Progressive 3D Model Acquisition with a Commodity Hand-held Camera

Zhuoliang Kang and Gérard Medioni
University of Southern California
Los Angeles, CA 90089
{zkang|medioni}@usc.edu

Abstract

We present a system for progressive 3D model acquisition with a commodity hand-held camera. The pipeline starts with a real-time scanning stage accomplished using a sparse point-based tracker for camera pose estimation and a dense patch-based tracker for dense reconstruction. While the user scans the target object, a model composed of local planar patches is continuously updated and displayed for visual feedback. This live feedback loop allows the user to choose new viewpoints based on the state of the current reconstruction, and determine if the model is completely covered in desired details. After live scanning is completed, our system refines the reconstructed patches into denser and more accurate patches through an offline model refinement procedure. We demonstrate the ability of our system on various real datasets.

1. Introduction

Acquisition of 3D models for real-world objects is essential for applications in various fields. Active scanners, such as structured-light scanner and laser scanner, provide high quality range maps which can be merged into a single 3D model. The reconstructed models are highly accurate, but the set up is cumbersome. Image-based approaches attempt to extract the 3D model from a set of images or video sequences based on photometric information from different views. Recent advances in computer vision enable accurate 3D model acquisition with reconstruction error on the order of millimeters without particular hardware, except a digital camera.

The standard off-line 3D modeling method starts with a data collection stage where images covering different views of the target are collected. The camera poses of the collected images are estimated using Structure-from-Motion (SfM) framework. Multi-view stereo algorithms are then used to generate the 3D model. Off-line methods are able to generate accurate 3D model through incorporating information from all collected data. If the user is not satisfied with the reconstructed model, e.g., object is not fully covered, the user needs to collect additional images and repeat the reconstruction pipeline. Thus, the acquisition of a visually pleasing 3D model can be time-consuming with several iterations of data collection and reconstruction loop. Real-time approaches have been proposed to overcome this problem. During scanning, the reconstructed model is continuously updated and displayed as feedback to guide the user for the next viewpoint. However, the accuracy of the reconstructed model is not guaranteed due to the lack of global information.

As 3D printing technology becoming popular and affordable, we are interested in providing a simple-to-use 3D model acquisition solution incorporating the good elements from both off-line methods and real-time methods. In this paper, we propose a system with a commodity hand-held camera combining several state-of-the-art methods in computer vision and graphics, which offers following specialties:

- **Real-time scanning:** data is collected through a live scanning stage, where immediate feedback of a continuously-updated model allows the user to determine if the region of interest is reconstructed in desired detail and choose new viewpoints based on the state of the current model.
- **Off-line model refinement:** after live scanning is completed, the obtained reconstruction is refined into a more accurate model through an off-line model refinement procedure utilizing information from all collected data.
- **Detailed reconstruction with a commodity device:** the resolution of the reconstructed structural details is controlled by the image resolution. It enables our system to capture highly-detailed structures with a commodity device through scanning the object in close-up views.
- **No need for inertial information:** no inertial information from GPS or inertial measurement unit (IMU)

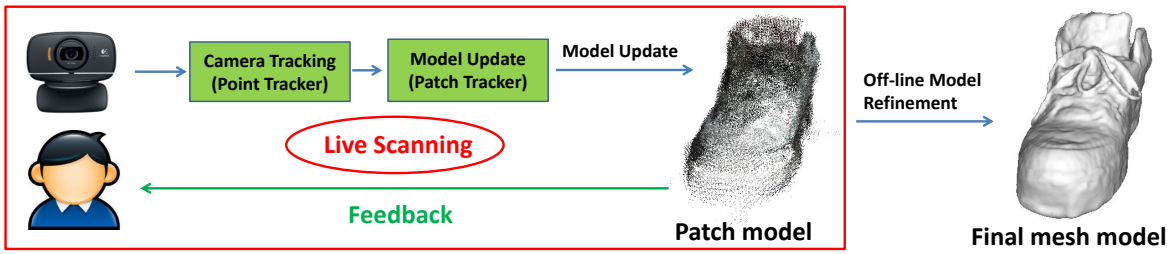

Figure 1. Overall pipeline of the proposed system.

is needed.

The rest of the paper is organized as follows. In section 2, we list the related works. In section 3, we introduce the system overview. We explain the point tracker in section 4 and the patch tracker in section 5. In section 6, we describe the model refinement procedure. In section 7, we present experimental results. In section 8, we conclude our paper with future work.

