different depths, the amount of perspective distortion, and other aspects of picture composition *in post-capture*. One key ingredient of our approach is the definition of multi-perspective cameras that can image different depths of the scene with different focal lengths. We have developed a real-time tool that shows how the composition changes as the user changes these parameters. We also propose a novel method to estimate high-quality, dense 3D estimation despite the limited parallax induced by a dolly-in motion. Finally, we adapt the unstructured lumigraph rendering approach [Buehler et al. 2001] to deal with multi-perspective cameras.

Throughout the paper we use *animated figures* to better convey our results. They are indicated with a **bold** caption, and can be viewed in a media-enabled PDF viewer, such as Adobe Reader, by clicking on them.

2 RELATED WORK

Computational zoom has three different components: dense depth reconstruction from a dolly-in sequence, the specification of a desired multi-perspective camera, and image synthesis of the final result. Here we provide an overview of related work in each area.

2.1 Dense 3D Reconstruction

We need to estimate dense depth-maps for multiple frames in order to synthesize images under multi-perspective camera projections. An excellent survey of multi-view stereo reconstruction methods is available in Seitz et al. [2006]. Many multi-view stereo (MVS) reconstruction methods [Fuhrmann et al. 2014; Furukawa and Ponce 2010] are designed to give a dense 3D model for a well sampled scene. However, for image stacks captured with a dolly-in motion, these approaches give very sparse reconstructions, leaving large holes in the estimated depth-maps and cannot be used for our image synthesis application. This problem occurs because the dolly-in motion, by construction, causes the epipole of each camera to fall inside the field of view of other cameras, making it harder to reconstruct geometry in these regions. We show the output of Furukawa and Ponce [2010] and Fuhrmann et al. [2014] on our captured image stacks in Fig. 7.

Instead, we rely on a class of 3D-reconstruction algorithms that estimate 3D plane parameters for each patch in an image. For example, PatchMatch stereo reconstruction [Bleyer et al. 2011] estimates plane parameters at each image patch using randomized PatchMatch [Barnes et al. 2009] to efficiently search the large space of plane parameters. Heise et al. [2015] extend this approach to the multi-view case with a variational framework. Galliani et al. [2015] present a fast local-matching approach that allows to extend PatchMatch stereo to multi-view on a GPU. This is one of the top performing algorithms according to standard benchmarks. However, these

[1]Extension distortion is a function of camera-to-subject distance alone, and is independent of the focal length. However, shorter focal lengths allow the photographer to move closer to the subject and still keep it entirely in frame, thus exacerbating this effect.

approaches all assume a lateral camera motion, and do not handle a dolly-in motion well, since the latter motion reduces the parallax towards the center of the images. In contrast, we propose a multi-pass reconstruction approach designed for dolly-in camera motion which gives us dense, accurate depth maps. Specifically, we leverage the plane parameters to estimate the change in scale between patches and thereby compute depths where parallax is negligible. We build upon Galliani et al.'s fast patch-matching algorithm to do this.

2.2 Multi-perspective Imaging

As their name suggests, multi-perspective images combine multiple perspectives of a scene into a single image. There has been a lot of work in multi-perspective modeling, rendering, and imaging, as detailed in the state-of-the-art survey by Yu et al. [2008].

Here we present some of the work in multi-perspective imaging that combine images captured with different motions of a pinhole camera into a single image. Sietz et al. [2003] and Zomet et al. [2003] synthesize multi-perspective images and show different visualizations of the scene by re-sampling a video cube obtained by moving cameras in a continuous manner. Multi-perspective panoramas [Agarwala et al. 2006; Kopf et al. 2010; Roman et al. 2004] combine images captured by a sideways-moving camera to generate long, image panoramas of urban façades. Yu et al. [2004] show multi-perspective rendering of objects by combining different perspectives of objects captured on a turntable. In our work, the multi-perspective images are formed by combining images captured by roughly moving the camera in a dolly-in or dolly-out motion.

Our multi-perspective camera model is closely related to the Graph Camera model [Popescu et al. 2009] that combined multiple pinhole cameras in general positions. However, the Graph Camera model is mainly focused on visualizing virtual 3D models, and shows very limited use for real-world scenes. More recently, Lieng et al. [2012] proposed an interactive system to generate multi-perspective images by combining multiple viewpoints. Their system pastes part of one image into a polygonal portal in another image, and hence has limited applicability for general composition.

There has also been some work that manipulates the perspective of objects using a single photograph. For example, Fried et al. [2016] allows changing the perspective distortion of a face in post-processing. This approach is limited as it can remove the undesirable perspective distortion only for close-up portraits. Our approach, on the other hand, is more general and allows a user to change the perspective of any object, although at a higher computational cost and with some limitations as we discuss later. Carroll et al. [2010] provide an interface to distort the perspective in (mostly) architectural photographs. The user provides constraints on vanishing points and lines, which are then used to warp the image, thereby enabling the simulation of telephoto or wide-angle projections, among others. Although our approach can achieve similar effects for certain scenes, it differs in several important ways. Specifically, we do not focus on architectural scenes and do not require the explicit modification of vanishing points and lines. Rather a user can selectively change the perspective based on depth, which requires handling occlusions, disocclusions, and fine-grained segmentation.

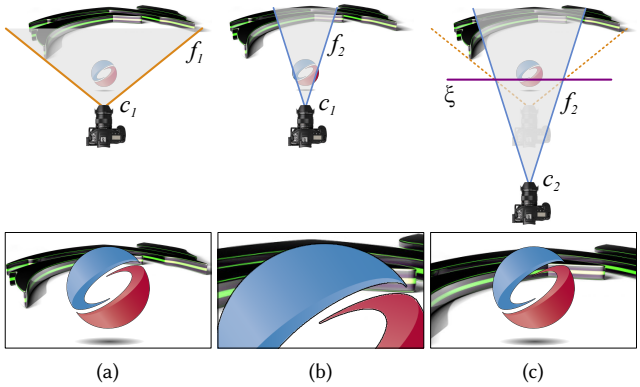


Fig. 2. Given a scene and a camera position c_1, two different focal lengths will produce the same image, only with a different magnification and cropping, (a) and (b). However, if the camera is also moved, the relative magnification of objects at different depths changes. In fact, objects that lie at the intersection of the fields of view of cameras at different locations (marked by ξ in (c)) will have the same magnification and the rest of the scene will be zoomed either in or out, see text.

2.3 Image-based Rendering

Image-based rendering (IBR) uses captured images of a 3D scene to synthesize arbitrary views [Shum et al. 2007]. Although most previous work focused on novel-view synthesis with a pin-hole projection, our goal is to synthesize new views with a multi-perspective camera projection. To do this, we borrow ideas from previous work with some modifications. In the original light-field rendering work, Levoy and Hanrahan [1996] showed that image-based rendering can be done in ray space by capturing many images of the scene without needing an intermediate geometric representation of the scene. However, others have shown that more accurate results are possible if the captured images are augmented with approximate geometry [Chen and Williams 1993; Debevec et al. 1996; Gortler et al. 1996].

Building on that, Buehler et al. [2001] take images captured from arbitrary positions and orientations along with the approximate geometry of the scene to perform view interpolation. We modify this approach to synthesize images under multi-perspective camera projection. Recent approaches by Chaurasia et al. [2013; 2011] tackle the problem of uncertain depth estimation by performing depth synthesis and using shape preserving warps to improve view interpolation. Chen et al. [2011] introduces interactive approach for 3D video editing that allows for viewpoint changes. Another set of recent approaches [Kopf et al. 2013; Sinha et al. 2012] are focused on handling scene reflections, glossy surfaces and transparent surfaces. Such approaches could be adopted in our image synthesis framework to further improve the quality of our results.

