Monocular Depth from Small Motion Video Accelerated

Christopher Ham[1], Ming-Fang Chang[2], Simon Lucey[2], Surya Singh[1]

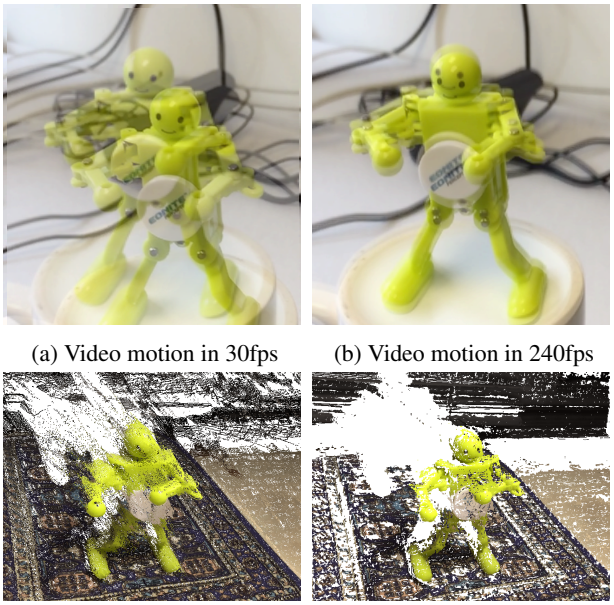
The University of Queensland[1],
St Lucia Qld 4072, Australia
{c.ham,spns}@uq.edu.au

Carnegie Mellon University[2]
5000 Forbes Ave, Pittsburgh, PA
15213, United States
{mingfanc,slucey}@andrew.cmu.edu

Abstract

We propose a novel four-stage pipeline densely reconstructing depth from video sequences with small baselines with the goal of being fast. This work is particularly motivated by the high frame rate (HFR) ability of many modern smartphones which leads to smaller inter-frame motion and less motion blur, but with more image noise as the sensitivity is adjusted to compensate for the increased shutter speed. While small baselines lead to larger uncertainties in depth estimations, they allow for easier point tracking and assumptions of brightness constancy hold more true. In our pipeline we make use of the sub-pixel precision of direct photometric bundle adjustment to reduce the number of tracked points required to estimate an accurate pose. We show that by considering pixel intensities across the entire video, it is more robust to image noise. Instead of using the exhaustive plane sweeping approach of existing small baseline methods, dense depth maps are calculated efficiently using an algorithm inspired by PatchMatch[2, 3]. Instead of a stereo matching error, our algorithm minimizes the variance over multiple frames with a robust mean estimation.

We present a detailed, quantitative comparison between our method and the most recent small baseline method from Ha et al. [7] for speed and quality across and spectrum of baselines and sequence sizes using a synthesized, photorealistic dataset. The results suggest that our method has the ability to cope with a wider range of baselines and sequence sizes. We also compare qualitative results on real small motion clips from Ha et al. [7] in addition to our own, and show that our method outputs dense depth maps of similar or better quality and at least 10 times faster.



(a) Video motion in 30fps (b) Video motion in 240fps

(c) Ha et al. [7] (d) Ours

Figure 1: In this paper, we propose a 3D reconstruction pipeline for small baseline video sequence as (b). Experiment shows that our reconstruction result has better or similar quality than [7], as shown in (c)(d) and achieves an order of mangitute speedup.

1. Introduction

When reconstructing dense depth maps we are recovering the lost dimension in photographs that is the fundamental driving force behind the 3D vision community - one is regaining important information about the world that cannot be captured by 2D pixels. Smart devices have the potential to bring this power into the hands of many and lead to exciting Augmented Reality applications. Particularly, we are motivated by the high frame rate (HFR) capabilities of modern devices. Short HFR sequences are an attractive source for recovering dense depth maps since assumptions on brightness constancy hold more strongly, they observe fewer occlusions and less geometric distortion, and they can

be obtained from small motions such as slight hand-held motion.

While not designed to explicitly return depth maps, one might turn to multi-view stereo (MVS) or structure from motion (SfM) pipelines; however most are designed for wide baseline DSLR images. The few that are designed for mobile devices ([15, 14], 123D Catch[1]) are either closed source or not even publicly available and require a lot of time and care by the user– taking anywhere upwards of five minutes.

Alternatively there has been recent interest in estimating dense depths from small and incidental motion videos [17, 7]. These works, however, have focused heavily on obtaining qualitatively pleasing results for applications such as synthetic refocusing and object segmentation rather than metric reconstruction accuracy. In addition, Ha et al. describes in [7] that it takes more than 10 minutes to process 1280×720 full resolution image without relying on GPU computation.

The striking results presented in [7] rely on a refinement step that is readily be applied to *any* estimated depth map. With object scanning applications in mind, our focus in this paper is to recover a more accurate raw result.