2. Related work

As summarized in [4], multi-view stereo methods can be categorized into four classes based on the underlying model representation. Volume-based methods split the 3D space into a grid of discrete voxels and estimate a binary or probability occupancy status for each voxel [17]. Polygon mesh-based methods start with an initial polygon model and deform it to fit the information from different views [5]. Methods based on the fusion of depth maps estimate a depth map for each individual image. The depth maps are then fused to build a single 3D model [8][21]. Patch-based methods attempt to obtain a 3D model composed of local planar patches densely covering the object surface, which makes it flexible for objects with complicated shapes [4].

A variety of real-time methods based on active scanners have been proposed for 3D model acquisition. Rusinkiewicz et al. [15] proposed a system estimating range scans based on structured-light projected on the object surface. The range scans are registered using a fast Iterative Closest Point (ICP) method [16] and merged through VRIP [3]. KinectFusion [11] represents the 3D space with a discrete 3D volume and update it in real-time with depth maps obtained from a hand-held depth sensor. There are also image-based real-time 3D modeling system have been proposed to extract the 3D model from images in real-time. ProFORMA [13] updates a triangle mesh model while the user rotating the object in front of a stationary camera. The model is produced through delaunay tetrahedralisation of SfM points followed by a triangle carving step, which makes it unsuitable for objects with non-planar structural details. In [14], Pollefeys et al. proposed a real-time dense reconstruction system for urban scenes using a multi-camera rig. Depth maps are estimated using plane sweep algorithm and integrated to build the urban model through visibility-based depth map fusion [8].

Recently, several works have been proposed for real-time dense reconstruction with a single hand-held camera. In [10], accurate camera poses are obtained using PTAM [7]. Constrained scene flow updates the depth map for each camera bundle incorporating photometric information from different views. In [9], the depth map of each key frame is estimated using a variational method with discrete-continues optimization. Robust camera tracking is achieved through dense image alignment with the synthetic views rendered from the continuously-updated dense model. They have shown great performance for scene reconstruction in a workspace scenario. Their capabilities to acquire a complete 3D model have not been evaluated. A more recent work [18] has demonstrated the potential to do dense reconstruction on a mobile platform with the aid of on-device inertial sensors. In contrast, our system aims for 3D model acquisition using a commodity camera without inertial information from other sensors.

3. System Overview

The overall pipeline of our system is illustrated in Figure 1. It starts with a live scanning stage where the user scans the target object obtaining input video with a commodity hand-held camera. This stage is accomplished using a point-based tracker for camera pose estimation and a patch-based tracker for dense reconstruction. The point-based tracker maintains tracks of SIFT points and updates a model composed of a sparse set of 3D SIFT points for camera tracking. At the same time, the patched-based tracker maintains tracks of image patches and updates a patch model composed of local planar patches densely covering the surface. During scanning, reconstructed patches are continuously updated and displayed to the user as immediate feedback. After live scanning is completed, denser and more accurate patches are reconstructed through an off-line model refinement procedure. Erroneous patches are also filtered out by checking the visibility information in this step. The detailed mesh model is generated from the refined dense patches through Poisson surface reconstruc-

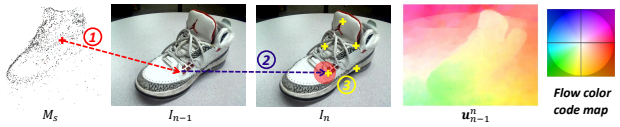

Figure 2. Illustration of the optical flow guided feature matching between a new frame I_n and the sparse model M_s

tion [6].

4. Sparse Point-based Tracker

Accurate and efficient camera pose estimation at each input frame is essential for live 3D model acquisition. This section describes the online point-based tracker used to estimate the 6DOF camera pose (position and viewing direction).

4.1. Camera Pose Estimation

For camera pose estimation, our system maintains and continuously updates a model M_s composed of a sparse set of 3D SIFT points. When a new frame I_n arrives, SIFT points on I_n are detected and matched with the 3D points stored in M_s. The camera pose of I_n is then estimated using Perspective-n-Point (PnP) method followed by an iterative method minimizing the re-projection error based on Levenberg-Marquardt optimization. Using the 2D-3D Image-Model feature matches, the camera pose for each new frame I_n is aligned with the maintained 3D model M_s, which effectively reduces the pose drift during scanning. The update of M_s is described in next section, and the feature matching between I_n and M_s is explained in section 4.3.