3 THE COMPUTATIONAL ZOOM FRAMEWORK

The goal of our computational zoom framework is post-capture modification of a picture's composition. We achieve this by allowing the manipulation of focal length and camera position, and by extending their interplay with the introduction of multi-perspective

cameras. We start by reviewing how these parameters affect the final image. First, recall that the field of view (FOV) of a lens is inversely proportional to its focal length, f:

$$\text{FOV} = 2 \cdot \arctan\frac{d}{2f}, \tag{1}$$

where d is the size of the sensor or film. The toy example in Fig. 2 demonstrates how this impacts the final image by showing a scene captured with different focal lengths and camera positions. Assume that the first focal length, f_1, is sufficiently short (i.e., the FOV is sufficiently wide) to capture most of the scene from position c_1, as shown in Fig. 2(a). Capturing an image from the same position, but with a longer focal length f_2, will zoom and crop this image (see Fig. 2(b)). Here, the narrow FOV maps a smaller part of the scene to the same area on the sensor or film, thus effectively magnifying it. A particularly interesting configuration is one where the camera with focal length f_2 is moved back so that the two FOV frustra intersect at the depth of the foreground object, as in Fig. 2(c): objects at the depth where the width of the two FOVs is the same will be imaged with the same magnification by the two cameras. However, the background will be zoomed in by the camera at c_2, since its FOV is narrower.

This observation allows us to formalize how a dolly-zoom video sequence is created: while the camera is moved in or out of the scene, the focal length must be changed so that the FOVs of different frames intersect at a plane, which we call the *dolly plane* ξ. This ensures that the magnification of a subject at the dolly plane stays the same throughout the sequence.

By simulating different focal lengths and camera positions, we enable the same flexibility that a photographer has in the field, but in post-capture. However, the magnification of objects at different depths in the scene still remains constrained in one of two ways. If either the focal length or the camera position are changed, the whole scene is zoomed in or out (Fig. 2(b)), though not necessarily by the same amount at all depths. If both the focal length and camera position are changed, a dolly plane is defined (Fig. 2(c)); when objects in front of the dolly plane are magnified, objects beyond it will be reduced in size, and vice versa. In the next section, we present a multi-perspective camera model that overcomes these limitations.

3.1 A Flexible Multi-perspective Camera Model

To decouple the magnification of different depth ranges, we can define different types of *multi-perspective* cameras where the focal length changes with depth. For instance, we can define a *long-short* configuration (see Fig. 3(c)), where the focal length decreases beyond a given plane, or a *short-long* configuration (see Fig. 3(d)), where the opposite is true.[2] The first configuration, for instance, would magnify objects in front of the dolly plane and zoom out those beyond it, thus capturing a larger part of the background. The second configuration would allow the opposite composition, and would increase the extension distortion of the foreground objects. To achieve such configurations, we can merge information from the two cameras c_1 and c_2. For instance, the short-long configuration in Fig. 3(d) can be synthesized by using camera c_1 for objects in front of the dolly plane, and camera c_2 for the rest of the scene.

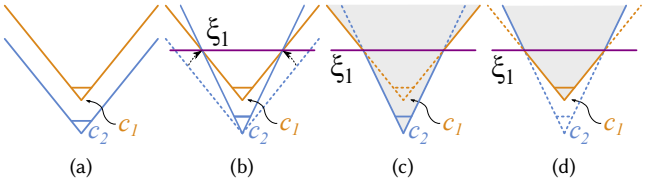


Fig. 3. Throughout the paper, we represent a camera with its center of projection and the boundaries of its field of view (FOV). (a) Given two cameras c_1 and c_2 with the same focal length but different positions, (b) we can zoom and crop the FOV of camera c_2 to match the FOV of c_1 at dolly plane ξ_1. (c) We define a *long-short* multi-perspective camera as one whose focal length decreases beyond certain depth, while (d) a *short-long* multi-perspective camera is one where focal length increases beyond certain depth.

To define multi-perspective cameras more formally, suppose we want to simulate a short-long configuration using two pin-hole cameras with centers of projection c_1 and c_2. Without loss of generality, assume both cameras have the same focal length f, Fig. 3(a), and that their projection matrices are P_1 and P_2, respectively (recall that the projection matrix subsumes the focal length in its representation).

Since we merge information from two cameras, the two FOVs must match at the dolly plane in order to avoid artifacts around the transition region. To enforce this, we modify the projection matrix P_2 to $\widehat{P}_2 = H^{\xi_1}_{2\to 1} P_2$, where $H^{\xi_1}_{2\to 1}$ is the homography induced by ξ_1 from c_2 to c_1. This homography specifies how points in camera c_2 map to camera c_1 when projected first to plane ξ_1. With few simple matrix operations, it is easy to see that:

$$\hat{x}_2 = \widehat{P}_2 X = x_1, \tag{2}$$

where X is a point at the dolly plane, x_1 is its projection onto c_1, and $\hat{x}_2$ is the projection of X in the new camera defined by $\widehat{P}_2$. Equation 2 shows that $\widehat{P}_2$ satisfies our requirement, as the image of a 3D point on the dolly plane is mapped to the same location on the sensor of the two cameras (here assumed to have the same resolution for simplicity).

Hence, images of an object that lies on the dolly plane are aligned when imaged using $\widehat{P}_2$ and P_1. An object that lies beyond the dolly plane is magnified when imaged using $\widehat{P}_2$ as compared to P_1 and an opposite effect happens for an object in front of dolly plane. We discuss this in detail and offer a derivation of Eq. 2 in the supplementary material.[3]

The image of a scene point X from of a short-long multi-perspective system can then be computed as:

$$x = \begin{cases} P_1 X & \text{for } X \in [c_1, \xi_1) \\ \widehat{P}_2 X & \text{for } X \in [\xi_1, \infty) \end{cases}, \tag{3}$$

where, with a slight abuse of notation, $[c_1, \xi_1)$ indicates the range of depths from the center of projection c_1 to plane ξ_1. The image by a long-short multi-perspective system can be similarly defined

[2] "Short" and "long" here do not refer to an absolute focal length, but rather indicate a relative change: for a short-long multi-perspective camera the focal length increases beyond the dolly plane.

[3] Supplementary material can be found here: https://doi.org/10.7919/F4VD6WCJ

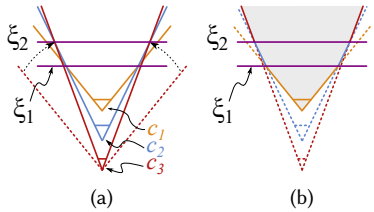


Fig. 4. (a) We can extend multi-perspective cameras to have more than one dolly plane by using multiple camera positions. (b) After appropriately warping the cameras, we can, e.g., define a multi-perspective camera whose focal length increases after each of the planes.

by swapping the cases in Eq. 3 (see supplementary material). This equation shows that we need to know per-pixel depth for each input image in order to create a multi-perspective image. We explain our depth estimation in detail in the next section.

To avoid abrupt perspective changes at the dolly plane, we can also define a *dolly transition region*. Mathematically, we represent this region as a sequence of dolly planes that approximate a smoothly-varying focal length with a piece-wise linear function. To show how this can be done, we add a third camera with projection matrix P_3 even farther from the scene, and define a second plane ξ_2. To make the FOV of this camera intersect the FOV induced by $\widehat{P}_2$ (see Fig. 4), we need to modify its projection matrix to be:

$$\widehat{P}_3 = H_{2\to1}^{\xi_1} H_{3\to2}^{\xi_2} P_3. \tag{4}$$

We can now extend this definition from three regions to N regions separated by $N-1$ dolly planes, such that, for each region K (where $1 \le K \le N$), we define:

$$x = \widehat{P}_K X \text{ if } X \in [\xi_{K-1}, \xi_K), \tag{5}$$

where

$$\widehat{P}_K = \prod_{i=1}^{K} H_{i\to i-1}^{\xi_{i-1}} P_K. \tag{6}$$

Since there is no camera at c_0, $H_{1\to0}^{\xi_0}$ is specified by the user and adjusts the projection of the first camera to incorporate desired transformations such as scaling, translation, or an arbitrary homography. It can also be simply set to identity in order to use the information from the first camera directly.

Equation 5 defines our multi-perspective camera model. Note that, although Eq. 6 assumes a specific order in which the cameras are used in each region (the camera at c_K is used in region K), it is trivial to extend it to work with a generic sequence, which may even include the same camera multiple times (we offer a generalization of Eq. 6 in the supplementary material).