We propose a novel four-stage pipeline for rapid depth map reconstruction from video clips: (i) interest point tracking; (ii) geometric bundle adjustment; (iii) photometric bundle adjustment (PBA); and (iv) dense depth reconstruction. The sub-pixel precision of PBA in stage (iii) allows our method to estimate poses of comparable accuracy to geometric bundle adjustment while using fewer points. In stage (iv) dense depths are calculated by an approach inspired by PatchMatch [3] but designed for multiple frames. Depth maps are obtained with considerably less computation than the exhaustive methods used by Ha et al. [7] and Yu and Gallup [17], nor are they discretized.

The pipeline is designed to significantly reduce computational costs compared to existing methods while retaining or even improving final reconstruction quality. When performing bidirectional KLT tracking, our implementation reuses the patch Hessian calculated for the reference frame. In the bundle adjustment we apply a damping to the depth and intrinsic parameters at the start – making the system more stable to solve. This reduces risk for overshooting and makes our optimization converge faster.

2. Prior Art

Small baseline video has drawn some attention in recent years. Work by Yu and Gallup [17], Ha et al. [7], and Joshi and Zitnick [10] focus on reconstructing dense depth maps from small baseline motions. Joshi and Zitnick [10] go one step further and show they can recover a dense depth map from tremors in a tripod-mounted camera. [17] and [7] take cues from structure from motion (SfM) in that they first track feature points, then estimate camera poses using a bundle adjustment, then finally reconstruct a dense depth map. Ha et al. [7] outperforms Yu and Gallup [17] by using an exhaustive plane sweeping search for the optimum depth at each pixel, filtering for likely occlusions, and correcting the depth map using Yang's non-local cost aggregation [16]. Although the final refinement step returns impressive looking depth maps, ideal for synthetic refocusing, the result is no longer metric having a tendency to flatten surfaces under its fronto-parallel assumption, and it can be also applied to our pipeline seamlessly if necessary. In this paper, when comparing reconstruction quality, we use filtered (yet unrefined) output from [7] which has regions of low confidence removed.

Joshi and Zitnick [10] have a similar multi-stage pipeline, however, they perform a dense plane-based optical flow to track across frames before using RANSAC to compute the projection matrices of each frame. This allows the algorithm to perform well for micro baselines but, according to our experiments, the added computational expense is unnecessary for small *handheld* motions in HFR videos.

In contrast to small baselines, Newcombe et al. [12] and Wendel et al. [15] propose methods that estimate live dense depth maps for SLAM systems. However, they rely on GPU implementations to obtain such performance, and expect larger baselines than those explored in this work, so less care is required when solving for pose and structure.

The PatchMatch algorithm and its variations [2, 3] offer a computationally efficient way to estimate dense depths for stereo baselines. At its core it takes advantage of the large number of pixels to randomly sample the space. While no explicit spacial smoothing is employed, it is the assumption that neighboring pixels are likely to have a similar depth that makes this algorithm so effective.

Taking cues from Ha et al. [7], we use a distorted-to-undistorted (D-U) radial distortion model when solving for the lens distortion in both the geometric bundle adjustment, and direct photometric bundle adjustment refinement. As shown by Tamaki et al. [13] and Fitzgibbon [6], this parameterization is equally valid to the more popular undisorted-to-distorted (U-D) model, with the choice coming down to whichever will be the most computationally efficient as inverting this function comes with its own challenges.

3. Reconstruction Pipeline

For the task of reconstructing dense depth maps from small baseline motions, we use a four stage pipeline: (i) sparse interest point tracking; (ii) sparse geometric bundle adjustment; (iii) sparse photometric bundle adjustment; (iv) dense depth reconstruction with the custom cost function.

[1] http://www.123dapp.com/catch

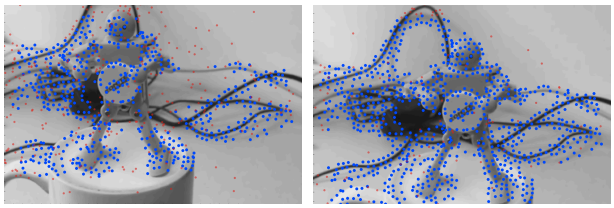


Figure 2: Point tracking result. Points tracked successfully are labeled by blue.

Notation. Lower-case non-bold letters are scalars (x). Lower-case bold letters are vectors ($\mathbf{x}$). Upper-case bold letters are matrices ($\mathbf{X}$). Tilde accents indicate the homogeneous form of a vector ($\tilde{\mathbf{x}} = \begin{bmatrix}\mathbf{x}\\1\end{bmatrix}$). We parameterize the structure of the scene using inverse depths which we represent as $d = \frac{1}{z}$.

3.1. Point Tracking

The bundle adjustment stage requires a set of feature correspondences across the sequence. We extend the Kanade-Lucas-Tomasi (KLT) tracker [11] in order to perform fast bidirectional tracking. Using patches from reference image, points are tracked forward from the latest estimate. It is then tracked back to the reference image. Points with a bidirectional error greater than 0.1 pixels are labeled as a failed track for that frame. Point tracking result is shown in Figure 2.

Bidirectional tracking is an effective way to reject outliers, but it can be slow as it must estimate a new Hessian for each point in each frame. In our implementation we approximate the new Hessians with those already obtained in the reference frame. We observe significant speed-ups with negligible effect on the tracking robustness.