4.2. Point Model Update

As each new frame I_n arrives, SIFT points between successive frames are matched and chained into point tracks across multiple frames. The matching of features between successive frames is described in next section. When the baseline of a track is large enough,we generate a 3D SIFT point through triangulation and insert it into M_s. In practice, we triangulate a track into 3D point when the angle between the viewing rays cast from the initial detection and the detection on the current frame I_n is larger than 10°. To avoid inserting duplicate points in M_s associated with the same SIFT feature, a point track is deleted if its detection on the current frame is matched with a 3D SIFT point already stored in M_s. The SIFT points on I_n that are associated with neither existing point track nor points in M_s are added as initial detections to start new a track.

It is not necessary to update M_s and add new tracks at each frame due to the small variance between successive frames in the input video sequence. In practice, we attempt to update M_s and add new tracks at key frames only. A key frame is added when the time exceeds more than ten frames and the camera moves more than a minimum distance since last key frame. The camera poses of the first two key frames are estimated using 5-point method [12] to initialize M_s.

4.3. Optical Flow guided Feature Matching

Feature matching is one of the most time-consuming step of Structure-from-Motion (SfM) framework. To make use of the small variance in video sequences, we compute dense optical flow between successive frames to guide the matching of SIFT feature points. Feature matching between a new frame I_k and the sparse model M_s is accomplished through 3 steps as illustrated in Figure 2:

1. **Back-projection**: for each 3D SIFT point X stored in M_s, we back-project it onto the previosu frame I_{n-1} and obtain its projection $\mathbf{x}_{n-1}$. Visibility is checked based on the point normal at X, which is estimated as described in section 5.2.

2. **Dense optical flow**: we compute the dense optical flow between the previous frame I_{n-1} and the current frame I_n using Total-Variation L_1 (TV-L_1) optical flow [22], which minimizes an L_1 norm data term and a total-variation regularization term:

$$\mathbf{E}_\mathbf{u} = \int_\Omega (\lambda \left| I_{n-1}(\mathbf{x}) - I_n(\mathbf{x} + \mathbf{u}^n_{n-1}(\mathbf{x})) \right| + \left| \nabla \mathbf{u}^n_{n-1} \right|) d\mathbf{x} \tag{1}$$

   where $\mathbf{u}^n_{n-1}(\mathbf{x})$ represents the displacement from I_{n-1} to I_n at point $\mathbf{x}$. Then, the projection of X on I_n can be evaluated by adding the displacement to its projection on I_{n-1}:

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{u}^n_{n-1}(\mathbf{x}_{n-1}) \tag{2}$$

3. **Local search**: we now search in a small region around $\mathbf{x}_n$ on I_n to find the matchig SIFT point for X. In all experiments, a match is found if the dot product of their normalized feature descriptors is larger than 0.8. The search region is defined as a circle with radius of 3 pixels centered around $\mathbf{x}_n$. If multiple SIFT points on I_n are found to be matched with X, we drop them all to avoid false matches with ambiguous features, e.g., in regions with homogeneous texture. Feature matching between successive frames I_{n-1} and I_n is achieved in the same way without the back-projection step.

4.4. Bundle Adjustment

Whenever a new key frame inserted, we run a full Bundle Adjustment involving obtained key frames by minimizing the total re-projection error:

$$\mathbf{E}_{\theta_i, X_j} = \sum_i \sum_j d(\mathbf{P}(\theta_i, X_j), \mathbf{x}_{ij})^2 \tag{3}$$

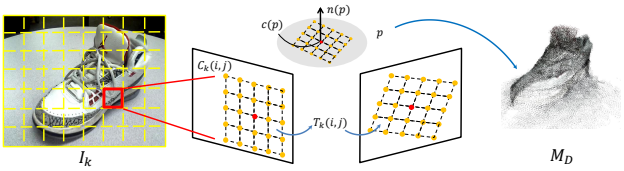

Figure 3. Each key frame I_k is divided into $\mu \times \mu$ pixels image cells associated with patch tracks to continuously update the patch model M_D.

where θ_i represents the camera pose of i-th key frame, X_j is the j-th 3D SIFT point stored in M_s, $d(\mathbf{P}(\theta_i, X_j), \mathbf{x}_{ij})$ denotes the geometric error between the predicated projection of X_j on i-th key frame and its actual projection $\mathbf{x}_{ij}$. Running time is an essential concern for a live scanning system. In our implementation, dense optical flow guided feature matching provides accurate initial pose estimation. We are able to achieve sub-pixel accuracy after a few iterations of bundle adjustment. We also use the recent progress in parallelized bundle adjustment [20] to further speed up the process. More details of the implementation and running time analysis are described in section 7.1.