The ability to define multiple dolly planes allows us to create dolly transition regions, and, more importantly, gives us flexibility over which depth range is imaged with which projection matrix. Note that any combination of long and short focal length sequences is valid: in Fig. 1, for instance, the final result is produced with a long-short-long multi-perspective camera. We show a second example of a long-short-long multi-perspective composition in Fig. 5. While any number of dolly-transition regions could be defined, empirically we found that two regions, each of which defined by two to five dolly planes, are sufficient for most scenes.

Discussion. Our framework can be seen as an application of image-based modeling and rendering (IBMR). However, unlike previous methods, we need to combine information from cameras that are potentially far from the desired view. When the depth estimation is inaccurate, the farther the cameras, the more visible the artifacts from warping. This puts more stringent constraints on the quality of the depth estimation. In Sec. 4, we describe a novel depth estimation method designed to overcome these issues. Moreover, our image synthesis strategy must account for the unavoidable depth uncertainty, and must cope with large disocclusion regions caused by the light bending as the focal length changes. We describe our multi-perspective image synthesis algorithm in Sec. 5.

4 DEPTH ESTIMATION

For each image in the input stack, we need to compute a dense, high-quality depth-map in order to apply Eq. 5. The depth estimation algorithm needs to achieve the following goals:

- **Maximize geometric consistency across the image stack.** Because we combine the content of multiple images, we must ensure that corresponding points in the different images are the projection of the same 3D point. This can be achieved by enforcing geometric consistency between the different depth maps.
- **Strike a balance between high depth certainty and pixel coverage.** We need high-quality depth to warp images correctly. A high threshold on depth certainty would produce high-quality depth-maps, but may result in a large number of pixels not having a depth estimate.
- **Maximize photometric consistency across the image stack.** When warping one image to another, a slightly incorrect depth that produces a photometrically-consistent warped source is acceptable.[4]
- **Cope with limited parallax.** The input stack is captured while walking into the scene, causing the epipoles of image pairs to project inside the images themselves.

To address these issues we propose the multi-pass, patch-based, multi-view stereo (MVS) strategy, shown in Fig. 6. The input to our system is the dolly-in stack, as well as the camera parameters estimated from it with VisualSFM, a standard structure-from-motion package [Wu 2011; Wu et al. 2011]. From these, we first estimate both depth and normal maps using each image as a reference against other images in the stack (Sec. 4.1). Initially, there is no guarantee that the estimated values are consistent across different images, because the maps are computed for each image independently. Therefore, we explicitly enforce consistency across the stack. For this task, the combination of normals and depth allows us to better deal with the lack of parallax, as we can define matching criteria other than simple disparity (Sec. 4.2). However, we still need depth and normals information for the areas where the previous step rejected inconsistent parameters. Therefore, we introduce a cross-bilateral propagation step (Sec. 4.3). Finally, we use an multi-pass approach

[4]The rationale, common to other view-interpolation methods, is that an inaccurate estimate of the depth that allows the images to warp onto each other correctly, should also allow to warp them to novel views that are close to the original camera locations.

(a) Four input images from the stack (b) Undistorted foreground (c) Set desired size for the tree (d) Magnify far background

Fig. 5. Sample multi-perspective workflow. (a) Four of the input pictures captured with a fixed focal length while walking away from the scene. We would like the man in the foreground to be the same size in the final result as in the top left image. However, in that image his legs are distorted, so we remove the distortion by zooming and cropping the bottom-left image in (a) to get the result as shown in (b). However, this changes the composition of the rest of the image. (c) A long-short composition allows us to bring the tree back into frame. (d) Finally, a long-short-long multi-perspective camera magnifies the far background, thus achieving the desired composition.

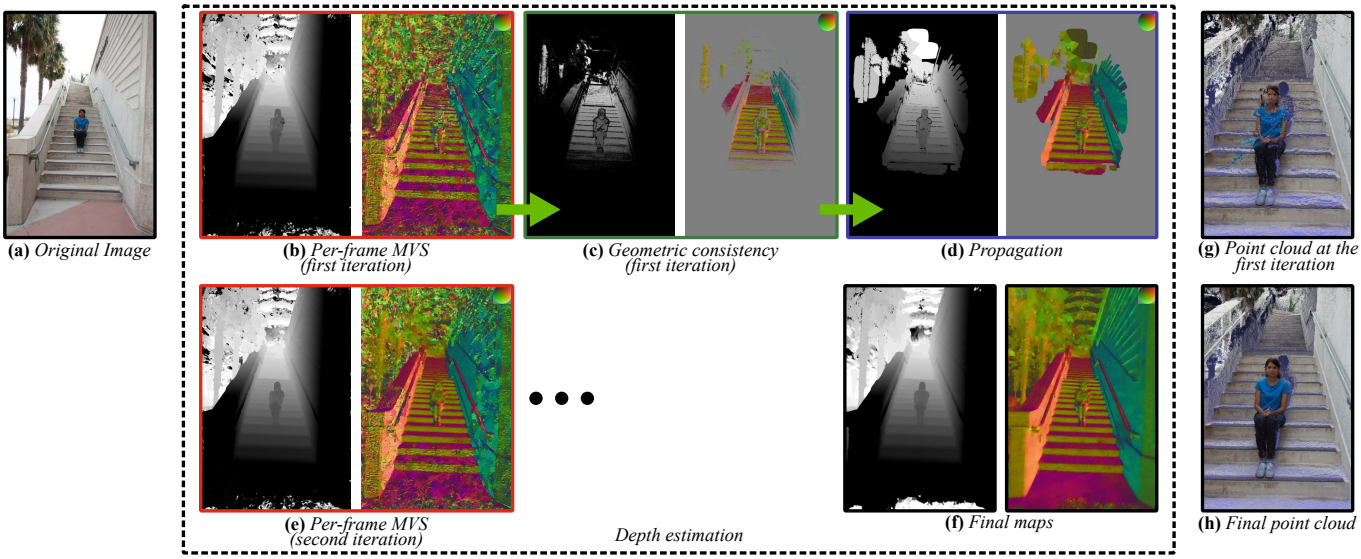


Fig. 6. Our depth reconstruction pipeline. For each image in the stack such as the one shown in (a), we compute (b) depth and normal maps using multi-view stereo (MVS) by enforcing photometric consistency (see Sec. 4.1). (c) We then enforce geometric consistency across all these depth and normal maps. This results in more accurate maps, but there may be missing values. (d) We then perform cross-bilateral propagation to fill in missing information, and (e) use both the propagated and original values in a multi-pass scheme. (f) Our approach yields dense, high-quality maps, suitable for computational zoom applications. Insets (g) and (h) show the reconstruction quality after only enforcing photometric consistency, and after the whole pipeline, respectively.

to determine high-certainty correspondences, while ensuring that we have a depth estimate for a sufficient number of pixels (Sec. 4.4).

4.1 Per-frame Multi-view Stereo

We take the relatively standard approach of enforcing photometric consistency to perform our depth reconstruction. For each pixel p of each image I_i in the stack, we estimate the parameters of the plane tangent to the corresponding 3D scene point, $\pi_i(p) = [\mathbf{n}^T d]$, where $\mathbf{n}$ is the plane normal and d is the depth, both specified in a global coordinate system. To do this, we use the PatchMatch stereo approach proposed by Bleyer et al. [2011], and extended to MVS by Galliani et al. [2015]. Inspired by the PatchMatch algorithm [Barnes et al. 2009], these methods use alternating propagation and randomization steps to generate tangent plane hypotheses, thus allowing for an efficient exploration of the space of parameters.

For this stage we follow the method by Galliani et al., which we briefly describe here for completeness. Given $\pi_i(p)$, we can compute the coordinates of q, the corresponding pixel in a second image I_j, through the plane-induced homography: $q = H_{i\to j}^{\pi_i(p)} p$. We can similarly compute the neighborhood $N_j(q)$ corresponding to the neighborhood $N_i(p)$. We can then define the photometric consistency induced by the plane $\pi_i(p)$ as:

$$E_{\text{photo}}(\pi_i(p)) = \sum_{j \in J_i(p)} \rho(N_i(p), N_j(q)), \tag{7}$$

where $\rho(\cdot)$ is a dissimilarity (distance) function that includes bilateral weights to decrease the influence of pixels in $N_i(p)$ that may differ from pixel p. We elaborate on the details of Eq. 7, including the definition of the neighborhood and the formula of the plane-induced homography, in the supplementary material. Finally, $J_i(p)$ is the set of the Q images in the stack that yield the smallest $E_{\text{photo}}(\pi_i(p))$. We discuss how we select Q in Sec. 4.4. Unlike Galliani et al., we do not constrain that images in $J_i(p)$ to be separated by a large viewing angle.