Additionally, we increase the likelihood of successful tracks over wider baselines (compared to Ha et al. [7]) in the following ways

- The estimated position of a point from the previous frame is used as the initialization for the current frame;
- An affine warp estimated from good tracks in the previous frame is applied to the patch before tracking, increasing robustness to geometric appearance change;
- Failed tracks are kept and still updated according to nearest the 10 neighbors, increasing the likelihood of a successful retrack in the next frame.

Finally, any points that have failed tracking on more than a quarter of the sequence are totally rejected before passing on to the bundle adjustment.

3.2. Bundle Adjustment

We choose bundle adjustment to jointly optimizes camera poses, inverse depth of tracked points, camera distortion parameters and focal length, so that we don't need to calibrate camera focal length beforehand. This stage is critical for estimating the camera intrinsics and establishing good initial poses and depths for photometric refinement. Ha et al. [7] propose a bundle adjustment formulation capable of estimating focal length, and the first and second order radial distortion intrinsics. In their implementation the Ceres Solver [1] is used which employs higher level routines to optimize the system reliably, but comes at the cost of time.

To reduce the runtime of the bundle adjustment, we optimize by solving for an analytically derived Gauss-Newton system. In order to solve the system reliably, we apply a damping reminiscent of the Levenberg-Marquardt algorithm (LMA) but only targeting the inverse depths and camera intrinsics. By not damping the pose parameters, we find that the solution converges more quickly than when using full LMA. After each iteration the damping factor is reduced by a factor of 0.9, which allows the optimization to focus more on the poses initially before refining the depths and camera intrinsics.

In the following, we assume that all image coordinates are relative to the image center. As in [7], we calculate the residuals as the reprojection error in the *undistorted* image domain. This means that we use a distorted-to-undistorted (D-U) model, rather than the popular undistorted-to-distorted model (U-D), which is equally valid as shown in [13, 6]. We define the function α that radially undistorts a point by

$$\alpha(\mathbf{x}) = 1 + k_1\|\mathbf{x}\|^2 + k_2\|\mathbf{x}\|^4, \tag{1}$$

where k_1 and k_2 are the first and second order radial distortion parameters respectively.

We define π, the projection function which projects a point in the reference frame to a target frame, by

$$\pi(\mathbf{x}, \boldsymbol{\omega}, \mathbf{t}, d) = \langle \mathcal{R}(\boldsymbol{\omega})\mathcal{N}(\mathbf{x}) + d\mathbf{t} \rangle, \tag{2}$$

$$\mathcal{N}(\mathbf{x}) = \begin{bmatrix} \frac{1}{f}\alpha\left(\frac{\mathbf{x}}{f}\right)\mathbf{x} \\ 1 \end{bmatrix}, \tag{3}$$

$$\mathcal{R}(\boldsymbol{\omega}) = \begin{bmatrix} 1 & -\omega_z & \omega_y \\ \omega_z & 1 & -\omega_x \\ -\omega_y & \omega_x & 1 \end{bmatrix}, \tag{4}$$

$$\left\langle \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right\rangle = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix}, \tag{5}$$

where f represents the focal length; $\boldsymbol{\omega}, \mathbf{t} \in \mathbb{R}^3$ indicate the rotation and translation of a frame with respect to the reference; $\mathcal{N}$ transforms an image point into normalized, ho-

mogeneous image coordinates; $\mathcal{R}$ returns a rotation matrix under a small angle approximation.

For M frames and N points, the energy of the system is defined by following formula, where $\|.\|_\gamma$ indicates the Huber norm,

$$E_{\text{BA}} = \sum_{i=1}^{M}\sum_{j=1}^{N} \|r_{ij}\|_\gamma \tag{6}$$

$$r_{ij} = \mathbf{x}_{ij}\alpha(\tfrac{1}{f}\mathbf{x}_{ij}) - f\pi(\mathbf{x}_{0j}, \boldsymbol{\omega}_i, \mathbf{t}_i). \tag{7}$$

We construct and solve the Gauss-Newton system by

$$\mathbf{W} = \text{diag}(\mathbf{w}_\gamma + \mathbf{w}_\lambda), \tag{8}$$

$$\mathbf{H} = \mathbf{J}^\intercal\mathbf{W}\mathbf{J}, \tag{9}$$

$$\mathbf{g} = \mathbf{J}^\intercal\mathbf{W}\mathbf{r}, \tag{10}$$

$$\Delta\mathbf{p} = \mathbf{H}^{-1}\mathbf{g}, \tag{11}$$

$$\mathbf{p}_k := \mathbf{p}_{k-1} + \Delta\mathbf{p}, \tag{12}$$

$$\mathbf{w}_\lambda := 0.9\mathbf{w}_\lambda \tag{13}$$

where $\mathbf{r} \in \mathbb{R}^{MN}$ is the vector of residuals; $\mathbf{J} \in \mathbb{R}^{MN\times P}$ is the Jacobian of $\mathbf{r}$; $\mathbf{p} \in \mathbb{R}^P$ is the stacked vector of the three instrinsic parameters (f, k_1, k_2), M camera poses and N inverse depths with $P = 3 + 6M + N$ parameters; and $\mathbf{W} \in \mathbb{R}^{MN\times MN}$ is a diagonal matrix of weights from the Huber loss function, $\mathbf{w}_\gamma$, and the additional factor λ for damping intrinsic and inverse depth update, $\mathbf{w}_\lambda$. The Hessian is constructed as a sparse matrix and the system in (12) is solved using SciPy's sparse solve method[2].