5. Dense Patch-based Tracker

Point-based tracker provides the camera pose and updates the 3D model M_S composed of sparse SIFT feature points. To fully capture the geometry of the target object, our system generates another 3D model M_D composed of local planar patches densely covering the visible surface. Inspired by [4], we set each patch p representing a local planar surface with center position $c(p)$ and unit normal $n(p)$. During scanning, M_D is continuously-updated and displayed as immediate feedback for the user. We describe the patch tracks in next section, and the model update in section 5.2.

5.1. Patch Tracks

As shown in Figure 3, our system maintains a set of patch tracks and updates them during live scanning. Whenever a key frame I_k is inserted as described in section 4.2, it is divided into a regular grid of $\mu \times \mu$ pixels image cells (μ is set as 7 in all experiments). Each image cell $C_k(i,j)$ is then associated with a patch track $T_k(i,j)$ comprising point trajectories for all pixels in $C_k(i,j)$. When a new frame I_n arrives, point trajectories in $T_k(i,j)$ are updated by adding the point displacements obtained from the dense optical flow between the last frame I_{n-1} and current frame I_n. Sub-pixel accuracy is achieved through bilinear interpolation. The point trajectories are refined using their projections on the corresponding epipolar lines on I_n.

Figure 4. Illustration of scanning for structural details. (a) a sample input frame captured in distane view. (b) the patches reconstructed from distance views is sparse. (c) a sample input frame captured in close-up view for the head region. (d) the reconstructed patches from close-up views are denser and in smaller scales.

5.2. Patch Model Update

Whenever a key frame arrives, our system updates the patch tracks as described above and attempts to add new patches into the patch model M_D. If the angle between the viewing rays casted from the initial projection and the projection on the current frame is larger than 10° for all point trajectories in patch track $T_k(i,j)$, we generate a candidate patch p from $T_k(i,j)$ by triangulating the point trajectories in $T_k(i,j)$ into a set of 3D points. The depth value of each pixel in $C_k(i,j)$ is saved for later refinement as described in section 6. The patch center $c(p)$ is set as the 3D point associated with the center pixel of $C_k(i,j)$. To obtain the patch normal, we eigen-decompose the covariance matrix of the 3D points into eigenvectors μ_i and the eigenvalues λ_i $(i = 1, 2, 3)$. The patch normal $n(p)$ is set as μ_3, which corresponds to the minimal eigenvalue λ_3. Patch p is considered as a valid patch if $\lambda_1/\lambda_2 < 2$ and $\lambda_2/\lambda_3 > 10$. The non-planar patches and the patches that are stretched into a single direction are filtered out as invalid patches.

5.3. Scanning for Structural Details

The density and the scale of the reconstructed patches are controlled by the resolution of the source image. Patches that are generated from images captured in close-up views have a smaller scale. Reconstructed patches are also denser in these regions as each patch is associated with an image cell with fixed size. As shown in Figure 4, the reconstructed patches generated from the input frames in distant views are sparser with structural details lost. On the other hand, frames captured in close-up views generate denser patches capturing structural details in smaller scales. The updated patch model M_D is displayed to the user in the course of live scanning process. This immediate feedback of the current reconstruction allows the user to choose next movement and achieve reconstruction in desired resolution by browsing the regions of interests in close-up views.

6. Model Refinement

The reconstructed patches from live scanning are not quite accurate due to the error in the patch tracks which are computed based on the accumulated dense optical flow.

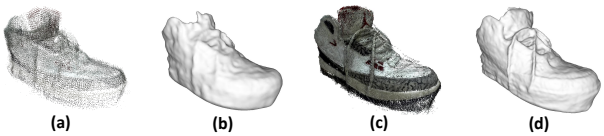

Figure 5. Illustration of depth map refinement. (a) the reconstructed coarse patches after live scanning. (b) the mesh model generated from the coarse patches. (c) denser and accurate patches after depth map refinement. (d) the mesh model generated from the dense patches. Mesh models are generated through Poisson surface reconstruction under the same parameters.