4.2 Geometric Consistency Across the Stack

At this point, for each pixel p in every input image I_i, we have an estimate of the tangent plane parameters $\pi_i(p)$. These parameters are estimated using as many images from the stack as possible, but taking one image at a time as reference I_i. This can potentially yield different estimates for different images. However, in the IBR stage we combine information from multiple images into one, thus requiring the depth estimates to be consistent across the stack.

To enforce depth consistency explicitly, we first compute a geometric error between the plane parameters estimated for two images I_i and I_j. To do this, we can use the plane parameters $\pi_i(p)$ to find the 3D point X_p associated with pixel p, and then project it onto I_j to find pixel q, which, in turn, will map to 3D point X_q. A common approach to checking for consistency is to compute the error in disparity space [Galliani et al. 2015].

However, because our images are captured with a dolly-in motion, the epipoles are within the images themselves, making disparity a poor criterion for our approach. Instead, we could use the plane parameters themselves to measure consistency. Specifically, plane parameters $\pi_i(p)$ and $\pi_j(q)$ are consistent when the induced homographies warp corresponding patches from I_i and I_j to the same 3D patch. Therefore, our consistency error could ideally be computed by comparing the product of the two homographies to the identity matrix I:

$$E = H_{j\to i}^{\pi_j(q)} \cdot H_{i\to j}^{\pi_i(p)} - I. \tag{8}$$

Note that the error in Eq. 8 is a 3×3 matrix that subsumes several measures of consistency, not all of which are associated with a physical meaning, so it is not clear how to weigh these terms to compute a scalar error. Instead, we focus on two specific criteria that Eq. 8 incorporates that have a physical impact on our results, and which can be computed from the plane parameters: reprojection consistency and scaling error. We also add a third consistency measure based on color information to improve the quality of our reconstruction. We describe these three metrics below.

Reprojection consistency. To determine whether the depth values associated with the pixels in image I_i are consistent with those of pixels in image I_j, we compute a reprojection error as

$$E_r^{\{i,j\}}(p) = |p - \hat{p}|. \tag{9}$$

To obtain $\hat{p}$, we first project p to its corresponding pixel q in I_i using the 3D point X_p. Then, we project q back to I_i using the 3D point X_q to find $\hat{p}$. This criterion has two shortcomings. First and foremost, for pixels close to the epipole, any X_p and X_q would yield a small $E_r^{\{i,j\}}$. Second, the reprojection $\hat{p}$ uses only one of the plane parameters, the depth.

Scaling error. The problem near the epipole can be addressed using scale, since the homography induced by plane $\pi_i(p)$ scales a patch from image i based on its depth and orientation, regardless of its proximity to the epipole. Specifically, we can compute the scale change induced by homograpy H as:

$$\text{Scale}(H) = \sqrt{\det\left(\begin{bmatrix} h_{11} - h_{13}\cdot h_{31} & h_{12} - h_{13}\cdot h_{32} \\ h_{21} - h_{23}\cdot h_{31} & h_{31} - h_{23}\cdot h_{32} \end{bmatrix}\right)}, \tag{10}$$

where h_{rc} are the elements of H, normalized by h_{33}. Given the scales $s_{i,j} = \text{Scale}(H_{i\to j}^{\pi_i(p)})$ and $s_{j,i} = \text{Scale}(H_{j\to i}^{\pi_j(q)})$, we define the scaling error as:

$$E_s^{\{i,j\}}(p) = |s_{i,j} \cdot s_{j,i} - 1|. \tag{11}$$

While there exist planes with different normals that could produce a low $E_s^{\{i,j\}}$, we find that this measure is sufficiently discriminative for our purposes.

Color consistency. If two pixels are imaging the same point in the scene, their color information should be consistent, under the typical assumption of a diffuse shading model. Therefore, we define an additional color consistency term:

$$E_c^{\{i,j\}}(p) = \max_{C=\{R,G,B\}} \left|C_i(p) - C_j(q)\right|, \tag{12}$$

where $C_i(\cdot)$ is one color component in image i at a particular pixel. In practice, we allow a little tolerance on the exact location of the pixels to account for the artifacts due to color interpolation in the different stages of the imaging pipeline: instead of only looking at p, we compute Eq. 12 for all of the pixels in a 3×3 neighborhood around it, and take lowest error among all of them.

Combining these three metrics, we then say that the plane parameters $\pi_i(p)$ are *Q-consistent* if there exist at least Q images I_j for which the following Boolean expression is true:

$$\left(E_r^{\{i,j\}}(p) < 1.5\right) \wedge \left(E_s^{\{i,j\}}(p) < 0.1\right) \wedge \left(E_c^{\{i,j\}}(p) < 15\right). \tag{13}$$

The thresholds used above were found empirically. We discuss the choice of Q, which is the same as in Eq. 7, in Sec. 4.4. Once we find the Q-consistent plane parameters, we fuse (i.e., average) them to further reduce their noise [Galliani et al. 2015]. We then update the plane parameters for all the consistent points with the fused estimate. To easily identify consistent points to use for initialization in our multi-pass approach (Sec. 4.4), we maintain a *consistency indicator map* for each frame $\mathbf{C}_i(p)$ that is 1 if $\pi_i(p)$ is Q-consistent, and 0 otherwise.

4.3 Cross-bilateral Propagation of Consistent Plane Parameters

At this point, for each input image I_i, we have regions for which we have Q-consistent estimates of tangent plane parameters. However, we typically lack an estimate for some of the pixels, in particular for stringent (high) values of Q. To address this problem, we make the common assumption that pixels that are close to each other in image space and have similar colors are likely to belong to the

same surface, and therefore share similar plane parameters. Hence, we use image I_i as a guide to propagate plane parameters from the consistent regions to the nearby inconsistent areas. A simple cross-bilateral filter would propagate information, but also smooth the Q-consistent estimates. To preserve the consistent information, we apply the same cross-bilateral filter to a second map initialized with 1's for pixels associated with a Q-consistent estimate, and 0's elsewhere. Normalizing the output of the original cross-bilateral filter by this second map ensures that the original values are left unchanged [Gallo et al. 2015].

Note that there is no guarantee that the propagated plane parameters are Q-consistent as well. For this reason, together with $\mathbf{C}_i(p)$, we maintain a second consistency map, $\mathbf{B}_i(p)$, that indicates the pixels whose estimate was propagated from nearby regions rather than having been estimated directly. This allows us to use them as initialization values for subsequent passes of our algorithm, as we explain in Sec. 4.4. Also note that, after this step, a portion of the pixels might still not have an estimate of their plane parameters.

4.4 A Multi-pass Approach for Setting Q

In Sec. 4.2, we introduced Q-consistency, which indicates the number of images from the stack used to compute Eq. 7, as well as the minimum number of images that should satisfy Eq. 13 in order for the plane parameters to be labeled as consistent. Choosing the right value for Q is critical. If a 3D point corresponding to pixel p is visible only in a small subset of images because of occlusions, using a high Q could penalize the correct plane $\pi_i(p)$. On the other hand, using a Q that is too small will reduce the quality of the depth estimation. Although our goal is to have the best depth estimate for each pixel in all images, for some pixels we must settle for depth estimates with higher uncertainty.

Therefore, we propose to use a multi-pass approach. We first estimate high-quality plane parameters using a high value of Q, at the cost of doing so for a small portion of the pixels. We then progressively relax our requirement on the quality of the estimation by lowering Q, which yields a depth estimate for increasingly larger portions of the pixels.