3.3. Photometric Bundle Adjustment

We take inspiration from the pioneering work of Engel et al. [5, 4] for photometric bundle adjustment, who show that only a sparse subset of textured points are required to accurately recover camera poses and point depths.

In this stage, we refine the camera poses and depths by directly minimizing the intensity errors of patches projected from the reference image to other images in the sequence. In addition to providing sub-pixel alignment of frames, a photometric bundle adjustment allows us to consider the reprojected intensity errors across multiple images. This is important when we consider that in reality we observe frame to frame appearance variation due to image noise independent of changes in geometry. It also refines by pushing to sub-pixel level accuracy.

Interest point tracking estimates each patch at each frame independently. For small baselines, bundle adjustment of pose and inverse depths can become more sensitive to small perturbations that occur due to image noise. Ha et al. [7] compensate for this kind of error by tracking a large number of interest points (upwards of 2000 in their sequences). The larger system, however, makes it slower to track and optimize.

By directly optimizing intensity residuals, photometric bundle adjustment works more closely with image statistics – operating at origin of the noise. By jointly optimizing camera poses and inverse depths, it is able to account for image noise across the entire sequence. We therefore track a sparser set of points in the first two stages of the pipeline to obtain a rapid initialization, then we refine our estimate with a photometric objective.

We deploy a Gauss-Newton optimization similar to Engel et al. [4], but with only one reference frame and the only camera intrinsic we solve for is focal length, f. By solving for a compositional update to the rigid pose, we remove the small angle approximation made in the bundle adjustment stage.

Given rectified images, we wish to minimize the reprojected intensity of a point in the reference to other frames in the sequence. We define this energy as

$$E_{\text{PBA}} = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{\mathbf{x}\in\mathcal{P}_j} \|r_{ij}\|_\gamma, \tag{14}$$

$$r_{ij} = I_i(f\mathcal{W}(\tfrac{1}{f}\mathbf{x}_j; \Delta\boldsymbol{\omega}_i, \Delta\mathbf{t}_i, \Delta d_j)) - I_0(\mathbf{x}_j), \tag{15}$$

where I_0 and I_i are the reference and target images respectively, $\mathcal{P}_j$ is the set of patch pixels used for point j, $\|\cdot\|_\gamma$ is the Huber norm, and project pixel $\mathbf{x}$ into frame i by

$$\mathcal{W}(\mathbf{x}_j; \Delta\boldsymbol{\omega}_i, \Delta\mathbf{t}_i, \Delta d_j) \tag{16}$$

$$= \langle \mathcal{R}(\Delta\boldsymbol{\omega}_i)\mathbf{R}_i\tilde{\mathbf{x}}_j + (\Delta d_j + d_j)(\Delta\mathbf{t}_i + \mathbf{t}_i)\rangle\,. \tag{17}$$

where $\mathcal{R}$ linearizes the rotation update as defined by (4), and $\mathbf{R}_i$ is the latest rotation matrix for frame i is updated at each iteration by

$$\mathbf{R}_i := \mathcal{R}(\Delta\boldsymbol{\omega}_i)\mathbf{R}_i. \tag{18}$$

We iteratively solve for $\Delta\boldsymbol{\omega}_i, \Delta\mathbf{t}_i, \Delta d_j$ until convergence.

3.4. Dense Depth Reconstruction

For dense depth reconstructions we base our method on the PatchMatch algorithm [2]. It differs in that our cost function assesses the variance over all frames instead of just a stereo pair. One could deploy it with the same variance cost used by Ha et al. [7]. However, given a depth hypothesis we propose a more robust way of estimating the mean (before estimating the total variance) by giving a higher weight to nearby frames.

The goal of calculating the mean intensity is to estimate the *true* intensity of the reference pixel. Often the raw intensity of the pixel itself is used, but image noise can affect accuracy of this value. Spatial smoothing can reduce noise, but reduces the ability to recover high frequency detail. Ha et al. [7] choose to smooth the intensities temporally

[2]SciPy Documentation: https://goo.gl/PwX2mT

by estimating the mean of the projected intensities across all frames, but this is only valid for a correct depth estimate.

It is well understood in multi-view geometry that a narrow baseline leads to lower certainty in the estimated depth since it is more sensitive to perturbations in the projected point positions. By the same, yet inverted, logic we argue that when calculating the mean intensity, it is better to weight sampled pixels from closer frames, since the projected point position is less sensitive to changes in the depth.

$$r_j(d) = \frac{1}{M} \sum_{i=1}^{M} \sum_{\mathbf{x} \in \mathcal{P}_j} |I_i(\mathbf{x}'_i) - \mu(d; \mathbf{x})|, \quad (19)$$

$$\mu(d; \mathbf{x}) = \sum_{i=1}^{M} w_i I_i(\mathbf{x}'_i), \quad (20)$$

where $\mathbf{x}'_i$ denotes the projected point location in frame i according to the known camera pose and current inverse depth hypothesis; w_i is weighting given to each frame according the inverse distance of its center to the reference frame

$$w_i = \frac{\|\mathbf{t}_i\|^{-1}}{\sum_{i=1}^{M} \|\mathbf{t}_i\|^{-1}}, \quad (21)$$

such that nearby frames are weighted more highly.