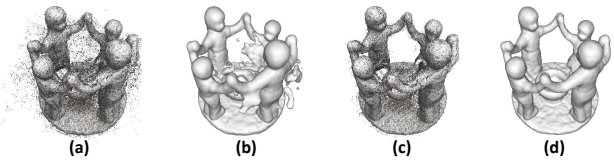

Figure 6. Illustration of patch filtering. (a) patch model before patch filtering which contains erroneous patches. (b) the mesh model generated from (a) which includes several artifacts. (c) patch model after filtering out invalid patches. (d) the mesh model generated from (c) which is clean and accurate. Mesh models are generated through Poisson surface reconstruction under the same parameters.

Also, structural details in small scales are lost as we use image cells with a large size (7×7 pixels) for robust patch estimation. There are also erroneous patches in the model which are mainly associated with the texture-less regions where the optical flow displacements are not reliable. To overcome these problems, we generate a patch model that is denser and more accurate through a depth map refinement procedure as described in the next section. The erroneous patches are filtered out based on the visibility information as described in section 6.2. In the end, a detailed mesh model is generated from the refined patches through Poisson surface reconstruction.

6.1. Depth Map Refinement

As shown in Figure 5 (a)(b), the patch model obtained from live scanning is composed of sparse patches in large scales, which builds a coarse mesh mode that is over-smooth with structural details lost. For each key frame I_k, an initial inverse depth map $\mathbf{h}_k$ is obtained based on the depth information generated from the dense patch tracker. We refine $\mathbf{h}_k$ by incorporating photometric information of the neighboring key frames of I_k through a variational multi-view stereo method. The neighboring key frames $I_i \in \mathcal{N}(k)$ (we use 10 neighboring key frames in all experiments) are selected as the key frames obtained around I_k in time order. In particular, we estimate an inverse depth offset map $\mathbf{dh}_k$ representing the deformation of the initial model. $\mathbf{dh}_k$ is computed by minimizing an energy function including a non-convex photometric error term and a convex regularization term:

$$\mathbf{E_{dh}} = \int_\Omega \lambda C_k(\mathbf{x}, \mathbf{h}_k(\mathbf{x}) + \mathbf{dh}_k(\mathbf{x}))d\mathbf{x} + \int_\Omega \|\nabla \mathbf{dh}_k(\mathbf{x})\|_\epsilon d\mathbf{x} \tag{4}$$

where $\|\nabla \mathbf{dh}(\mathbf{x})\|_\epsilon$ is a Huber norm regularization term used to smooth the offset map. $C_k(\mathbf{x}, h)$ measures the photometric error across its neighboring views incorporating occlusions:

$$C_k(\mathbf{x}, h) = \sum_{i \in \mathcal{N}(k)} \rho_k^i(\mathbf{x}) \left| I_k(\mathbf{x}) - I_i(\pi_i(\pi_k^{-1}(\mathbf{x}, h))) \right| \tag{5}$$

where $\pi_k^{-1}(\mathbf{x}, h)$ is the operator to compute the position of the 3D point projected from pixel $\mathbf{x}$ on I_k when assigned to inverse depth value h, and $\pi_i(\pi_k^{-1}(\mathbf{x}, h))$ is the operator to compute the pixel location on I_i back-projected from this 3D point. $\rho_k^i(\mathbf{x}) \in \{0, 1\}$ represents the visibility of pixel $\mathbf{x} \in I_k$ observing from I_i which is computed through depth test based on the initial inverse depth maps. By minimizing the energy function in Equation 4, we deform the initial patches to ensure photometric consistency across neighboring frames and force the deformation to be smooth. Details of the optimization can be found in [9][2].

We ray-cast the refined inverse depth maps to generate denser and more accurate patches. The patch center and its normal are computed int the same way as described in section 5.2. We associate each patch with an image cell in a smaller size (3×3 pixels in our implementation) to generate denser patches capturing structural details in small scales. As shown in Figure 5 (c)(d), the refined patches are denser and more accurate which lead to a detailed mesh model with structural details well captured.

6.2. Patch Filtering

We incorporate two types of visibility information to filter out invalid patches. Inspired by [18], we filter out erroneous patches that are not consistent among neighboring frames within a small baseline. Particularly, we warp the reconstructed patches onto $N_d = 4$ neighboring frames to check the consistency based on the refined depth maps. If the depth difference is larger than a tolerance value on more than $N_d = 2$ neighboring frames, the patch is filtered out as an erroneous patch. We also filter out erroneous patches that violate occlusion consistency across large baselines: we warp each patch onto non-neighboring frames to check if it occludes patches in the target frame. In our implementation, patches that are violating occlusion consistency for more than $N_v = 3$ frames are also filtered out as erroneous patches. Figure 6 illustrates an example of the reconstructed patches before and after patch filtering.