Specifically, we set $Q = \min\{N/2, 15\}$ in the first pass, where N is the total number of images. In each subsequent pass, we lower Q roughly logarithmically. We initialize each pass with the plane parameters found in previous iterations for pixels for which $\mathbf{C}_i(p) = 1$ (i.e., they passed the consistency check) or $\mathbf{B}_i(p) = 1$ (i.e., their parameters were propagated from neighboring regions), and with random values for the remaining pixels. We then enforce the Q-consistency and propagation steps described earlier, updating the plane parameters everywhere except for pixels with $\mathbf{C}_i(p) = 1$. This preserves the higher-quality parameters found in earlier passes. Even after a few passes, some pixels may still not have consistent parameters estimates. For such regions, we perform a final cross-bilateral propagation as described in Sec. 4.3, but without an additional consistency check.

4.5 Results of Our Depth Reconstruction

We show results of our depth reconstruction algorithm for different scenes in Figs. 6 and 7. Figure 7(a)-(c), in particular, provides a comparison with state-of-the-art algorithms, which struggle to produce a dense point cloud due to the limited parallax induced by the dolly-in motion. Even only after the first pass, our method produces a denser point cloud, Fig. 7(d). Note that both the result by Galliani et al. [2015], Fig. 7(c), and ours, Fig. 7(d), are obtained with the same Q; our method produces denser results thanks to the geometric consistency criteria we describe in Sec. 4.2. Figure 7(e)-(g) shows our result after the final pass for views that are significantly different than any of the original camera positions. Our point clouds are generally dense and accurate, aside from regions that are seen by few (or no) cameras, and for large texture-less regions, such as the sky or the woman's black coat.

5 IMAGE SYNTHESIS

Given the stack of input images and their estimated depth-maps from the previous section, we now want to synthesize novel multi-perspective projections of the scene. To do this, we build upon the unstructured lumigraph rendering (ULR) method of Buehler et al. [2001], which was designed to synthesize a novel view of a scene given proxy geometry and a stack of images. However, our problem differs from the original ULR approach in two ways: 1) we need to synthesize multi-perspective images, and 2) we have depth-maps instead of proxy geometry.

5.1 Multi-perspective Epipolar Consistency

One simple, yet powerful observation made by Buehler et al. is that if a ray required for the novel view passes exactly through the center of projection of one of the physical cameras, the value for that pixel in the novel view should be taken directly from that camera. They refer to this property as *epipole consistency*. We now describe how this maps to the multi-perspective case, assuming that we have perfect depth information.

Epipolar consistency with a single dolly plane. Without loss of generality, we focus on the short-long multi-perspective case shown in Fig. 8(a). Here, the black lines indicate the rays r_i that project the scene points X_i to the multi-perspective image. Note that r_1 is the same ray that would project X_1 to c_1. On the other hand, r_2 bends at the dolly plane. While no physical ray exists that matches r_2, the ray that maps X_2 to c_2 matches the segment of r_2 that lies beyond the dolly plane (dotted line from X_2 to c_2). The epipole consistency then dictates that we take the pixel for X_1 from c_1, and the pixel for X_2 from c_2.

More formally, given the two input images (I_1 and I_2) of the same size, their respective depth maps (D_1 and D_2), and the depth of the dolly plane, z, we can compute the multi-perspective image I_{MP} as:

$$I_{\text{MP}} = I_1 \odot \mathbb{1}_{\{D_1 \leq z\}} + I_2 \odot \mathbb{1}_{\{D_2 > z\}} \odot \mathbb{1}_{\{D_1 > z\}}, \qquad (14)$$

where $\odot$ is an element-wise product, and $\mathbb{1}_A$ is an indicator vector of the size of each image, which has a 1 for the elements in set A, and 0 otherwise. Note that the images and depth maps here have already been warped using the homography discussed in Sec. 3.1, and that the depth maps are represented in a common, global coordinate system. Also, this equation is applied at every pixel, but we have omitted the explicit dependence on pixel position for clarity.

Equation 14 first segments the images with respect to the dolly plane, and then combines pixels from I_1 that are in front of the

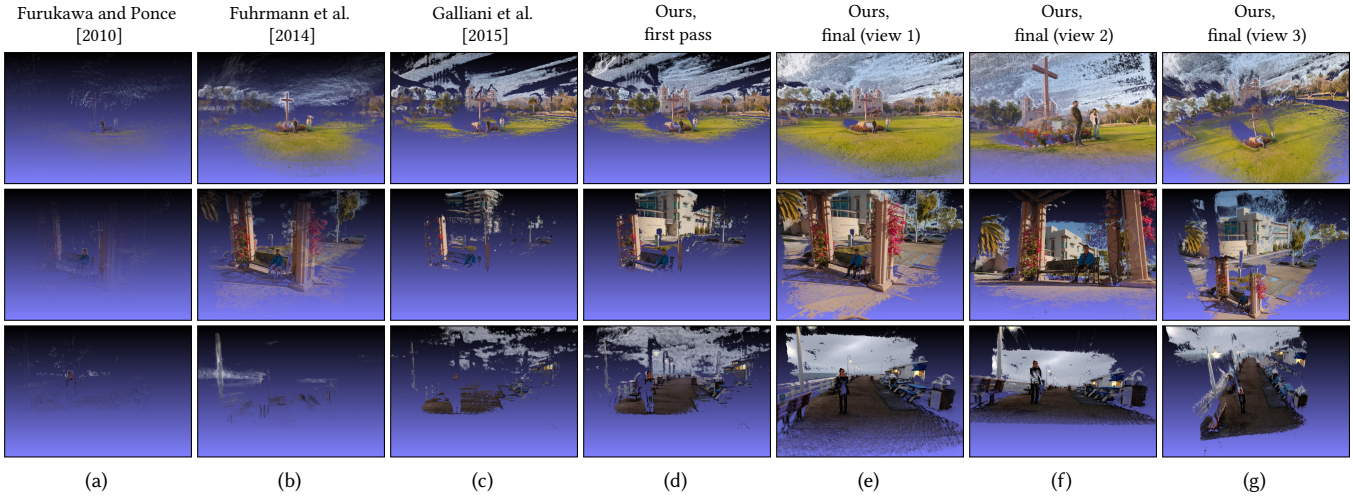


Fig. 7. Depth reconstruction comparisons. State-of-the-art methods, (a) through (c), produce sparse reconstructions because of the limited parallax induced by a dolly-in motion. After the first pass, our method produces a denser result than the method by Galliani et al. [2015] (note that Q is the same for (c) and (d)). After our multi-pass approach, our point clouds are dense and accurate, thus allowing to render views that are significantly far from any of the original views, (e) through (g). The problematic regions for our approach are those that are not seen from any of the original cameras, and some texture-less regions, such as the woman's black coat.

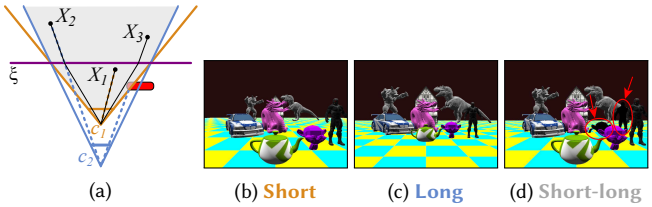


Fig. 8. (a) Short-long multi-perspective projection diagram, (b,c) input images, and (d) result of applying Eq. 14. Note the holes in (d), which are caused by points beyond the dolly plane that are occluded in (c), but visible in the multi-perspective result because the projection rays bend.

dolly plane with pixels from I_2 that are behind the dolly plane in both I_1 and I_2.[5] Although this equation guarantees that the epipolar consistency is satisfied for the single dolly plane case, it does not handle occlusions. For example, in Fig. 8(a), X_3 is occluded in c_2 because of the red object, as shown by the blue dotted line. However, it is visible in the multi-perspective image because r_3 bends around the object. This results in holes in I_{MP}, see Fig. 8(d). We observe that the hole regions H can be identified as:

$$H = \mathbb{1}_{\{D_1 > z\}} \odot \mathbb{1}_{\{D_2 < z\}}, \tag{15}$$

and we describe how we deal with them in Sec. 5.2.

Epipolar consistency with multiple dolly planes. In practice, we always want to use more than one dolly plane—even in the simple short-long or long-short cases, multiple dolly planes allow for a smoother transition between the two focal lengths.