4. Experiments

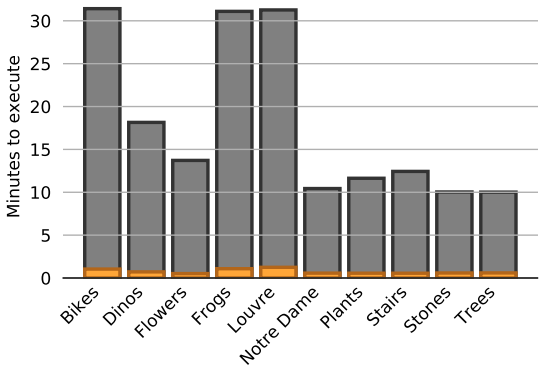
We compare our method with the most recent small baseline method Ha etal. [7] and OpenMVS. We extensively evaluate our method on: synthetic datasets consisting of a large number of different baselines, frame counts, and realistic motion; handheld video clips from [7].

While the striking results for small motion clips presented in [7] rely on a refinement step, we argue that the same process can be applied to our output. Our focus in this paper is to recover a more accurate raw result. As such, we compare with the filtered (high variance points removed, but non-refined) output from their algorithm.

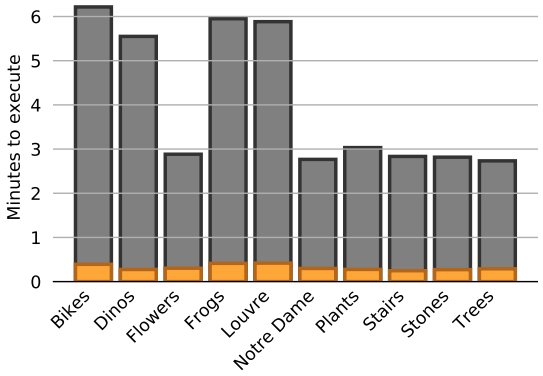
OpenMVS, together with structure-from-motion pipeline OpenMVG, is a widely-used open-source library for structure-from-motion and dense 3D reconstruction. It utilizes classical feature-matching methods to get camera poses and PatchMatch [2] for dense 3D reconstruction. We include OpenMVS + OpenMVG results wherever possible, however it has not been designed for small baselines and fails for most sequences

4.1. Hardware

All experiments were run on a machine with a dual core 2.5GHz Intel™ Core i5 CPU and 8Gb of RAM. Our handheld video is recorded using an iPhone6 at 240fps. We turned off Optical Image Stabilization (OIS) and automatic focus to remove their interference with the true camera motion.



(a) Full Resolution



(b) Half Resolution

Figure 3: Time taken to process the outdoor small motion dataset. Our method (*orange*) outperforms that of Ha et al. [7] (*grey*) by at least an order of magnitude in almost all cases.

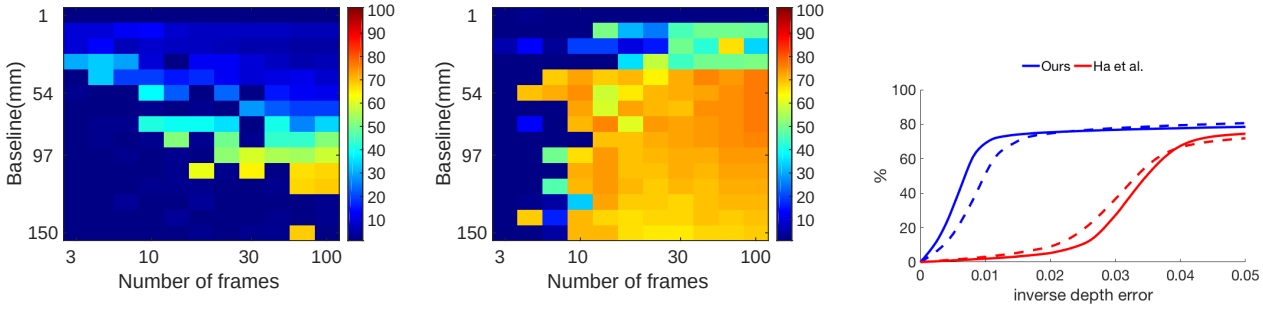
4.2. Parameters

Unless otherwise specified, we initialize our pipeline with 600 of the best quality Harris corners [9], default parameters for compared methods were used. To achieve comparable depth map quality, we adopt 128 plane sweeping level setting for method in [7] (default is 64 level). In contrast, our method estimates dense depths in the continuous domain. Our experiments demonstrate that it achieves similar or better reconstruction quality with significantly less computation.

