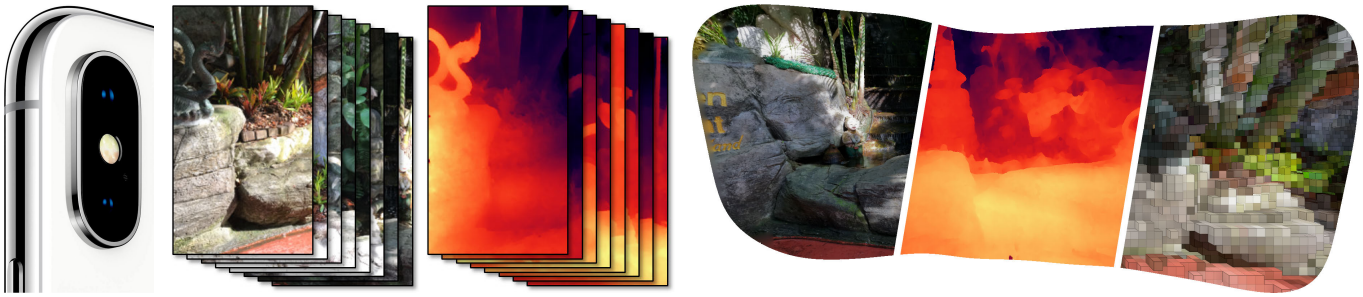
Instant 3D Photography

PETER HEDMAN, University College London*
JOHANNES KOPF, Facebook



Dual camera phone

Input burst of 34 color-and-depth photos, captured in 34.0 seconds

Our 3D panorama (showing color, depth, and a 3D effect), generated in 34.7 seconds.

Fig. 1. Our work enables practical and casual 3D capture with regular dual camera cell phones. *Left:* A burst of input color-and-depth image pairs that we captured with a dual camera cell phone at a rate of one image per second. *Right:* 3D panorama generated with our algorithm in about the same time it took to capture. The geometry is highly detailed and enables viewing with binocular and motion parallax in VR, as well as applying 3D effects that interact with the scene, e.g., through occlusions (right).

We present an algorithm for constructing 3D panoramas from a sequence of aligned color-and-depth image pairs. Such sequences can be conveniently captured using dual lens cell phone cameras that reconstruct depth maps from synchronized stereo image capture. Due to the small baseline and resulting triangulation error the depth maps are considerably degraded and contain low-frequency error, which prevents alignment using simple global transformations. We propose a novel optimization that jointly estimates the camera poses as well as spatially-varying adjustment maps that are applied to deform the depth maps and bring them into good alignment. When fusing the aligned images into a seamless mosaic we utilize a carefully designed data term and the high quality of our depth alignment to achieve two orders of magnitude speedup w.r.t. previous solutions that rely on discrete optimization by removing the need for label smoothness optimization. Our algorithm processes about one input image per second, resulting in an end-to-end runtime of about one minute for mid-sized panoramas. The final 3D panoramas are highly detailed and can be viewed with binocular and head motion parallax in VR.

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

Additional Key Words and Phrases: 3D Reconstruction, Image-stitching, Image-based Rendering, Virtual Reality

* This work was done while Peter was working as a contractor for Facebook.

Authors' addresses: Peter Hedman, University College London*, p.hedman@cs.ucl.ac.uk; Johannes Kopf, Facebook, jkopf@fb.com.

0730-0301/2018/8-ART101 $15.00
https://doi.org/10.1145/3197517.3201384

1 INTRODUCTION

Virtual reality (VR) is a fascinating emerging technology that creates lifelike experiences in immersive virtual environments, with high-end headsets now widely available. Most content that is consumed in VR today is synthetic and needs to be created by professional artists. There is no practical way for consumers to capture and share their own real-life environments in a form that makes full use of the VR technology.

That is perhaps not surprising when considering the formidability of this problem: we are looking for a method that does not require expensive hardware and is easy to use even for novice users. Yet, it should create high-quality and *truly immersive* content, i.e., a 3D representation that supports binocular vision and head-motion parallax. Finally, consumers demand very fast processing times, on the order of seconds at most.

Panoramic images and video can be easily captured now with consumer 360° cameras. While the surrounding imagery provides some immersion, the realism is limited due to lack of depth and parallax. Stereo panoramas [Peleg and Ben-Ezra 1999] provide binocular depth cues by delivering different images to the left and right eye, but these images are static and do not provide motion parallax when the user turns or moves their head.

Full immersion can only be achieved using a 3D representation, such as the 3D models generated by multi-view stereo methods. However, when applied to typical panorama datasets, captured "inside-out" from a single vantage point, these methods have to deal with a very small baseline, which causes noisy results and holes in the reconstruction. The methods are also very sensitive to even slight scene motion, such as wind-induced motion in trees. The *Casual 3D Photography* system [Hedman et al. 2017] achieves some improvement in reconstruction quality, but it is slow with a runtime of several hours per scene.

In this paper, we present a new algorithm that constructs 3D panoramas from sequences of color-and-depth photos produced from small-baseline stereo dual camera cell phones, such as recent iPhones. We take these sequences with a custom burst capture app while casually moving the phone around at a half-arm's distance. The depth reconstruction is essentially free since it is integrated into native phone OS APIs and highly optimized. Using depth maps from dual cameras makes our algorithm somewhat robust to scene motion, because the synchronous stereo capture enables reconstructing depth even for dynamic objects, though stitching them might still result in visible seams.

Our method is fast and processes approximately one input image per second, about the same time it takes to capture. We stress the importance of this point, since we found that as our system became faster, it made our own behavior with regards to capture more opportunistic: we were suddenly able to capture spontaneously on the go and even "iterate" on scenes to try different viewing angles.

The output of our algorithm is a detailed 3D panorama, i.e., a textured, multi-layered 3D mesh that can be rendered with standard graphics engines. Our 3D panoramas can be viewed with binocular and head-motion parallax in VR, or using a parallax viewer on normal mobile and web displays (see the accompanying video for examples). We can also generate interesting geometric effects using the 3D representation (Figure 1, right).

We faced two challenges when developing this algorithm, which lead to our two main technical contributions:

(1) Due to the very small baseline of dual phone cameras depth estimation is highly uncertain, and it requires strong edge-aware filtering to smooth the resulting noise. However, this leads to low frequency errors in the depth maps that prevent simple alignment using global transformations. We present a novel optimization method that jointly aligns the depth maps by recovering their camera poses as well as solving for a spatially-varying adjustment field for the depth values. This method is able to bring even severely degraded input depth maps into very good alignment.
(2) Existing image fusion methods using discrete optimization are slow. We utilize a carefully designed data term and the high quality of our depth alignment to remove the need for label smoothness optimization, and replace it with independently optimizing every pixel label after filtering the data term in a depth-guided edge-aware manner. This achieves a speedup of more than two orders of magnitude.

After stitching, we convert the 3D panorama into a multi-layered representation by converting it to a mesh, tearing it at strong depth edges, and extending the back-layer into occluded regions while hallucinating new colors and depths. When viewing the panorama away from the default viewpoint this new content is revealed in disocclusions.

We demonstrate our algorithm on a wide variety of captured scenes, including indoor, outdoor, urban, and natural environments at day and night time. We also applied our algorithm to several datasets where the depth maps were estimated from single images using CNNs. These depth maps are strongly deformed from their ground truth and lack image-to-image coherence, but nevertheless our algorithm is able to produce surprisingly well-aligned and consistent stitched panoramas. A large number of these results is provided in the supplementary material and accompanying video.