Given $N-1$ dolly planes, our goal is to identify which of the N cameras should be used for the image of a given 3D point so that the epipolar consistency is respected. In other words, we seek to find the set of masks $\{\mathcal{M}_i\}_{i\in[1,N]}$ that indicates which regions from images $\{I_i\}_{i\in[1,N]}$ satisfy the epipolar consistency for a given multi-perspective camera. The multi-perspective image could then be synthesized as follows:

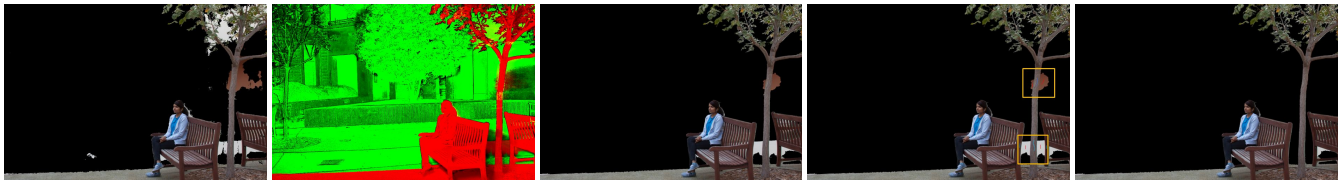
$$I_{\text{MP}} = \sum_{i=1}^{N} I_i \odot \mathcal{M}_i. \tag{16}$$

To do this, we first introduce the map $M_{i,j} = \mathbb{1}_{\{D_i \le z_j\}}$, which identifies the pixels in image i that are in front of dolly plane ξ_j at depth z_j, along with its complement $M^c_{i,j} = \mathbb{1}_{\{D_i > z_j\}}$. In the supplementary, we prove that:

$$\mathcal{M}_i = \begin{cases} M_{1,1} & \text{for } i = 1 \\ \left(\left[\!\!\left[\bigodot_{k=1}^{i-1} M^c_{k,k}\right]\!\!\right]\right) \odot M^c_{i,i-1} \odot M_{i,i} & \text{for } i \in [2, N-1], \\ \left(\left[\!\!\left[\bigodot_{k=1}^{i-1} M^c_{k,k}\right]\!\!\right]\right) \odot M^c_{i,i-1} & \text{for } i = N \end{cases} \tag{17}$$

where we use the symbol $\left[\!\!\left[\odot\right]\!\!\right]$ to denote the concatenation of element-wise products. This equation offers a simple rule to generate masks that guarantee epipolar consistency.

[5] To simplify our analysis, we assume that the dolly planes are orthogonal to the z-axis and hence segmentation using the dolly plane simply involves thresholding the depth maps D_i using depth z. Extending the analysis to slanted dolly planes is straightforward.



(a) Depth thresholding (b) Confidence map (c) Refined segmentation (d) User scribbles (e) Final result

Fig. 9. Robust depth-based image segmentation. Accurate foreground/background segmentation with respect to a dolly plane is needed for synthesizing multi-perspective images. (a) A naïve segmentation by simply thresholding the depth-maps is brittle and results in artifacts. (b) A confidence map is obtained by warping the naïve segmentations from the other cameras and filtering them based on color similarity. Here, the intensity of red indicates the number of foreground votes, and green denotes background votes. A simple majority voting scheme gives us good segmentations, which is further refined with image matting, shown in (c). Nevertheless, if the depth for some regions is consistently wrong in multiple images, we will get wrong segmentation results. When this happens, we allow user to perform interactive segmentation by detecting the problematic regions as shown by yellow boxes in (d) and then providing foreground/background scribbles to get the final result shown in (e).

5.2 ULR for Multi-perspective Camera Projections

The multi-perspective image generated with Eq. 16 may contain holes due to disocclusions. We fill these missing regions using information from other images with the ULR approach [Buehler et al. 2001], but modified in two ways. First, we define the weights for the fragments as:

$$w = \alpha\, w_{\text{ang}} + \beta\, w_{\text{dist}} + \gamma\, w_{\text{stretch}}, \tag{18}$$

where we fix $\alpha = 0.7$, $\beta = 0.15$, and $\gamma = 0.15$. For the angular weight, w_{ang}, which defines the angle between the desired view and the input view, we use the definition by Buehler et al. directly. However, because the camera motion is a dolly-in, the points around the epipole have low angular weight regardless of which camera they come from. The distance weight, w_{dist}, solves this problem by penalizing cameras c_k that are far from the desired camera c_i: $w_{\text{dist}} = exp(\|c_k - c_i\|/\sigma^2)$, where σ is set to 20% of the distance between the closest and farthest camera. Finally the texture stretch weight, w_{stretch}, penalizes fragments that are stretched anisotropically, as defined by Kopf et al. [2014].

Second, unlike the standard ULR approach, we do not have a proxy geometry but rather dense depth maps. Because of occlusions and different FOVs, some cameras c_k may see parts of the scene that are occluded in camera c_i. To prevent this from causing artifacts, we cluster the depth of the different fragments after warping them to camera c_i, and only average the ones from the closest cluster similar to the approach of Kopf et al. [2014].

5.3 Robust Depth-based Segmentation

Equations 14 and 16 require an accurate prediction of whether a pixel is in front or beyond a dolly plane. In practice, however, when the depth is even slightly inaccurate, a prediction based on hard-thresholding is noisy, as shown in Fig. 9(a). To address this issue, we propose a strategy to robustly segment the depth around the dolly plane.

Specifically, we need to decide whether point X, which projects to pixel p in image I, lies in front or beyond a certain plane. Barring occlusions, a stack of N images will yield $N-1$ additional observations of X. Therefore, we first warp the additional observations to image I. Each observation consists of a color and a binary decision on whether the point is in front or beyond the plane. We then remove the warped observations whose color does not match that of pixel p, and make a decision based on a majority vote of the remaining observations. Note that this is analogous to the work of Klose et al. [2015], where warping to image I corresponds to their gather step, and our majority vote to their filtering step.

(a) Two input images (b) **Animated multi-perspective result**

Fig. 10. This example shows how our method is able to deal with occlusions due to fine structures. Occasional errors can be handled with minor manual intervention as shown in Fig. 9.

5.4 Implementation Details

To produce the final results shown in the paper, we added a few small steps to the algorithm described so far. For example, at the dolly planes, the masks $\mathcal{M}_i$ change abruptly. To further reduce potential artifacts at those depths, we refine the segmentation by performing matting on the corresponding images using the approach by He et al. [2011]. Figure 9(c) shows an example of our final segmentation result.

Also, despite the generally high quality of our depth-based segmentation, we sometimes still observe artifacts, particularly at the boundary between textured and texture-less regions (see Fig. 9(c)). For such cases, we allow the user to interactively identify the problematic region, and mark the foreground/background regions with a few strokes, see Fig. 9(d). With this information we refine the segmentation using the standard graphcut algorithm by Boykov

Fig. 11. A typical workflow to achieve the desired composition in the viewer. The point cloud is color coded to match the color of the FOV they are imaged with. **Animated sequence.**

and Kolmogorov [2004], see Fig. 9(e). Empirically, we found that we need to perform this step in only few datasets, and usually the interaction is limited to one, sometimes two regions. Out of all the results shown in this paper, only the datasets of Figs. 10 and 17 required a minimal user interaction, shown in Fig. 9(d) for Fig. 10.

Lastly, in our pipeline, we use images of 15MP or more. For efficiency reasons, rather than computing depth on the full-resolution images, we use 2MP images to estimate the depth and perform joint bilateral upsampling [Kopf et al. 2007] to obtain the full-resolution depth-maps.

6 THE COMPUTATIONAL ZOOM WORKFLOW

The traditional paradigm for capturing a picture requires a photographer to visualize the desired composition, adjust the capture parameters to produce it, and finally trigger the shutter. Computational zoom offers the ability to manipulate the composition of the picture in post-processing. The first, more obvious, advantage of this approach is that the requirement to commit to a particular composition at capture time is lifted. Furthermore, our framework allows one to explore compositions that may not have been apparent when the picture was taken. Finally, it allows one to experiment with focal lengths that were not available at capture time, or that are not possible with a physical lens, as with multi-perspective imaging.