Dataset				Running-time statistics		
Name	**Size**	**# Frames**	**# Key Frames**	**Scanning**	**Refinement**	**Total**
stone	640 × 480	770	69	1m 16s	2m 44s	4m 00s
shoe	640 × 480	690	61	1m 08s	2m 30s	3m 38s
bach	640 × 480	680	60	1m 06s	2m 28s	3m 34s
statue	1280 × 720	630	52	2m 09s	5m 10s	7m 19s
cleopatra	1280 × 720	588	47	2m 04s	4m 44s	6m 48s
gargoyl	1280 × 720	687	58	2m 21s	5m 41s	8m 05s
birdhouse	1280 × 720	270	18	1m 08s	2m 13s	3m 21s

Table 1. Dataset details and running-time analysis. # **Frame**: number of total frames. # **Key Frame**: number of key frames. **Scanning**: total time cost for live scanning. **Refinement**: total time cost for model refinement including depth map refinement and patch filtering. **Total**: total time cost including both scanning and model refinement.

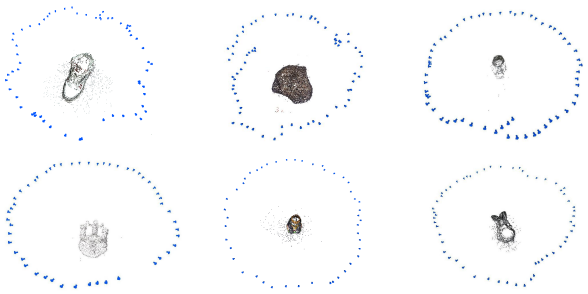

Figure 7. Illustration of the estimated camera motion as well as the reconstructed sparse model composed of SIFT feature points. Top row from left to right: results on *shoe*, *stone* and *bach*. Bottom row from left to right: results on *statue*, *cleopatra* and *gargoyl*

7. Experiments

7.1. Implementation and Datasets

We implement our system on a machine with Intel Xeon 3.6GHz quad-core processor and a GTX 590 graphics card. For real-time performance, we use SiftGPU [1] for feature detection, Multicore Bundle Adjustment for Bundle Adjustment [2] and FlowLib [3] for dense optical flow computation which are all parallelized on GPU. The depth map refinement algorithm is also parallelized on GPU with CUDA implementation.

We test our system on three different datasets of real-world objects. *stone*, *shoes* and *bazh* are collected in a lab environment without particular setup for lighting and background. For data collection, we use Logitech C520 which is a commodity webcam working under standard resolution of 640 × 480. Besides the experiments tested in the lab, we also evaluate our system on videos from two publicly available datasets. *statue*, *cleopatra* and *gargoyl* are three videos of objects placed on a table captured in a lab environment from **CLAM** [1] dataset. *birdhouse* from **Image Seq.** dataset [19] is captured in outdoors. Both datasets are collected using a commodity hand-held digital camera. Details of the used datasets are listed in Table 6.2, including the total number of frames, number of key frames, and image size.

[1] http://cs.unc.edu/∼ccwu/siftgpu/
[2] http://grail.cs.washington.edu/projects/mcba/
[3] http://gpu4vision.icg.tugraz.at/

7.2. Running-time analysis

Running time is important for the live scanning stage. Though parallelized using GPU, SIFT feature detection and the computation of dense optical flow are the most time-consuming parts in our system. In practice, we process them in parallel using two separate threads since they are independent of each other for each new frame. A few iterations of Bundle Adjustment is sufficient to provide accurate camera pose refinement. In our implementation, we use a value of 10 as the maximum number of iterations allowed. For live scanning, our system works at the frame rate of 10 Hz with a input resolution of 640 × 480. Although our system is not working at the full frame rate of the input stream (30 Hz for the webcam), it still provides a smooth scanning experience. The detailed running statistics for each dataset are listed in Table 6.2.

7.3. Results

7.3.1 Camera Pose Estimation

We illustrate the estimated camera poses of obtained key frames as well as the reconstructed sparse model M_s composed of SIFT feature points in Figure 7. For all experiments, our system is able to match with the feature points inserted in the beginning and close the loop when scanning the target object back to the original viewpoints.