2 PREVIOUS WORK

360° Photo and Video: Using dedicated consumer hardware, such as the Ricoh Theta, it is now easy to capture full 360° × 180° panoramas. Although this is often marketed as capture for VR, it does not make use of the most interesting capabilities of that technology, and the lack of binocular and motion parallax limits the realism.

Stereo Panoramas: Binocular depth perception can be enabled by stitching appropriate pairs of left-eye and right-eye panoramic images. This representation is often called *omnidirectional stereo* [Anderson et al. 2016; Ishiguro et al. 1990; Peleg et al. 2001; Richardt et al. 2013]. Recent systems enable the capture of stereo panoramic videos using multiple cameras arranged in a ring [Anderson et al. 2016; Facebook 2016], or using two spinning wide-angle cameras [Konrad et al. 2017].

Omnidirectional stereo has a number of drawbacks. In particular, the rendered views are not in a linear perspective projection and exhibit distortions such as curved straight lines and incorrect stereo parallax away from the equator [Hedman et al. 2017]. Even more importantly, the representation does not support motion parallax, i.e., the rendered scene does not change as the user moves their head, which considerably limits depth perception, and, hence, immersion.

Parallax-aware Panorama Stitching: Some panorama stitching methods compute warp-deformations to compensate for parallax in the input images. While this reduces artifacts, it does not address the fundamental limitation that this representation does not support viewpoint changes at runtime.

Zhang and Liu [2014] stitch image pairs with large parallax by finding a locally consistent alignment sufficient for finding a good seam. Perazzi et al. [2015] extend this work to the multi-image case and compute optimal deformations in overlapping regions to compensate parallax and extrapolate the deformation field smoothly in the remaining regions. Lin et al. [2016] handle two independently moving cameras whose relative poses change over time.

Recent work [Zhang and Liu 2015] demonstrates that these approaches also extend to omni-directional stereo. However, this line of work has not yet produced explicit 3D geometry, making them unable to produce head-motion parallax in VR.

Panoramas with Depth: An alternative to generating a left-right pair of panoramic images is to augment a traditional stitched panoramic image with depth information. Im et al. [2016] construct a panorama-with-depth from small baseline 360° video. However, the fidelity of the depth reconstruction does not seem sufficient for viewpoint changes (and this has not been demonstrated.) Lee et al. [2016] use depth information to compute a spatially varying 3D projection surface to compensate for parallax when stitching images captured with a 360° rig. However, similar to before mentioned work the surface is a low-resolution grid mesh. Zheng et al. [2007] create a layered depth panorama using a cylinder-sweep multi-view stereo algorithm. However, their algorithm creates discrete layers at fixed depths and cannot reconstruct sloped surfaces.

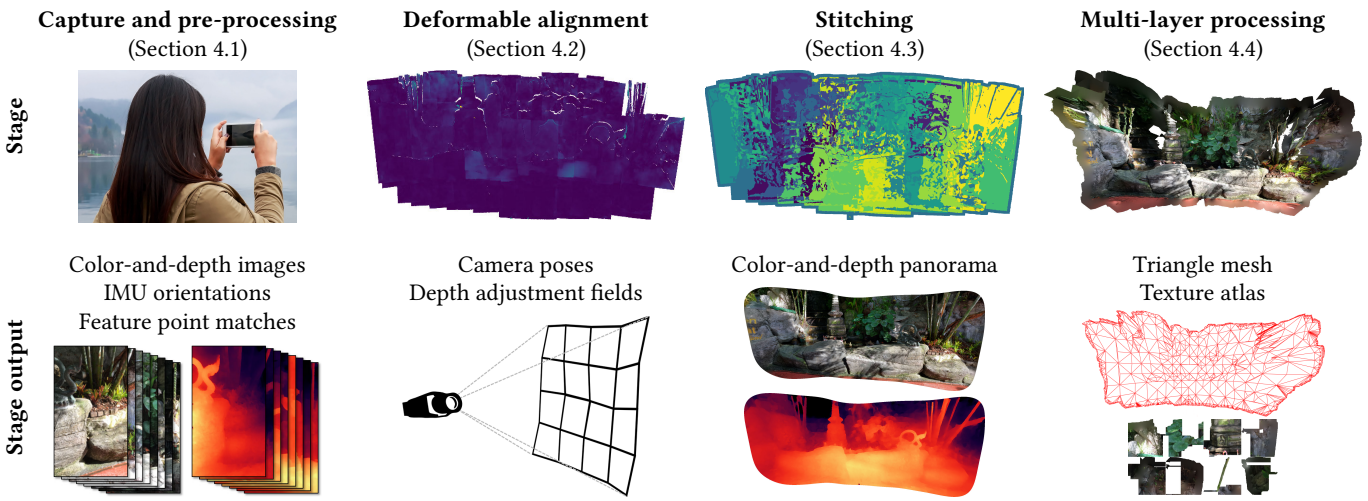


Fig. 2. Breakdown of the major algorithms stages and their outputs, which form the inputs to the next respective stage.

Multi-view Stereo: A long line of research in computer vision is concerned with producing depth maps or surface meshes from multiple overlapping images using multi-view stereo (MVS) algorithms [Seitz et al. 2006]. MVS algorithms are used in commercial photogrammetric tools for 3D reconstruction of VR scenes [Realities 2017; Valve 2016]. Huang et al. [2017] use MVS to obtain dense point clouds from video sequences captured with a single 360° camera.

MVS methods work best if the camera baseline is large, which is unfortunately not the case in the panorama capture scenario. In this case, it is difficult for the methods to deal with the triangulation uncertainty, which leads to artifacts, such as noisy reconstructions and missing regions. These methods are usually also slow, with runtimes ranging from minutes to hours. Hedman et al. [Hedman et al. 2017] improve the quality of reconstructed 3D panoramas, but their algorithm requires several hours of processing.

Light Fields: The light field representation [Gortler et al. 1996; Levoy and Hanrahan 1996] can generate highly realistic views of a scene with motion parallax and view-dependent effects. Recent work addresses unstructured acquisition with a hand-held camera [Davis et al. 2012]. The main disadvantage of this representation is that it requires a very large number of input views that need to be retained to sample from at runtime, and a custom rendering algorithm.

Bundle Adjustment with Depth: The popularisation of consumer depth cameras has inspired research on aligning and fusing multiple depth maps into globally consistent geometry [Izadi et al. 2011]. Dai et al. [Dai et al. 2017b] present system which integrates new depth maps in real-time, using bundle adjustment on 3D feature point correspondences to continuously maintain and refine alignment.

There has also been work on non-rigid deformations to refine alignment with active depth sensors. Zhou and Koltun [Zhou and Koltun 2014] perform 3D camera calibration during scanning, correcting for non-linear distortion associated with the depth camera. Whelan et al. [Whelan et al. 2015] show how to correct for drift by non-rigidly deforming the 3D geometry which has already been scanned.

In general, methods designed for depth cameras cannot directly be applied to narrow baseline stereo data, which is of much lower quality. Unlike the depth maps used in this paper, depth sensors provide absolute scale, maintain frame-to-frame consistency and can often be rigidly aligned to a high degree of accuracy.