To capitalize on these advantages, we need a viewer that allows the user to control composition at interactive rates. However, the depth segmentation described by Eq. 16, and the image-based rendering described in Sec. 5 in general, are too slow for interactive viewing. Since we do not need high-quality images to experiment with composition, we obviate this problem by showing a preview using only the 3D point cloud, which we manipulate with OpenGL. Then the user can specify horizontal and vertical translations in image space (within some constraints), one or more dolly planes, and the corresponding focal lengths.

Figure 11 shows an example of a typical workflow in our viewer. First, we decide on the desired amount of perspective distortion of the foreground. This can be done by selecting a dolly plane at the depth of the person in the foreground, and by adapting the dolly zoom as desired. This composition, however, crops out the tree. Therefore, a second dolly plane can be added just beyond the foreground subject, and the tree can be brought back in the picture by decreasing the focal length beyond the second dolly plane (points in green in the animation). Finally, we wish to magnify the person in the background leaving the rest of the composition untouched. We achieve this by adding a third dolly plane just behind the tree, and increasing the focal length behind it.

Note that the prototype viewer we describe here (also shown more extensively in the supplementary material) is designed to expose the fine level of control that our framework offers, and to show that such manipulation can be performed interactively. An exploration of better user-interaction strategies for computational zoom, such as tap-to-select, or pinch-to-zoom, would be the subject of interesting future work.

7 RESULTS

In this section, we describe several examples of how our computational zoom framework can be used to achieve the desired composition for different types of scenarios. All the scenes were captured with a DSLR camera and, depending on the complexity of the scene, we use between 15 and 40 input images for the depth reconstruction and synthesis. The images were taken mostly along a straight line, but in some cases we move sideways as well to capture information to fill disocclusions. Note that using frames from a video as input is also possible, so long as they are not affected by rolling shutter artifacts.

We begin with examples of extension distortion, which is an important element of composition that may be desirable depending on the story the photographer wants to tell. For instance, in Fig. 12 the photographer wants to use the subject and pillars in the foreground to frame the building in the background, as shown in Fig. 12(a). However, the perspective distortion of the bench is apparent in this image. Capturing the image from farther away and with a longer focal length removes this distortion, but crops the background (see Fig. 12(b)).

With our method, we can first apply a dolly zoom to get the desired foreground composition, and then "push the background away" with a long-short configuration, as shown in the animation in Fig. 12(c). Figure 12(d) shows the foreground, background, and transition regions, $\{\mathcal{M}_i\}_{i=1:3}$ from Eq. 17. Each color corresponds to a particular $\{\mathcal{M}_i\}$, and black regions are disocclusions for which we need to use the method described in Sec. 5.2. A similar composition is shown in Fig. 16.

The opposite effect is achieved in Fig. 13, where a short-long configuration leverages extension distortion to magnify the foreground person more than the cross (short focal length), while also bringing importance to the church in the back (long focal length).

Figures 1 and 5 show more elaborate uses of our computational zoom framework. The final multi-perspective result in Fig. 1 allows us to move away the green roller-coaster track to provide a natural image frame, while bringing the bridge in the back closer. Similarly, Fig. 5 shows how a long-short-long multi-perspective camera allows to remove the foreground distortion, and resizes the tree to frame the picture nicely, all while magnifying the subject in the back.

The animation in Fig. 10(b) shows that, thanks to the depth segmentation described in Sec. 5.3, our approach can deal with occlusions due to fine structures. However, occasional errors in depth estimation still happen. This example was generated with the manual intervention described in Sec. 5.4, and shown in Fig. 9(e).

(a) Desired background (b) Desired foreground (c) **Multi-perspective result** (d) Label map

Fig. 12. The desired foreground in (a) and the desired background composition in (b) can be combined together in the long-short, multi-perspective composition in (c). **The multi-perspective result is animated**. (d) A label map for one of the frames in the sequence, see text.

(a) Two input images (b) Multi-perspective result

Fig. 13. A short-long example, which magnifies the man in the foreground more than the cross (short focal length), while also appearing to bring the background closer (long focal length).

(a) Two input images (b) **Animated multi-perspective result**

Fig. 14. The bottom input image shows the desired composition; however, the left bench is not facing the camera, whereas it is in the top input image. Using a short-long multi-perspective zoom, we can have a combination of both.

Computational zoom can be used for more advanced manipulations of the composition. The animation in Fig. 14 shows how, by placing a dolly plane roughly in the middle of the circle of benches, a user can expand the circle by using a dolly zoom. Since the background is zoomed out by this operation, a short-long multi-perspective camera configuration can help magnify it back to its original size.

Finally, we can also combine multiple videos to obtain interesting effects. For instance, we can take two videos, one with the camera static and with a moving subject (shown in Fig. 15(a)), and the second captured like a regular dolly-in stack (Fig. 15(b)). Using user-supplied segmentation masks, we can then combine the videos into one where the person appears to be "pushing away" the background, as shown in the animation in Fig. 15(d). Full-resolution versions of these and other results can be found in the supplementary material.

8 DISCUSSION AND EVALUATION

Our framework enables the manipulation of important elements of composition, such as the perspective distortion of foreground objects, by allowing the FOV to change with depth. However, this operation effectively bends the light rays coming from the scene, which can produce visible distortions. In this sense, we are offering a tradeoff between different types of distortions. This phenomenon is particularly visible in the animation in Fig. 16(c) because of the tiles on the floor exacerbate the effect from curved rays. The choice of which distortion is preferable is up to the photographer.

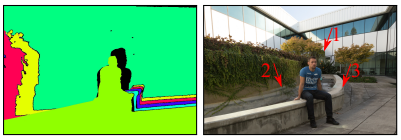
As with other stack-based computational photography algorithms, our framework may fail in the presence of dynamic scenes. However, all the examples in our paper were captured in uncontrolled environments. In fact, many stacks were captured in the presence of foliage, people walking (Fig. 1), and even cars driving in the background (Figs. 13 and 14). In general, we observe that isolated moving objects do not create objectionable artifacts. In fact, they are deemed geometrically inconsistent by the depth reconstruction and our rendering algorithm treats them as a dis-occlusion, thus filling them with other images not affected by motion in the area. In this sense, our geometric consistency criteria act as a deghosting algorithm.

For some of the scenes in the paper, the depth layers may be relatively simple to segment, as in Fig. 17. An experienced user could try to stitch together the images in the stack directly in an image editor to create the desired composition. While this is possible to some extent, our framework allows the user to experiment with different compositions interactively, some of which may not be immediately apparent. This exploration stage would not be possible in standard photo editing software. Moreover, as shown in Fig. 17(c), some disocclusion regions (marked in black) that may be too large for standard hole-filling tools; in contrast, we use the whole stack of images to generate their content. Finally, manually stitching together such an image stack without introducing artifacts is difficult when areas spanning a large depth range, such as the ground, are visible throughout the stack. We asked an advanced Photoshop

(a) Two input images and masks for FG (b) Two input images (c) First frame (d) Last frame (**Animated**)

Fig. 15. Multi-perspective, dolly-zoom video effect. An input video of the person doing a push motion is combined with a dolly-in video (without the person) as well as a foreground(FG) mask to create the illusion of the person "pushing away" the scene.

(a) Desired FG size (b) Desired FG composition (c) **Multi-perspective result**



(d) Label map (e) Photoshop layering

Fig. 16. Example of foreground(FG) distortion removal with a long-short camera. Our framework introduces a tradeoff between changing geometric distortion and bending light rays, as the lines on the ground show. **The multi-perspective result is animated**.

(a) Long (original) (b) Long-short (c) Label map

(d) Five input images from the stack

Fig. 17. While it is theoretically possible to combine an input stack (d) in photo editing software to create an output like (b), it would be quite difficult in practice due to the detailed segmentation required as well as large disocclusions, shown in black in (c).

(a) Two input images (b) **Animated multi-perspective result**

Fig. 18. Mismatches in our patch-based depth estimation do not usually lead to artifacts, as there are few of them and they mainly occur in plain areas. However, they can occur, such as in this example where artifacts appear in the sky above the lifeguard tower.

user to attempt reproducing some of the single dolly-plane results. Figure 16(e) shows one such result, which was produced in roughly 10 minutes. Note the artifacts due to disocclusions (1), and to the sharp discontinuity between foreground and background, (2) and (3). On the other hand, our framework automatically fills in the disocclusions with plausible content and computes a smooth and natural-looking transition between the different images.