4.3. Timings

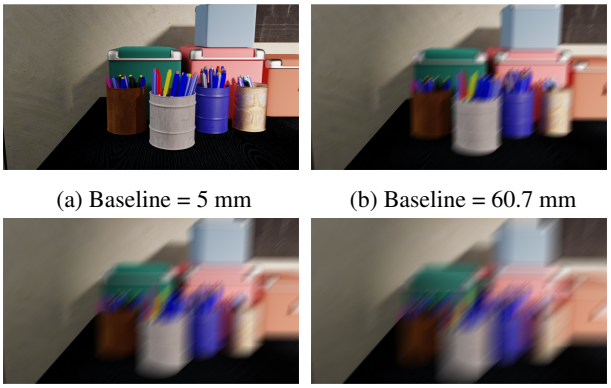
To compare the timing and quality of our method for small motion clips, we use the dataset in [7]. It contains a variety of small motion videos each with 30 frames captured with mixtures of the settings: (30FPS, 240FPS), (1920×1080, 1280×720).

While being at least an order of magnitude faster than



(a) Ha et al. [7] (b) Ours (c) Cumulative error curve

Figure 4: Heat map visualization for depth map accuracy. The color of heat map visualizes percentage of pixels with inverse depth error less than $0.015m^{-1}$. Example cumulative error curves for two of the sequences show the percentage of pixels with inverse depth error less than the corresponding threshold; the *solid* line is for the sequence with 75mm baseline and 70 frames, the *dashed* line is for the sequences with 75mm baseline and 10 frames.



(a) Baseline = 5 mm (b) Baseline = 60.7 mm

(c) Baseline = 130.4 mm (d) Baseline = 200 mm

Figure 5: Average image of different baseline sequences.

Ha et al. [7] (Fig. 3) our method is able to produce qualitatively higher quality depth maps (Fig. 9). We show the time comparison of each stage in supplementary video.

4.4. Baseline and Sequence Length Spectrum

In this experiment, we compare the quality of the depth map using a spectrum of different baselines and number of frames. In practical cases, there is no way to restrict users when taking videos, so a good system should achieve stable performance across the spectrum.

With Blender™ (ver. 2.78) we render a realistic image sequences and generated ground truth depth maps and camera poses. The resolution of our rendered image is 1280×720. The sequences are synthesized over a spectrum of baselines and number of frames. The motions are linear with total distances ranging from 1 to 150 millimeters. Figure 5 visualizes baselines by average images of the sequences used in the experiment shown in Figure 4. Pixels without valid ground truth depth value are not counted.

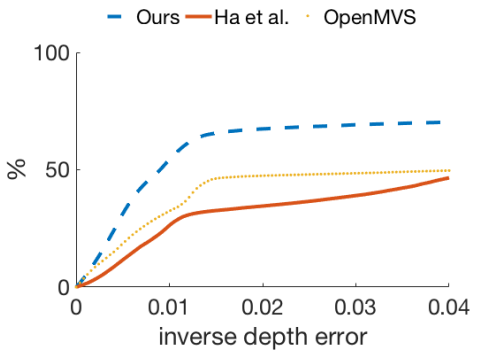
Inverse Depth Error We compare errors in the inverse depth domain because the reprojected pixel displacement remains the same for all depths for the same inverse depth error, and so better aligns with the source of the errors (image patch alignment).

We visualize our results in Figure 4. Considering the real distance in common use case, we visualize the heat map to show the percentage of depth map pixels within an error of $0.015m^{-1}$, which corresponds to less than 5cm at the mean depth in our scene (1.79m), where failure cases are represented by dark blue. Errors larger than 5cm are considered invalid. To determine the relative depth scale to ground truth, we compute initial relative scale by minimizing least squares error between camera pose and ground truth camera pose extracted from Blender. Then we perform exhaustive search for the scale which leads to maximum number of pixels with inverse depth error less than $0.015m^{-1}$

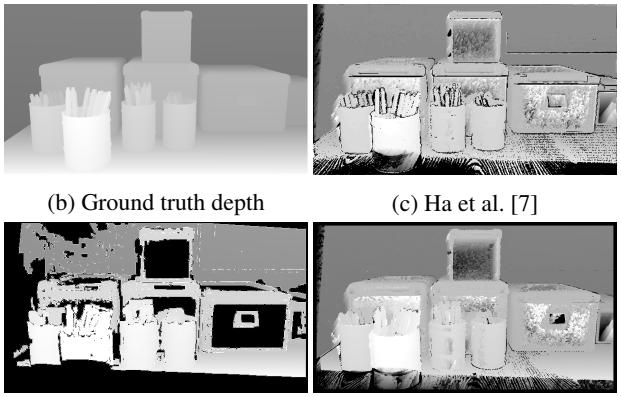
The result show an improved tolerance to baselines and number of frames when using our method over [7]. [7] breaks down when the baseline is too small (failed to estimate from image difference) and drops performance when baseline is larger (failed to find correct point correspondences). We also note a trend that using more frames within a fixed baseline helps increase the accuracy of the depth map. For a more detailed view of the depth map errors, we visualize the cumulative error curve for two cases in Figure 4. OpenMVS + OpenMVG is not comparable to other methods in this test. It can only cope with larger baseline and fails in most cases.