3 OVERVIEW

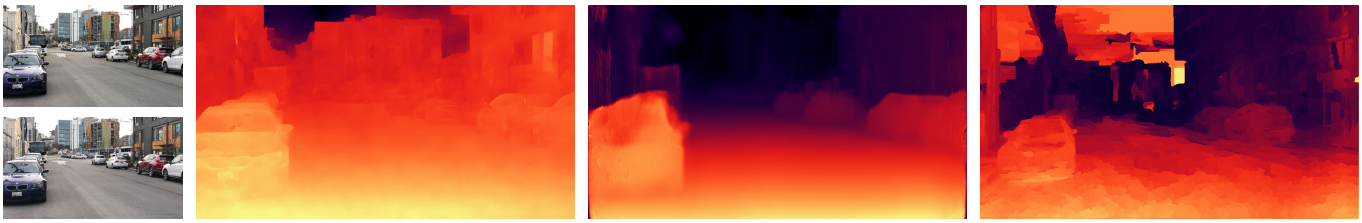
The goal of our work is to enable easy and rapid capture of 3D panoramas using readily available consumer hardware.

3.1 Dual Lens Depth Capture

Dual lens cameras capture synchronized small-baseline stereo image pairs for the purpose of reconstructing an aligned color-and-depth image using depth-from-stereo algorithms [Szeliski 2010]. The depth reconstruction is typically implemented in system-level APIs and highly optimized, so from a programmer's and a user's perspective, the phone effectively features a "depth camera". Several recent flagship phones feature dual cameras, including the iPhone 7 Plus, 8 Plus, X, and Samsung Note 8. Such devices are already in the hands of tens of millions of consumers.

The small baseline is both a blessing and a curse: the limited search range enables quickly establishing dense image correspondence but also makes triangulation less reliable and causes large uncertainty in the estimated depth. For this reason, most algorithms employ aggressive edge-aware filtering [Barron et al. 2015; He et al. 2010], which yields smoother depth maps with color-aligned edges, but large low-frequency error in the absolute depth values. In addition, the dual lenses on current-generation phones constantly move and rotate during capture due to optical image stabilization, changes in focus, and even gravity[1]. These effects introduce a non-linear and spatially-varying transformation of disparity that adds to the low-frequency error from noise filtering mentioned above.

[1] see http://developer.apple.com/videos/play/wwdc2017/507 at 17:20-20:50, Slides 81-89.



Stereo image pair (a) iPhone 7+ depth map (b) Monodepth [Godard et al. 2017] (c) DfUSMC depth map [Ha et al. 2016]

Fig. 3. Estimating depth maps using various algorithms. Note relative scale difference and low-frequency deformations between different maps. (a) Small baseline stereo depth computed by the native iOS algorithm on an iPhone 7+. (b) Single image CNN depth map [Godard et al. 2017]. (c) Depth from accidental motion result [Ha et al. 2016] (we actually used a short video clip to produce this result).

In Figure 3, you can see depth maps reconstructed using different stereo algorithms on this kind of data. As revealed in the figure, there is a significant amount of low-frequency error in the depth maps. Since our focus is not stereo matching, we use the depth maps from the native iPhone 7 Plus stereo algorithm for all of our experiments.

An important detail to note is that many small baseline stereo methods (including the one running on the iPhone) do *not* estimate absolute depth, but instead produce *normalized* depth maps. So, aligning such depth map involves estimating scale factors for each of them, or, in fact, sometimes even more complicated transformations.

3.2 Algorithm Overview

Our 3D panorama construction algorithm proceeds in four stages:

Capture (Section 4.1, Figure 2a): The input to our algorithm is a sequence of aligned color-and-depth image pairs, which we capture from a single vantage point on a dual lens camera phone using a custom burst capture app.

Deformable Depth Alignment (Section 4.2, Figure 2b): Due to the small camera baseline and resulting triangulation uncertainty, the input depth maps are not very accurate, and it is not possible to align them well using global transformations. We resolve this problem using a novel optimization method that jointly estimates the camera poses as well as spatially-varying adjustment maps that are applied to deform the depth maps and bring them into good alignment.

Stitching (Section 4.3, Figure 2c): Next, we stitch the aligned color-and-depth photos into a panoramic mosaic. Usually this is formulated as a labeling problem and solved using discrete optimization methods. However, optimizing label smoothness, e.g., using MRF solvers, is very slow, even when the problem is downscaled. We utilize a carefully designed data term and the high quality of our depth alignment, to replace label smoothness optimization with independently optimizing every pixel after filtering the data term in a depth-guided edge-aware manner. This achieves visually similar results with more than an order of magnitude speedup.

Multi-layer Mesh Generation (Section 4.4, Figure 2d): In the last stage, we convert the panorama into a multi-layered and textured mesh that can be rendered on any device using standard graphics engines. We tear the mesh at strong depth edges and extend the backside into the occluded regions, hallucinating new color and depth values in occluded areas. Finally, we simplify the mesh and compute a texture atlas.

4 ALGORITHM

4.1 Capture and Preprocessing

We perform all of our captures with an iPhone 7 Plus using a custom-built rudimentary capture app. During a scene capture session, it automatically triggers the capture of color-and-depth photos (using the native iOS stereo algorithm) at 1 second intervals.

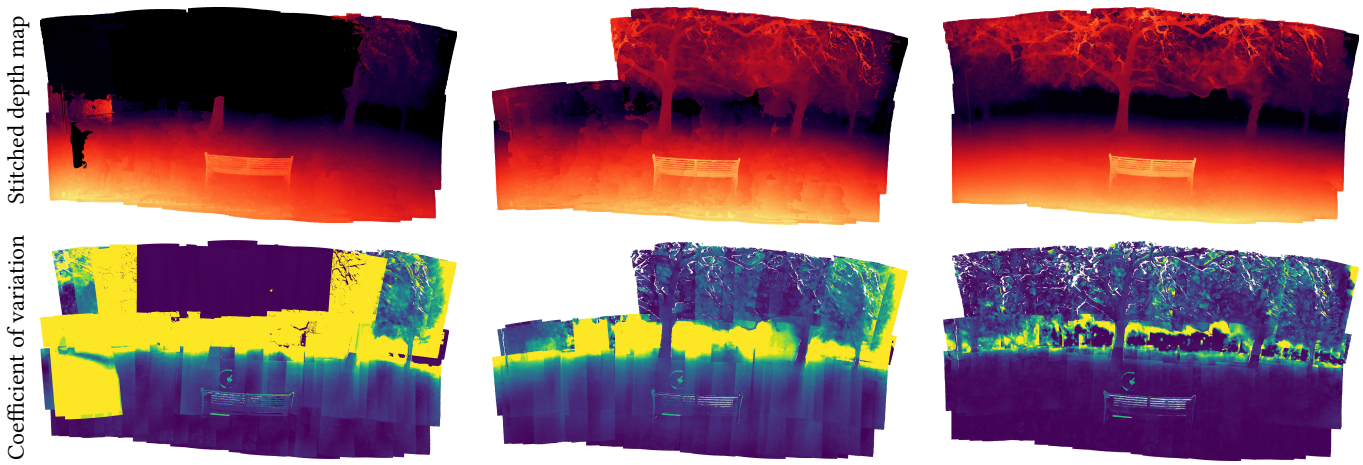
The capture motion resembles how people capture panoramas today: the camera is pointed outwards while holding the device at half-arms' length and scanning the scene in an arbitrary up-, down-, or sideways motion. Unfortunately, the field-of-view of the iPhone 7 Plus camera is fairly narrow in depth capture mode (37° vertical), so we need to capture more images than we would with other cameras. A typical scene contains between 20 and 200 images.

The captured color and depth images have 720×1280 pixels and 432×768 pixels resolution, respectively. We enable the automatic exposure mode to capture more dynamic range of the scene. Along with the color and depth maps, we also record the device orientation estimate provided by the IMU.