Like any other patch-based method, our depth-reconstruction algorithm struggles in large, texture-less regions. The cross-bilateral propagation step described in Sec. 4.3 helps but, for very large regions such as the sky, it is not sufficient. Luckily, in these areas, the artifacts due to warping images with the wrong depth are not usually apparent, exactly because of the lack of texture. Figure 18 is a counter-example in which artifacts are visible above the lifeguard tower, and thus shows a limitation of our approach.

In terms of timing, the depth estimation is the most time consuming stage of our algorithm, which takes less than 2 hours for a stack of roughly 30 images. After defining a composition in real-time as described in Sec. 6, the production of the high-quality version of the image takes a couple of minutes for a 1 mega-pixel image.

Finally, we comment about the resolution of the captured images. For the results in this paper we used high-resolution images as an input, which certainly helps to maintain detail when the images are magnified by the homographies during composition. However, the quality of the depth and normals we recover would allow us to perform high-quality patch-based super-resolution, should high-resolution images not be available. This is an interesting area for future exploration. Moreover, certain cameras such as the Light camera, the iPhone 7, and other hand-held devices are equipped

with multiple cameras with different focal lengths that could also allow us to capture the required data.

9 CONCLUSION

In this paper, we have presented computational zoom, a framework that enables photographers to adjust the composition of an image post capture. In particular, it allows photographers to change the camera distance as well as the focal length using a stack of images that were acquired with a dolly-in motion. This already gives photographers a flexibility that was difficult to achieve previously.

Furthermore, we introduced a multi-perspective camera model that can generate compositions that are not attainable with physical lenses. This gives the photographer full control over the relative composition at different depths—allowing one to "push" certain depth ranges further away, while bringing others in. The photographer is presented with an interactive viewer to try different compositions before the framework generates a high-quality result. To demonstrate this flexibility, we showed various results that are not possible to capture with real imaging systems.

ACKNOWLEDGMENTS

We thank Arjun Muralidharan, Chitra Karanam, Desire Luong, Ekta Prashnani, Nihar Talele, Qiaodong Cui, Robert Maier, and Saandeep Depatla for helping with the datasets.

REFERENCES

Aseem Agarwala, Maneesh Agrawala, Michael Cohen, David Salesin, and Richard Szeliski. 2006. Photographing Long Scenes with Multi-viewpoint Panoramas. *ACM Trans. Graph.* 25, 3 (2006), 853–861.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Trans. Graph.* 28, 3, Article 24 (2009), 24:1–24:11 pages.

Michael Bleyer, Christoph Rhemann, and Carsten Rother. 2011. PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In *BMVC*. 14.1–14.11.

Yuri Boykov and Vladimir Kolmogorov. 2004. An Experimental Comparison of Min-cut/Max-flow Algorithms for Energy Minimization in Vision. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 9 (2004), 1124–1137.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured Lumigraph Rendering. In *Proceedings SIGGRAPH*. 425–432.

Robert Carroll, Aseem Agarwala, and Maneesh Agrawala. 2010. Image Warps for Artistic Perspective Manipulation. *ACM Trans. Graph.* 29, 4, Article 127 (2010), 9 pages.

Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Trans. Graph.* 32, 3, Article 30 (2013), 12 pages.

Gaurav Chaurasia, Olga Sorkine, and George Drettakis. 2011. Silhouette-aware Warping for Image-based Rendering. In *EGSR*. 1223–1232.

Jiawen Chen, S. Paris, J. Wang, W. Matusik, M. Cohen, and F. Durand. 2011. The Video Mesh: A Data Structure for Image-based Three-dimensional Video Editing. In *IEEE ICCP*. 1–8.

Shenchang Eric Chen and Lance Williams. 1993. View Interpolation for Image Synthesis. In *Proceedings SIGGRAPH*. 279–288.

Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. 1996. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach. In *Proceedings SIGGRAPH*. 11–20.

Ohad Fried, Eli Shechtman, Dan B. Goldman, and Adam Finkelstein. 2016. Perspective-aware Manipulation of Portrait Photos. *ACM Trans. Graph.* 35, 4, Article 128 (2016), 10 pages.

Simon Fuhrmann, Fabian Langguth, and Michael Goesele. 2014. MVE: A Multi-view Reconstruction Environment. In *Eurographics Workshop on Graphics and Cultural Heritage*. 11–18.

Yasutaka Furukawa and Jean Ponce. 2010. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.* 32, 8 (2010), 1362–1376.

Silvano Galliani, Katrin Lasinger, and Konrad Schindler. 2015. Massively Parallel Multiview Stereopsis by Surface Normal Diffusion. In *IEEE CVPR*. 873–881.

Orazio Gallo, Alejandro Troccoli, Jun Hu, Kari Pulli, and Jan Kautz. 2015. Locally Non-rigid Registration for Mobile HDR Photography. In *IEEE CVPR Workshops*. 49–56.

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The Lumigraph. In *Proceedings SIGGRAPH*. 43–54.

K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun. 2011. A Global Sampling Method for Alpha Matting. In *IEEE CVPR*. 2049–2056.

Philipp Heise, Brian Jensen, Sebastian Klose, and Alois Knoll. 2015. Variational PatchMatch MultiView Reconstruction and Refinement. In *IEEE CVPR*. 882–890.

Felix Klose, Oliver Wang, Jean-Charles Bazin, Marcus Magnor, and Alexander Sorkine-Hornung. 2015. Sampling Based Scene-space Video Processing. *ACM Trans. Graph.* 34, 4, Article 67 (2015), 11 pages.

Johannes Kopf, Billy Chen, Richard Szeliski, and Michael Cohen. 2010. Street Slide: Browsing Street Level Imagery. *ACM Trans. Graph.* 29, 4, Article 96 (2010), 8 pages.

Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. 2007. Joint Bilateral Upsampling. *ACM Trans. Graph.* 26, 3, Article 96 (2007).

Johannes Kopf, Michael F. Cohen, and Richard Szeliski. 2014. First-person Hyper-lapse Videos. *ACM Trans. Graph.* 33, 4, Article 78 (2014), 10 pages.

Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. 2013. Image-based Rendering in the Gradient Domain. *ACM Trans. Graph.* 32, 6, Article 199 (2013), 9 pages.

Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. In *Proceedings SIGGRAPH*. 31–42.

Henrik Lieng, James Tompkin, and Jan Kautz. 2012. Interactive Multi-perspective Imagery from Photos and Videos. *Comput. Graph. Forum* 31, 2pt1 (2012), 285–293.

Voicu Popescu, Paul Rosen, and Nicoletta Adamo-Villani. 2009. The Graph Camera. *ACM Trans. Graph.* 28, 5, Article 158 (2009), 8 pages.

Augusto Roman, Gaurav Garg, and Marc Levoy. 2004. Interactive Design of Multi-Perspective Images for Visualizing Urban Landscapes. In *Proc. Conference on Visualization*. 537–544.

Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. 2006. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *IEEE CVPR*. 519–528.

Steven M. Seitz and Jiwon Kim. 2003. Multiperspective Imaging. *IEEE Comput. Graph. Appl.* 23, 6 (2003), 16–19.

Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. 2007. *Image-Based Rendering*. Springer.

Sudipta N. Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-based Rendering for Scenes with Reflections. *ACM Trans. Graph.* 31, 4, Article 100 (2012), 10 pages.

Changchang Wu. 2011. VisualSFM : A Visual Structure from Motion System. http://ccwu.me/vsfm/

C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. 2011. Multicore Bundle Adjustment. In *IEEE CVPR*. 3057–3064.

J. Yu and L. McMillan. 2004. A Framework for Multiperspective Rendering. In *EGSR*. 61–68.

Jingyi Yu, Leonard McMillan, and Peter Sturm. 2008. Multiperspective Modeling, Rendering, and Imaging. In *ACM SIGGRAPH ASIA 2008 Courses*. Article 14, 36 pages.

A. Zomet, D. Feldman, S. Peleg, and D. Weinshall. 2003. Mosaicing New Views: The Crossed-Slits Projection. *IEEE Trans. Pattern Anal. Mach. Intell.* 25, 6 (June 2003), 741–754.