7.3.2 Reconstruction

The reconstructed models are illustrated in Figure 8. For the objects placed on a round table, we detect and remove the patches on the table through RANSAC plane fitting. Results demonstrate that our system is flexible for various types of objects. Note that the structural details, such as shoe ties,

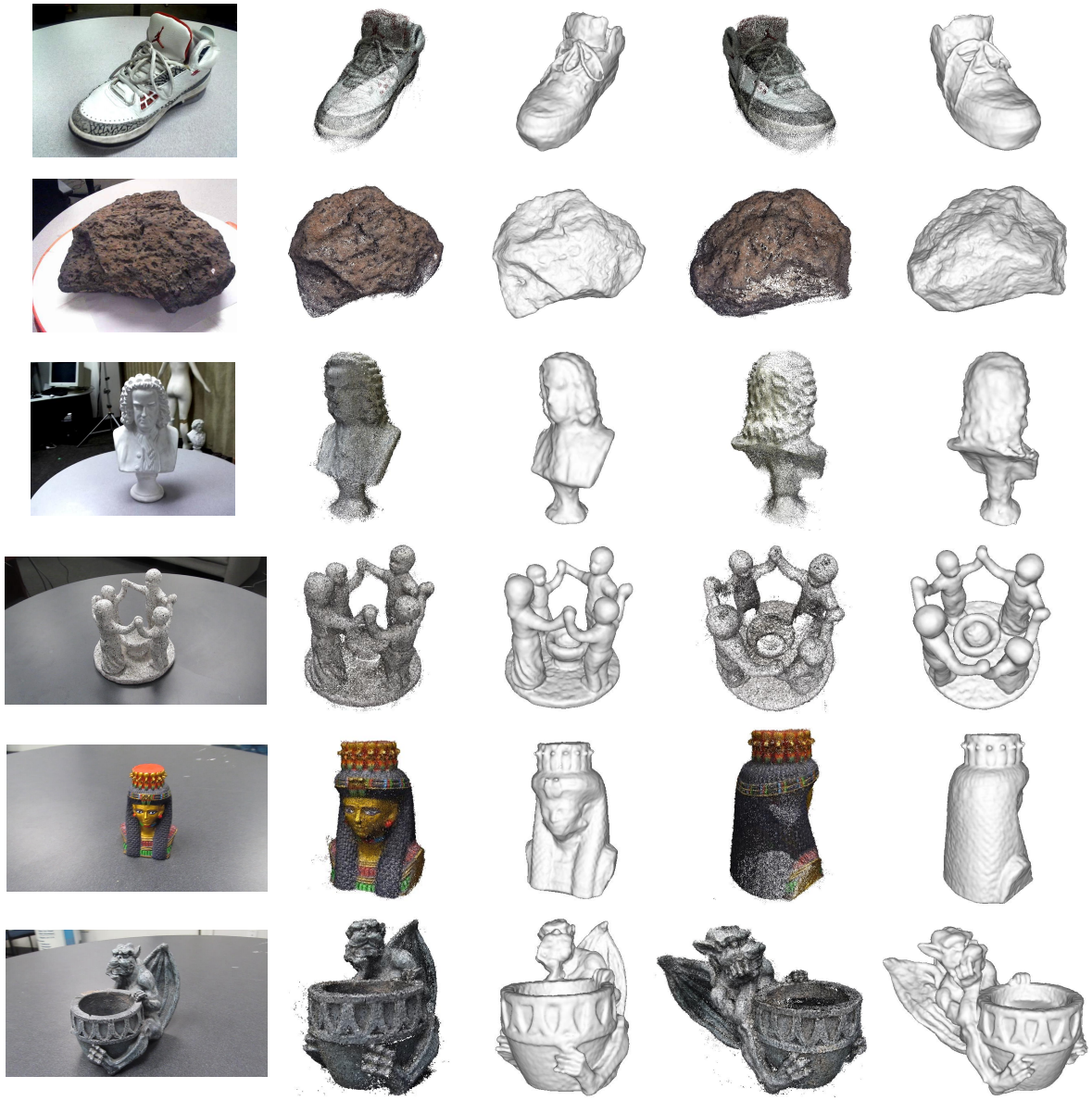

Figure 8. Reconstruction results including a sample input image, reconstructed dense patches and the detailed mesh model from 2 different views. From top to bottom: *shoe*, *stone*, *bazh*, *statue*, *cleopatra* and *gargoyl*