Feature extraction and matching: As input for the following alignment algorithm, we compute pairwise feature matching using standard methods. We detect Shi-Tomasi corner features [Shi and Tomasi 1994] in the images, tuned to be separated by at least 1% of the image diagonal. We then compute DAISY descriptors [Tola et al. 2010] at the feature points. We use the IMU orientation estimate to choose overlapping image pairs, and then compute matches using the FLANN library [Muja and Lowe 2009], taking care to discard outliers with a ratio test (threshold = 0.85) and simple geometric filtering, which discards matches whose offset vector deviates too much from the median offset vector (more than 2% of the image diagonal). All this functionality is implemented using OpenCV.

4.2 Deformable Depth Alignment

Our first goal is to align the depth maps. Since the images were taken from different viewpoints, we cannot deal with this in 2D image space due to parallax. We need to recover the extrinsic camera poses (orientation and location), so that when we project out the depth maps they align in 3D.



(a) Our global affine alignment (Eq. 4) (b) Global alignment to SFM point cloud (c) Our deformable alignment (Eq. 8)

Fig. 4. Aligning depth maps with low-frequency errors. We show stitches and the coefficient of variation (see text) for various methods. (a) Our algorithm with a global affine model (Eq. 4). Many depth maps got pushed to infinity. (b) Aligning each depth map *independently* with Eq 4 to a high-quality reconstruction. The result is better, but there are many visible seams and floaters due to the impossibility to fit the inaccurate depth maps with simple global transformations. (c) Our algorithm with the spatially-varying affine model yields excellent alignment.

4.2.1 Rigid alignment: We achieve this goal by minimizing the distance between reprojected feature point matches. Let f_A^i be a feature point in image A and $M = \{(f_A^i, f_B^i)\}$ the set of all matched pairs. We define a reprojection loss as follows:

$$E_{reprojection} = \sum_{(f_A^i, f_B^i) \in M} \rho\Big(\|P_{A\to B}(f_A^i) - f_B^i\|_2^2 \Big), \tag{1}$$

where $\rho(s) = \log(1+s)$ is a robust loss function to reduce sensitivity to outlier matches, and $P_{A\to B}(f)$ is a function that projects the 2D point f from image A to image B:

$$P_{A\to B}(f) = n\Big(R_B^\mathsf{T}\big(\overbrace{R_A\,\tilde{f}\,d_A(f) + t_A}^{\text{3D point in camera } A\text{'s coord}} - t_B\big)\Big), \tag{2}$$

3D point in world space

where $(R_A,\ tA)$ and $(R_B,\ t_B)$ are the rotation matrix and translation vectors for images A and B, respectively, $\tilde{f}$ is the homogeneous-augmented version of f, $d_A(f)$ is the value of image A's depth map at location f, and $n([x,y,z]^\mathsf{T}) = [\frac{x}{z}, \frac{y}{z}]^\mathsf{T}$. Note, that this formulation naturally handles the wrap-around in 360° panoramas.

Similar reprojection losses are common in geometric computer vision and have been used with great success in many recent reconstruction systems [Schönberger and Frahm 2016]. However, our formulation has a subtle but important difference: since we have depth maps, we do not need to optimize the 3D location of feature point correspondences. This significantly simplifies the system in several ways: (1) it drastically reduces the number of variables that need to be estimated, to just the camera poses, (2) we do not have to link feature point matches into long tracks, and (3) the depth maps helps reduce uncertainty, making our system robust to small baselines and narrow triangulation angles.

Eq. 2 assumes that the camera intrinsics as well as lens deformation characteristics are known and fixed throughout the capture. If this is not the case, extra per-camera variables could be added to this equation to estimate these values during the optimization.

Minimizing Eq. 1 w.r.t. the camera poses is equivalent to optimizing a rigid alignment of the depth maps. However, since most small baseline depth maps are normalized (including the ones produced by the iPhone), they cannot be aligned rigidly.

4.2.2 Global transformations: We resolve this problem by introducing extra variables that describe a global transformation of each depth map. Our first experiment was trying to estimate a scale factor s_A for each depth map, i.e., by replacing $d_A(f)$ in Eq. 1 with

$$d_A^{scale}(f) = s_A\, d_A(f), \tag{3}$$

where s_A is an extra optimization variable per image. However, this did not achieve good results, because, as we learned, many depth maps are normalized using unknown *curves*. We tried a variety of other classes of global transformations, and achieved the best results with an affine transformation in disparity space (i.e., $1/d$):

$$d_A^{affine}(f) = \Big(s_A\, d_A^{-1}(f) + o_A\Big)\Big)^{-1}, \tag{4}$$

s_A and o_A are per-image scale and offset coefficients, respectively.

Figure 4a shows a typical result of minimizing Eq 1 with the affine model. Many depth maps are incorrectly pushed at infinity, because the optimizer could not find a good way to align them otherwise. In the bottom row we visualize the coefficient of variation of depth samples per pixel, i.e., the ratio of the standard deviation to the mean. This is a scale-independent way of visualizing the amount of disagreement in the alignment. As a sanity check we also tried to independently align each depth map to a high quality SFM reconstruction of the scene (computed with COLMAP [Schönberger and Frahm 2016]) that can be considered ground truth (Figure 4b). Even with this "best-possible" result for the model the stitch is severely degraded by seams and floaters.

Through our experimentation we found that it is ultimately not possible to bring this kind of depth maps into good alignment using simple global transformations because of to the low frequency error that is characteristic for small baseline stereo depth maps due to the triangulation uncertainty.

4.2.3 Deformable alignment: Our solution to this problem is to estimate spatially-varying adjustment fields that *deform* each depth map and can therefore bring them into much better alignment. We modify the affine model in Eq. 4 to replace the global scale and offset coefficients with regular grids of 5×5 values that are bilinearly interpolated across the image.

$$d_A^{deform}(f) = \left(s_A(f)\, d_A^{-1}(f) + o_A(f)\right)^{-1}, \tag{5}$$

where $s_A(f) = \sum_i w_i(f)\, \hat{s}_A^i$, and $o_A(f) = \sum_i w_i(f)\, \hat{o}_A^i$, and $w_i(f)$ are bilinear interpolation weights at position f.

To encourage smoothness in the deformation field we add a cost for differences between neighboring grid values:

$$E_{smoothness} = \sum_A \sum_{(i,\,j)\in N} \left\|\hat{s}_A^i - \hat{s}_A^j\right\|_2^2 + \left\|\hat{o}_A^i - \hat{o}_A^j\right\|_2^2 \tag{6}$$

While $E_{reprojection}$ is agnostic to scale, $E_{smoothness}$ encourages setting the disparity scale functions $\hat{s}_A^i$ very small, which results in extremely large reconstructions. To prevent this, we add a regularization term that keeps the overall scale in the scene constant:

$$E_{scale} = \sum_A \sum_i \left(\hat{s}_A^i\right)^{-1} \tag{7}$$

The combined problem that we solve is:

$$\underset{\{R_I, t_I, \hat{s}_i, \hat{o}_i\}}{\text{argmin}} \quad E_{reprojection} + \lambda_1 E_{smoothness} + \lambda_2 E_{scale}, \tag{8}$$

with the balancing coefficients $\lambda_1 = 10^6$, $\lambda_2 = 10^{-4}$.

Figure 4c shows the improvement achieved by using the deformable model. The ground plane is now nearly perfectly smooth, there are no floaters, and thin structures such as the lamp post are resolved much better.