4.5. Synthetic with Hand Motion

We test the depth reconstruction methods in a synthetically generated scene with real handheld motions, allowing us to compare with ground truth pose and structure. We record motions by using a checker board and the Matlab



(a) Cumulative error for synthetic hand sequence



(b) Ground truth depth (c) Ha et al. [7]

(d) OpenMVS (e) Ours

Figure 6: Depth maps and cumulative error with the ground truth for the synthetic dataset with real hand-held motion.

Camera Calibrator toolbox to track camera poses. The camera was moved in a circular motion roughly parallel to the reference image plane.

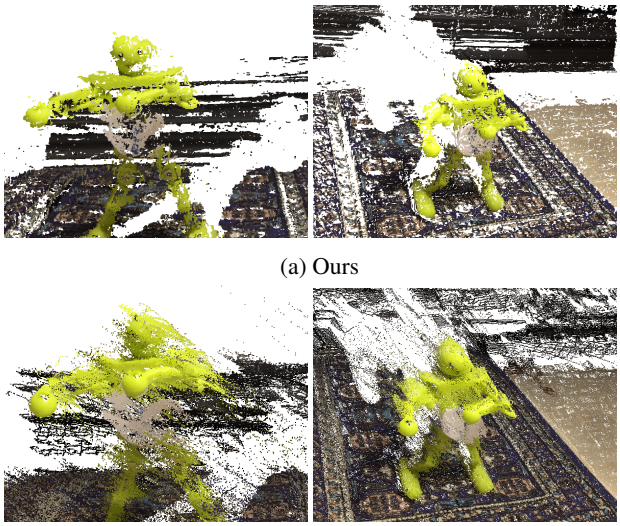
The results show that our depth map has better accuracy than other methods (Fig. 6). In this case, OpenMVS is able to reconstruct a 3D point cloud, but the number of valid pixels is lower than our result.

4.6. Object-centric Videos

We capture two object-centric videos using an iPhone 6 at 240 FPS: one of a toy robot, the other of a dragon model with a thin tail. We visualize the reconstructed depth maps from different views in order to give a better insight to the difference between the compared methods. In the toy sequence (Fig. 7) we observe significantly fewer spurious points for our method while retaining a similar density. In the dragon sequence (Fig. 8) Ha et al. [7] recover more of the tail but still at the cost of spurious points.

4.7. Small-baseline video dataset

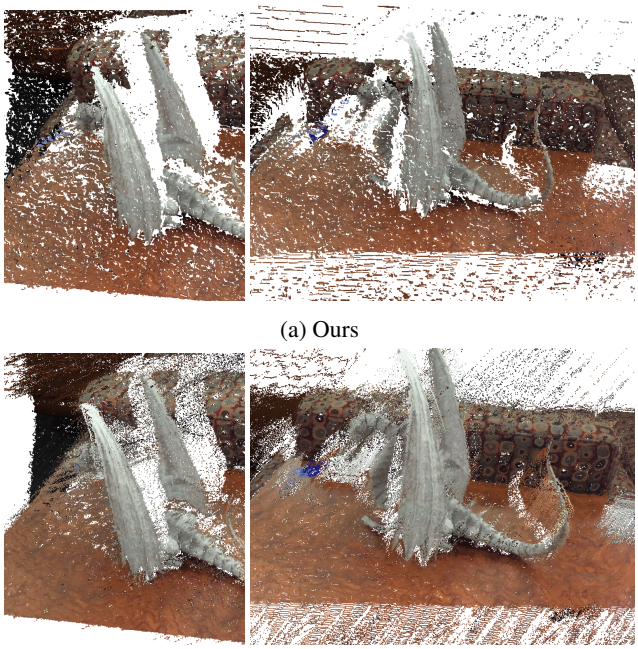
We also evaluate our result using small-baseline dataset provided by [7]. The resulting depth map is visualized in



(a) Ours

(b) Ha et al. [7]

Figure 7: Point cloud visualization comparing reconstructed depth maps of the toy sequence. Ha et al. [7] outputs many more spurious points, particularly around occlusions and in homogeneous areas.



(a) Ours

(b) Ha et al. [7]

Figure 8: Point cloud visualization comparing reconstructed depth maps of the dragon sequence. Ha et al. [7] outputs many more spurious points, particularly around occlusions and in homogeneous areas.

Figure 9. We can observe that our depth map has comparable quality to [7], and outperforms in occlusion boundary