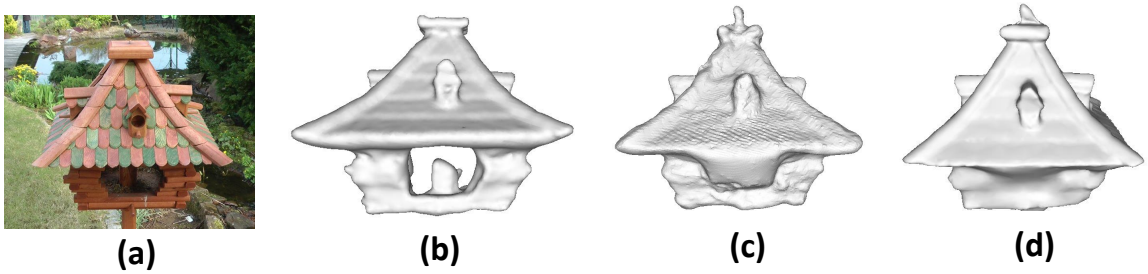

Figure 9. Comparsion of the reconstruction results on *birdhouse*. (a) a sample input image. (b) reconstructed mesh model from our system. (c) model generated from [19]. (d) model generated from Autodesk's 123D Catch

holes in the stone, are well captured in the reconstructed model. We also compare our results on *birdhouse* with the model reconstructed from [19] and Autodesk's 123D Catch [5], which is a mature commerical solution for image-based 3D modeling. As shown in Figure 9, our system is able to deal with the complicated geometric shapes, such as the holes within the birdhouse, with the patch-based representation.

7.3.3 Problematic cases

Texture-less objects, such as *bazh*, remain a challenge for our system where the dense optical flow is not reliable. We filter out erroneous patches through the visibility consistency test. There are still false patches producing artifacts in the reconstructed models. In the future, we attempt to address this problem combining the mesh-based global optimization as did in [5][19].

8. Conclusion and Future work

In this paper, we have proposed an end-to-end practical system for 3D model acquisition using a commodity hand-held camera. The pipeline starts with a live scanning stage where the immediate feedback of the reconstructed model allows the user to ensure the current state of the model before next movement. During scanning, structural details can be well captured by scanning the region of interest in close-up viewpoints. Our system is able to handle objects with complicated shapes using a patch-based model representation. After live scanning completed, an off-line model refinement procedure is taken to refine the reconstructed model and filter out erroneous patches.

Inspired by the recent advances in mobile dense reconstruction [18], one of our future work is to compare the live scanning stage of our system to the mobile application with and without the aid of on-device inertial measurement unit (IMU). Another future work is to incorporate global mesh optimization procedure to better handle the texture-less regions and occlusions.

9. Acknowledgments

This work was supported by grant DE-NA0001683 from the U.S. Department of Energy.

References

[1] J. Balzer and S. Soatto. Clam: Coupled localization and mapping with efficient outlier handling. In *CVPR*, 2013. 6

[2] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. 5

[3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. 2

[4] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 32(8):1362–1376, 2010. 2, 4

[5] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *CVPR*, 2009. 2, 8

[6] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 3

[7] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, 2007. 2

[8] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nistér, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *ICCV*, 2007. 2

[9] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, 2011. 2, 5

[10] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *CVPR*, 2010. 2

[11] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 2

[12] D. Nistér. An efficient solution to the five-point relative pose problem. *PAMI*, 26(6):756–770, 2004. 3

[13] Q. Pan, G. Reitmayr, and T. Drummond. Proforma: Probabilistic feature-based on-line rapid model acquisition. In *BMVC*, 2009. 2

[14] M. Pollefeys, D. Nistér, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, et al. Detailed real-time urban 3d reconstruction from video. *IJCV*, 78(2):143–167, 2008. 2

[15] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 438–446. ACM, 2002. 2

[16] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001. 2

[17] S. N. Sinha, P. Mordohai, and M. Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *ICCV*, 2007. 2

[18] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3d reconstruction on mobile phones. In *ICCV*, 2013. 2, 5, 8

[19] B. Ummenhofer and T. Brox. Dense 3d reconstruction with a hand-held camera. In *DAGM*. 2012. 6, 7, 8

[20] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *CVPR*, 2011. 4

[21] C. Zach. Fast and high quality fusion of depth maps. In *3DPVT*, 2008. 2

[22] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *DAGM*, 2007. 3