4.2.4 Optimization Details: Since Eq. 8 is a non-linear optimization problem, we require a good initialization of the variables. We initialize the camera rotations using the IMU orientations, and the locations by pushing them forward onto the unit sphere, i.e., $t_A = R_A \cdot [0, 0, 1]^\mathsf{T}$. We found it helpful to initialize the deformation field to enlarge the depth maps, i.e., $\hat{s}_A^i = 0.1$, $\hat{o}_A^i = 0$, because in this way reprojected feature points are visible in their matched images.

We use the Ceres library [Agarwal et al. 2017] to solve this non-linear least-squares minimization problem using the Levenberg-Marquardt algorithm. We represent all rotations using the 3-dimensional axis-angle parameterization and use Rodrigues' formula [Ayache 1989] when they are applied to vectors. The optimization usually converges within 50 iterations, which takes about 10 seconds.

4.2.5 Discussion: Due to the non-rigid transformations in the optimization the camera poses that we recover are not necessarily accurate anymore. We inspected the recovered poses visually and found that they qualitatively look similar to results obtained by SFM, but we have not performed a careful analysis to verify their degree of accuracy.

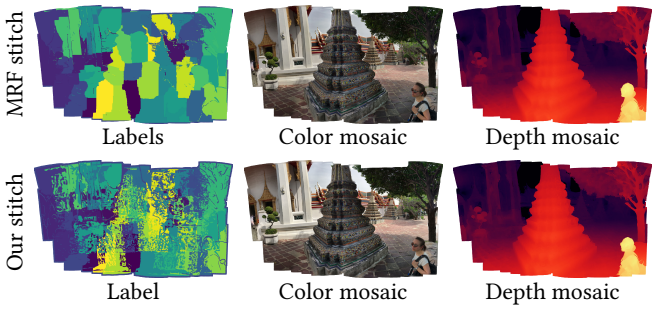


Fig. 5. Comparing stitching using MRF optimization (top row, runtime 3.25 minutes) vs. our algorithm (bottom row, runtime 0.5 seconds). While labels change more frequently in our solution, the color and depth mosaics are visually very similar to the MRF result.

We tried also even richer models than Eq. 4. In particular we have tried 3D grids that represent *bilateral space* adjustments with depth and luminance as range domains. However, while we found slight improvements in the results we did not deem it significant enough to warrant the extra complexity.

We also experimented with using 3D distance between matched features points' projection into world space in our loss. However, this did not work well, since a trivial solution is to shrink the scene until it vanishes.

Eq. 8 is quite robust and does not depend strongly on the initialization. We tried other initializations, e.g., setting $t_A = [0, 0, 0]^\mathsf{T}$, which worked fine as well. We have not encountered any scene in our experiments where the optimization got stuck in a poor local minimum.

4.3 Stitching

Now that we have 3D aligned depth photos, our next goal is to stitch them into a seamless panoramic mosaic. This enables removing outliers in the depth maps and also makes rendering faster by removing redundant content.

First, we compute a center of projection for the panorama by tracing the camera front vectors backwards and finding the 3D point that minimizes the distance to all of them. Then, we render all the color and depth maps from this central viewpoint into equirectangular panoramas (see supplementary document for details). The full panoramas are 8192 × 4096 pixels, though for each image, we only keep a crop to the tight bounding box of the pixels actually used.

As in previous work, we formulate the stitching as a discrete labeling problem, where we need to select for every pixel p in the panorama a source image α_p from which to fetch color and depth.

4.3.1 Data Term: A "good" source α_p for the target pixel p should satisfy a number of constraints, which we formulate as penalty terms.

Depth Consensus: If a source has the correct depth, it tends to be consistent with other views. Therefore, we count how many other views $n(p, \alpha_p)$ are at similar depth, i.e., their depth ratio is within $[0.9, 1.1]$, and define a depth consensus penalty, similar to earlier

work [Hedman et al. 2017]:

$$E_{consensus}(p, \alpha_p) = \max\left(1 - \frac{n(p, \alpha_p)}{\tau_{consensus}},\ 0\right), \tag{9}$$

where $\tau_{consensus} = 5$ determines how many other depth maps need to agree before the penalty reaches zero and we consider the labeling to be completely reliable.

Image Boundaries: We prefer pixels from the image center, because there the depth maps are more reliable and there is more space for seam-hiding feathering around them. We define an image boundary penalty $E_{boundary}(p, \alpha_p)$ that is set to 1 if a source pixel is close to the boundary in its original image (i.e., within 5% of the image width), and 0 otherwise.

Saturated Pixels: To maximize detail in the resulting panorama, we define a term that avoids overexposed source pixels:

$$E_{saturated}(p, \alpha_p) = \begin{cases} 1 & \text{if } l(p, \alpha_p) > \tau_{saturated} \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where $l(p, \alpha_p)$ is the luminance of a source pixel, and $\tau_{saturated} = 0.98$.

Combined Objective: By putting together the previous objectives we obtain the per-pixel data term:

$$E_{data} = E_{consensus} + \lambda_3 E_{boundary} + \lambda_4 E_{saturated}, \tag{11}$$

with the balancing coefficients $\lambda_3 = 1$, $\lambda_4 = 3$.

4.3.2 Optimization: Independently optimizing Eq. 11 for every pixel is fast, but yields noisy results since labels may change very frequently. The canonical way to achieving smoother results is to define a pairwise smoothness term that encourages fewer label changes that are placed in areas where they tend to be least visible. However, this makes the problem considerably harder and requires using slow MRF solvers. For example, Hedman et al. [2017] report runtimes of several minutes for solving a downscaled stitching problem.

We found that we can achieve very similar looking results faster with independent per-pixel optimization, by applying a variant of cost-volume filtering [Hosni et al. 2013] which first filters the data term with a depth-guided edge aware filter:

$$E_{soft\text{-}data}(p, \alpha_p) = \sum_{\Delta \in W_d^\alpha} w_d^\alpha(p, p + \Delta) \cdot E_{data}(p, \alpha_p). \tag{12}$$

However, instead of using a single global guide, we determine unique filter weights w_d^α for each source image α using a guided filter [He et al. 2010], guided by the normalized *disparities* in α. In our experiments, we use a filter footprint that spans 2.5% of the image width and set the edge-aware parameter $\epsilon = 10^{-7}$.

In Figure 5 we compare our result with an MRF solution using the color and disparity smoothness terms defined by Hedman et al. [2017]. While our stitch exhibits more frequent label changes the stitched color and depth mosaics are visually very similar.

4.3.3 Color Harmonization: Since we capture images with auto-exposure enabled, we need to align the exposures to create a seamless panorama. Following the insight from Reinhard et al. [2001], we convert the images to the channel-decorrelated CIELAB color space, and then process each channel independently. We solve a linear system to compute global affine color-channel adjustments (i.e., scale and offset) for each source image, such that the adjusted color values in the overlapping regions agree as much as possible. We further reduce visible seams by feathering the label region boundaries with a wide radius of 50 pixels. In a supplementary document we provide more implementation details.

(a) Naïve mesh (b) Naïve mesh + tears (c) Multi-layer mesh

Fig. 6. (a) Naïvely meshing by connecting all vertices yield stretched triangles at depth edges. (b) Tearing the mesh avoids this, but reveals holes. (c) Our multi-layer meshes extend the back-side at depth edges smoothly into the occluded region and reveal inpainted colors and depths.