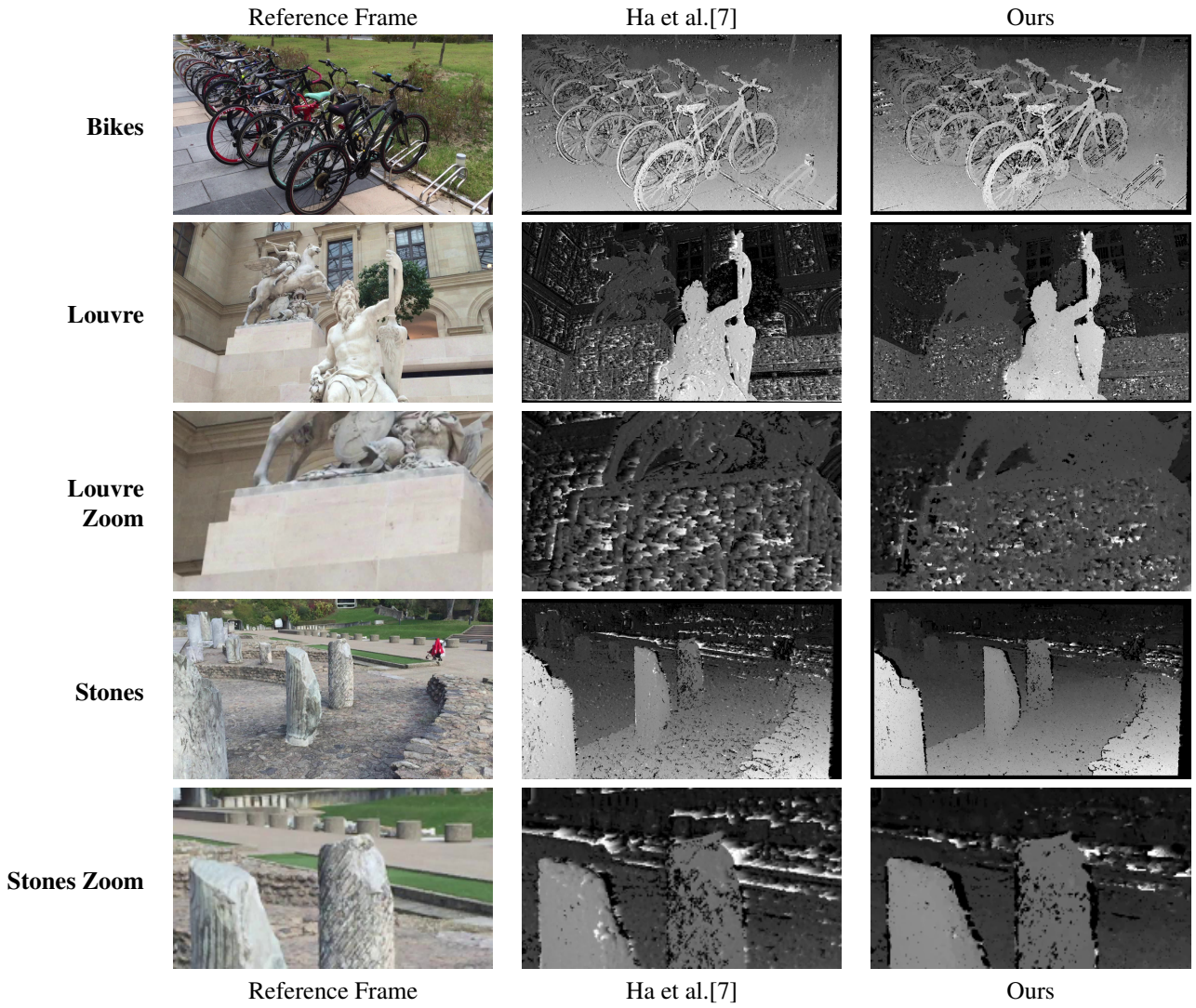


Figure 9: Depth map comparison with dataset from [7]. We zoom in on areas of interest to show the difference in quality. Our method is better able to reject occlusions and produces smoother depth maps in homogeneous areas.

and homogeneous area. More point cloud visualization are provided in supplementary video.

5. Conclusion

We proposed a pipeline for constructing depth map from small video clips. Compared with previous methods, our method can cope with a wider range of baselines and number of frames, and significantly more computationally efficient than Ha et al. [7]. Future work will use the presented pipeline for more complete object-centric tracking and reconstruction, including keyframe management and further acceleration using inverse compositional techniques [8].

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[3] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo-stereo matching with slanted support windows. In *BMVC*, volume 11, pages 1–11, 2011.

[4] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *arXiv preprint arXiv:1607.02565*, 2016.

[5] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[6] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[7] H. Ha, S. Im, J. Park, H.-G. Jeon, and I. So Kweon. High-quality depth from uncalibrated small motion clip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5413–5421, 2016.

[8] C. Ham, S. Lucey, and S. Singh. Proxy Templates for Inverse Compositional Photometric Bundle Adjustment. *ArXiv e-prints*, Apr. 2017.

[9] C. Harris and M. Stephens. A combined corner and edge detector. Citeseer, 1988.

[10] N. Joshi and C. L. Zitnick. Micro-baseline stereo. Technical report, Technical Report MSR-TR-2014-73, 2014.

[11] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[12] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[13] T. Tamaki, T. Yamamura, and N. Ohnishi. Unified approach to image distortion. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 584–587. IEEE, 2002.

[14] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3d reconstruction on mobile phones. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 65–72, 2013.

[15] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. Dense reconstruction on-the-fly. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1450–1457. IEEE, 2012.

[16] Q. Yang. A non-local cost aggregation method for stereo matching. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1402–1409. IEEE, 2012.

[17] F. Yu and D. Gallup. 3d reconstruction from accidental motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3986–3993, 2014.