4.4 Multi-layer Processing

The final step of our algorithm is to convert the panorama into a triangle mesh that can be rendered on any device using standard graphics engines. Naïvely creating a triangle mesh by connecting all pixels to their 4-neighbors yields stretched triangles at strong depth edges that are revealed when the viewpoint changes (Figure 6a). Our solution resembles somewhat the "two-layer merging" algorithm of Hedman et al. [2017]. However, an important difference is that our stitcher does not produce back-surface stitches, since the baseline is too small to reconstruct significant content in occluded regions (while we use similar camera trajectories the field of view of our camera is smaller, hence there is less overlap between images). If scenes were captured with a wider baseline and/or more wide-angle camera the two-layer stitch-and-merge algorithm mentioned above could be adapted at the expense of slower runtime.

In our algorithm, every mesh vertex corresponds to a pixel position in the panorama that is "pushed out" to a certain depth. Each vertex is connected to at most one neighbor in each of the 4 cardinal directions. Since our goal is to generate multiple layers, there can be multiple vertices at different depths for a single pixel position. We initialize the mesh by creating vertices for every panorama pixel and connecting them to their 4 neighbors.

We start the computation by detecting major depth edges in the mesh. Since the depth edges are soft and spread over multiple pixels, we apply a 9×9 median filter to turn them into step edges. Next, we tear the connection between neighboring vertices if their disparity differs by more than $5 \cdot 10^{-2}$ units. Sometimes the median filter produces small isolated "floating islands" at the middle of depth edges. We detect these using connected component analysis and merge them into either foreground or background, by replacing their depth with the median of depths just outside the floater. Figure 6b shows the mesh after tearing it at depth edges.

Table 1. Breakdown of the algorithm performance per stage for the scene from Figure 1.

Stage	Desktop Timing	Laptop Timing
Feature extraction and matching	6.6s	7.0s
Deformable alignment	9.8s	10.2s
Warping	6.6s	5.1s
Stitching	2.7s	2.6s
Color harmonization	1.6s	1.8s
Multi-layer computation	1.0s	1.1s
Mesh simplification	3.1s	3.5s
Texture atlas generation	3.3s	3.7s
Total	**34.7s**	**35.0s**

Next, we hallucinate new content in occluded parts by iteratively growing the mesh at its boundaries. In every iteration, each vertex that is missing a connection in one of the 4 cardinal directions grows in this direction and generates a new vertex at the same depth as itself. We connect new vertices with all neighboring boundary vertices whose disparity is within the threshold above. After running this procedure for a fixed number of 30 iterations, we prune the newly generated vertices by removing any but the *furthest* generated vertex at every pixel location. If the remaining generated vertex is in front of the original stitch we remove it as well. We synthesize colors for the newly generated mesh parts using diffuse inpainting. The resulting mesh smoothly extends the back-side around depth edges into the occluded regions. Instead of stretched triangles or holes, viewpoint changes now reveal smoothly inpainted color and depth content (Figure 6c).

In a supplementary document we provide some implementation details about simplification and texture atlas generation for the final mesh.

5 RESULTS AND EVALUATION

We have captured and processed dozens of scenes with an iPhone 7 Plus. 25 of these are included in this submission, see Figure 7, as well as the accompanying video and the supplementary material. These scenes span a wide range of different environments (indoor and outdoor, urban and natural) and capture conditions (day and night, bright and overcast). The scenes we captured range from about 20 to 200 source images, and their horizontal field-of-view ranges from 60° to 360°.

5.1 Performance

All scenes were processed using a PC with 3.4 GHz 6-core Intel Xeon E5-2643 CPU and a NVIDIA Titan X GPU and 64 GB of memory. Our implementation mostly consists of unoptimized CPU code. The GPU is currently only (insignificantly) used in the warping stage. We ran our system also on a slower 14" Razer Blade laptop with a 3.3 GHz 4-core Intel i7-7700HQ CPU and a NVIDIA GTX 1060 GPU. Interestingly, the warping stage performs faster on the laptop, most likely because CPU computation and CPU/GPU transfers dominate the runtime. Table 1 breaks out the timings for the various algorithm stages on both of these systems for an example scene. While our algorithm already runs fast, we note that there are significant further optimizations on the table. Since the deformable alignment has proven to be quite robust, we could replace the feature point detector and descriptor with faster to compute variants, e.g., FAST features and BRIEF descriptors. The alignment optimization could be sped up by implementing a custom solver, tailored to this particular problem. Our current warping algorithm is implemented in a wasteful way. Properly rewriting this GPU code would make this operation practically free. The stitching algorithm could be reimplemented on the GPU.

5.2 Alignment

Figure 8 shows a quantitative evaluation of our alignment algorithm. We processed the 25 scenes in Figure 7 using different variants of the algorithm and evaluate the average reprojection error (Eq. 1).

We also evaluate the effect of varying the grid size of our deformable model, which shows that the reprojection error remains flat across a wide range of settings around our choice of 5×5.

5.3 Single-image CNN Depth Maps

We experimented with other depth map sources. In particular, we were interested in using our algorithm with depth maps estimated from single images using CNNs, since this would enable using our system with regular *single*-lens cameras (even though the depth map quality produced by current algorithms is low). In Figure 9 we used the Monodepth algorithm [Godard et al. 2017] to generate single-image depth maps for three of our scenes. The original Monodepth algorithm works well for the first street scene, since it was trained on similar images. For the remaining two scenes we retrained the algorithm with explicit depth supervision using the RGBD scans in the ScanNet dataset [Dai et al. 2017a]. Even though the input depth maps are considerably degraded our method was able to reconstruct surprisingly good results. To better appreciate the result quality, see the video comparisons in the supplementary material.

5.4 SFM and MVS Comparison

We were interested in how standard SFM algorithms would perform on our datasets. When processing our 25 datasets with COLMAP's SFM algorithm 7 scenes failed entirely, in 7 more not all cameras were registered, and there was 1 were all cameras registered but the reconstruction was an obvious catastrophic failure. This high failure rate underscores the difficulty of working with small baseline imagery.

We also compare against end-to-end MVS systems, in particular the commercial Capturing Reality system[2] and Casual 3D [Hedman et al. 2017]. Capturing Reality's reconstruction speed is impressive for a full MVS algorithm, but due to the small baseline it is only able to reconstruct foreground. Casual 3D produces results of comparable quality to ours, but at much slower speed. Figure 10 shows an example scene, and the supplementary material contains video comparisons of more scenes.

[2] https://www.capturingreality.com

Fig. 7. Datasets that we show in the paper, video, and supplementary material. The two bottom rows show 360° panoramas.

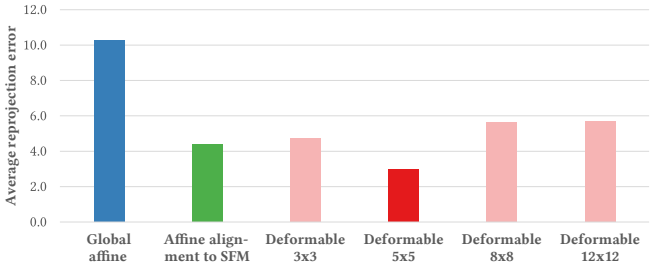


Fig. 8. Average reprojection error (Eq. 1, *without* the robust loss function) for different alignment methods, as well as our deformable alignment with different grid sizes.

5.5 Parallax-aware Stitching

We compared our algorithm with two monocular stitching algorithms that handle parallax in different ways. As-projective-as-possible warping (APAP) [Zaragoza et al. 2013] allows local deviations from otherwise globally projective warps to account for misalignment (Figure 11a). Note, that this does not always succeed (see detail crop in Figure 11c). Microsoft ICE[3] uses globally projective warps, but leverages carefully engineered seam finding and blending to hide parallax errors (Figure 11b and detail crop in 11d). Neither of these methods produce a depth panorama. While the source depth pixels could be stitched according to the label maps produced by these algorithms, this would not lead to good results, because the source depth maps are inconsistent and show the scene from different vantage points. Our algorithm resolves these inconsistencies and produces a coherent color-and-depth panorama (Figure 10c and detail crop in 11e).

[3]https://www.microsoft.com/en-us/research/product/computational-photography-applications/image-composite-editor/

5.6 Capture without Parallax

We evaluated the effect of varying the amount of parallax in the input images by capturing scenes while rotating the camera around the optical center without translating (as much as was possible), and comparing against a normal capture where we move the camera at half-arm's length. The resulting panoramas are visually very similar (see supplementary material). That said, theoretically our method should break down in the complete absence of parallax because the reprojection error in Equation 1 will become invariant to depth in this case. In practice, however, it is very difficult to completely avoid any parallax in the capture, and, fortunately, the natural way to capture panoramas is on an arc with a radius of about a half arm's length anyway.

5.7 Limitations

Our algorithm has a variety of limitations that lead to interesting avenues for future work.

Capture: The iPhone camera has a very narrow field-of-view in depth capture mode, because one of the lenses is wide-angle and the other a telephoto lens. If both lenses were wide-angle we would need to capture considerably fewer images to achieve the same

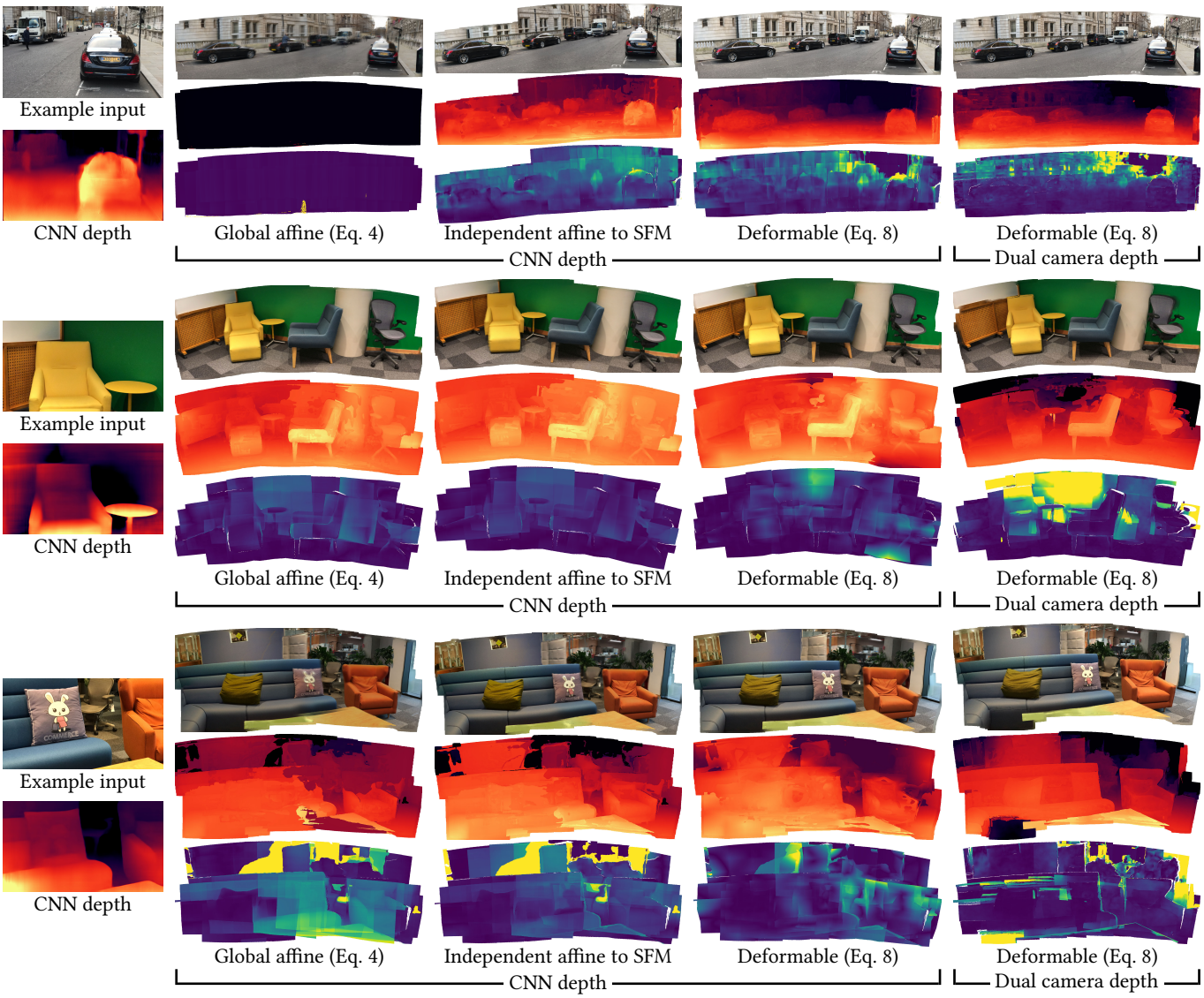


Fig. 9. Applying our algorithm to single-image CNN depth maps (3 middle columns), and comparing to a result for dual camera depth maps (right column). See the supplementary material for videos of the results.

amount of overlap. At the same time the baseline would increase, making the reconstruction problem easier.

Artifacts: Our results exhibit similar artifacts as other 3D reconstruction systems. In particular, these are floating pieces of geometry, incorrect depth in untextured regions, artifacts on dynamic objects. Compared to existing systems these problems are reduced (Figure 10), but they are still present. To examine these artifacts carefully we suggest watching the video comparison in the supplementary material.

Multi-layer processing: The hallucination of occluded pixels is rudimentary. In particular the simple back-layer extension algorithm leaves room for improvement. We plan to improve the inpainting of colors using texture synthesis.

Parameters: Like many end-to-end reconstruction algorithms we depend on many parameters. We proceeded one stage at a time, examining intermediate results, when tuning the parameters. All results shown anywhere in this submission or accompanying materials use the same parameter settings, provided in the paper.

6 CONCLUSIONS

In this paper we have presented a fast end-to-end algorithm for generating 3D panoramas from a sequence of color-and-depth images. Even though these depth maps contain a considerable amount

(a) Capturing Reality (2.1 minutes) (b) Casual 3D (1.8 hours) (c) Our result (34.7 seconds)

Fig. 10. Comparison against MVS systems: (a) Capturing Reality processes fast, but the reconstruction breaks down just a few meters away from the vantage point due to triangulation uncertainty. (b) Casual 3D produced a high quality result, but it is slow. (c) Our result has even better quality, and was computed over **200**× faster.

(a) APAP [Zaragoza et al. 2013] (b) Microsoft ICE (c) APAP detail (d) ICE detail (e) Our detail

Fig. 11. Comparison against monocular panoramas stitched with (a) As-Projective-As-Possible warping, and (b) Microsoft ICE. Note that these algorithms do not produce a stitched depth map. (c)-(d) show a detail crops from before mentioned algorithms. (e) corresponding detail crop from our result in Figure 10c.

of low-frequency error, our novel deformable alignment optimization is able to align them precisely. This opens up the possibility to replace discrete smoothness optimization in our stitcher with depth-guided edge-aware filtering of the data term and independently optimizing every pixel, achieving two orders of magnitude speedup.

We are excited about the many avenues for further improvement and research that this work opens up. Considering the performance discussion in Section 5.1 we believe a near-interactive implementation directly on the phone is within reach.

We have seen already how the availability of a fast capture and reconstruction system has changed our own behavior with respect to 3D scene capture. The way we capture scenes has become more opportunistic and impulsive. Almost all of the provided scenes have been captured spontaneously without planning, e.g., while traveling.

We are particularly excited about the promising first result using single-image CNN depth maps. The quality of these depth maps is still quite low. Yet, our alignment algorithm was able to conflate them, and stitching them using the consensus data term reduced artifacts further. Improving these results further is an interesting direction for further research and holds the promise of bringing instant 3D photography to billions of regular cell phones with monocular cameras.

ACKNOWLEDGEMENTS

The authors would like to thank Clément Godard for generating the depth maps for the CNN evaluation and Suhib Alsisan for developing the capture app.

REFERENCES

Sameer Agarwal, Keir Mierle, and Others. 2017. Ceres Solver. http://ceres-solver.org. (2017).

Robert Anderson, David Gallup, Jonathan T. Barron, Janne Kontkanen, Noah Snavely, Carlos Hernandez Esteban, Sameer Agarwal, and Steven M. Seitz. 2016. Jump: Virtual Reality Video. *ACM Transactions on Graphics* 35, 6 (2016).

Nicholas Ayache. 1989. *Vision Stéréoscopique et Perception Multisensorielle: Application à la robotique mobile.* Inter-Editions (MASSON). https://hal.inria.fr/inria-00615192

Jonathan T. Barron, Andrew Adams, YiChang Shih, and Carlos Hernández. 2015. Fast Bilateral-Space Stereo for Synthetic Defocus. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), 4466–4474.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017a. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2017).

Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. 2017b. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics 2017 (TOG)* 36, 3 (2017), Article no. 24.

Abe Davis, Marc Levoy, and Fredo Durand. 2012. Unstructured Light Fields. *Computer Graphics Forum (Proc. EUROGRAPHICS 2012)* 31, 2pt1 (2012), 305–314.

Facebook. 2016. Facebook Surround 360. https://facebook360.fb.com/facebook-surround-360/. (2016). Accessed: 2016-12-26.

Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. 2017. Unsupervised Monocular Depth Estimation with Left-Right Consistency. *CVPR* (2017).

Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. 1996. The Lumigraph. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), 43–54.

Hyowon Ha, Sunghoon Im, Jaesik Park, Hae-Gon Jeon, and In So Kweon. 2016. High-quality Depth from Uncalibrated Small Motion Clip. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

Kaiming He, Jian Sun, and Xiaoou Tang. 2010. Guided Image Filtering. *Proceedings of the 11th European Conference on Computer Vision (ECCV)* (2010), 1–14.

Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. 2017. Casual 3D Photography. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2017)* 36, 6 (2017), Article no. 234.

Aasma Hosni, Christoph Rhemann Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. 2013. Fast Cost-Volume Filtering for Visual Correspondence and Beyond. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 2 (2013), 504–511.

Jingwei Huang, Zhili Chen, Duygu Ceylan, and Hailin Jin. 2017. 6-DOF VR Videos with a Single 360-Camera. *IEEE VR 2017* (2017).

Sunghoon Im, Hyowon Ha, François Rameau, Hae-Gon Jeon, Gyeongmin Choe, and In So Kweon. 2016. All-Around Depth from Small Motion with a Spherical Panoramic Camera. *European Conference on Computer Vision (ECCV '16)* (2016), 156–172.

Hiroshi Ishiguro, Masashi Yamamoto, and Saburo Tsuji. 1990. Omni-directional stereo for making global map. *Third International Conference on Computer Vision* (1990), 540–547.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. *Proceedings of the 24th Annual ACM*

Symposium on User Interface Software and Technology (2011), 559–568.
Robert Konrad, Donald G. Dansereau, Aniq Masood, and Gordon Wetzstein. 2017. SpinVR: Towards Live-streaming 3D Virtual Reality Video. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2017)* 36, 6 (2017), article no. 209.
Jungjin Lee, Bumki Kim, Kyehyun Kim, Younghui Kim, and Junyong Noh. 2016. Rich360: Optimized Spherical Representation from Structured Panoramic Camera Arrays. *ACM Transactions on Graphics* 35, 4 (2016), article no. 63.
Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), 31–42.
Kaimo Lin, Nianjuan Jiang, Loong-Fah Cheong, Minh N. Do, and Jiangbo Lu. 2016. SEAGULL: Seam-Guided Local Alignment for Parallax-Tolerant Image Stitching. *14th European Conference on Computer Vision (ECCV)* (2016), 370–385.
Marius Muja and David G. Lowe. 2009. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *International Conference on Computer Vision Theory and Application VISSAPP'09)* (2009), 331–340.
Shmuel Peleg and Moshe Ben-Ezra. 1999. Stereo panorama with a single camera. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1999)* (1999), 395–401.
Shmuel Peleg, Moshe Ben-Ezra, and Yael Pritch. 2001. Omnistereo: panoramic stereo imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 3 (2001), 279–290.
F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang, S. Watson, and M. Gross. 2015. Panoramic Video from Unstructured Camera Arrays. *Computer Graphics Forum* 34, 2 (2015), 57–68.
Realities. 2017. realities.io | Go Places. http://realities.io/. (2017). Accessed: 2017-1-12.
Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. 2001. Color Transfer Between Images. *IEEE Comput. Graph. Appl.* 21, 5 (2001), 34–41.
Christian Richardt, Yael Pritch, Henning Zimmer, and Alexander Sorkine-Hornung. 2013. Megastereo: Constructing High-Resolution Stereo Panoramas. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2013)* (2013), 1256–1263.
Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).
Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* 1 (2006), 519–528.
Jianbo Shi and Carlo Tomasi. 1994. Good Features to Track. *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)* (1994), 593 – 600.
Richard Szeliski. 2010. *Computer Vision: Algorithms and Applications* (1st ed.). Springer-Verlag New York, Inc., New York, NY, USA.
Engin Tola, Vincent Lepetit, and Pascal Fua. 2010. DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. *IEEE Trans. Pattern Anal. Mach. Intell.* 32, 5 (2010), 815–830.
Valve. 2016. Valve Developer Community: Advanced Outdoors Photogrammetry. https://developer.valvesoftware.com/wiki/Destinations/Advanced_Outdoors_Photogrammetry. (2016). Accessed: 2016-11-3.
Thomas Whelan, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison. 2015. ElasticFusion: Dense SLAM Without A Pose Graph. *Proceedings of Robotics: Science and Systems* (2015).
Julio Zaragoza, Tat-Jun Chin, Michael S. Brown, and David Suter. 2013. As-Projective-As-Possible Image Stitching with Moving DLT. *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013), 2339–2346.
Fan Zhang and Feng Liu. 2014. Parallax-Tolerant Image Stitching. *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), 3262–3269.
Fan Zhang and Feng Liu. 2015. Casual Stereoscopic Panorama Stitching. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)* (2015), 2002–2010.
Ke Colin Zheng, Sing Bing Kang, Michael F. Cohen, and Richard Szeliski. 2007. Layered Depth Panoramas. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007)* (2007), 1–8.
Qian-Yi Zhou and Vladlen Koltun. 2014. Simultaneous Localization and Calibration: Self-Calibration of Consumer Depth Cameras. *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), 454–460.