Journal of Signal Processing Systems (2022) 94:329–343
https://doi.org/10.1007/s11265-021-01729-0

# Towards Real-time 3D Visualization with Multiview RGB Camera Array

**Jianwei Ke**[1] · **Alex J Watras**[1] · **Jae-Jun Kim**[1] · **Hewei Liu**[1] · **Hongrui Jiang**[1] · **Yu Hen Hu**[1]

Received: 1 June 2021 / Revised: 1 June 2021 / Accepted: 1 December 2021 / Published online: 27 January 2022


**Abstract**
A real-time 3D visualization (RT3DV) system using a multiview RGB camera array is presented. RT3DV can process multiple synchronized video streams to produce a stereo video of a dynamic scene from a chosen view angle. Its design objective is to facilitate 3D visualization at the video frame rate with good viewing quality. To facilitate 3D vision, RT3DV estimates and updates a surface mesh model formed directly from a set of sparse key points. The 3D coordinates of these key points are estimated from matching 2D key points across multiview video streams with the aid of epipolar geometry and trifocal tensor. To capture the scene dynamics, 2D key points in individual video streams are tracked between successive frames. We implemented a proof of concept RT3DV system tasked to process five synchronous video streams acquired by an RGB camera array. It achieves a processing speed of 44 milliseconds per frame and a peak signal to noise ratio (PSNR) of 15.9 dB from a viewpoint coinciding with a reference view. As a comparison, an image-based MVS algorithm utilizing a dense point cloud model and frame by frame feature detection and matching will require 7 seconds to render a frame and yield a reference view PSNR of 16.3 dB.

**Keywords** Real-time 3D reconstruction and rendering · Structure-from-Motion · Multiview feature tracking

## 1 Introduction

A multiview camera array contains multiple cameras mounted on a rigid rig, capturing videos in a synchronous manner. By properly arranging camera orientations, one may estimate a dynamic 3D model of the foreground objects from the component videos and synthesize a new video of stereo vision of these objects from a desired view angle. A real-time 3D visualization (RT3DV) system consisting of a camera array and a processing platform will synthesize the stereo video when the camera array is capturing the dynamic scene.

An RT3DV system is very similar to the multi-view stereo (MVS) algorithm [1–5] in that a 3D surface model will be estimated based on multiple images (video frames). However, the primary design objective of current MVS systems is to accurately reconstruct a 3D surface model of a foreground object. As such, a dense point cloud 3D model [1–5] often needs to be estimated based on time-consuming optimization procedures. Hence, many of the traditional MVS pipeline modules may not be cost-effective. RT3DV, on the other hand, is developed to output a stereo video from a given viewing direction in real-time. A 3D model is needed here only to provide depth sensation of a stereo vision from a given view orientation.

In developing the RT3DV system, we focus on developing low complexity 3D surface model estimation and update methods such that the resulting algorithm may be executed on a commodity computing platform at a video frame rate (real-time). The low-complexity 3D surface model contains

This work was supported by the National Institute of Biomedical Imaging and Bioengineering (NIBIB) of the US National Institutes of Health (NIH) under award number R01EB019460.

✉ Jianwei Ke
jke9@wisc.edu

Alex J Watras
watras@wisc.edu

Jae-Jun Kim
JKIM207@mgh.harvard.edu

Hewei Liu
hliu265@wisc.edu

Hongrui Jiang
hongrui@engr.wisc.edu

Yu Hen Hu
yhhu@wisc.edu

[1] The Department of Electrical and Computer Engineering, University of Wisconsin-Madison, 1415 Engineering Drive, Madison WI 53706, USA

very few triangular meshes, formed with a sparse set of 3D key points. Periodically, the coordinates of these 3D key points are estimated from matching 2D key points in temporally synchronized video frames of multiple video streams via epipolar geometry and trifocal tensor. Between successive periodic refreshing of 2D key points detection and matching, their 2D positions are updated in intermediate frames using visual feature tracking. These innovative 3D modeling approaches reduce computation time per video frame by orders of magnitudes. Given the low-complexity 3D surface model and a desired view point, a stereo view may be rendered by mapping the texture of each triangular surface from the corresponding triangle in the closest view. Since the objective of RT3DV is 3D visualization, the image quality of the rendered video frame will be evaluated rather than the accuracy of the 3D coordinates of the 3D surface model.

We built a proof-of-concept camera array [6–8] and developed the RT3DV algorithm on this platform. In Fig. 1(a), five miniature cameras form a camera array acquiring videos synchronously. In Fig. 1(b), three objects used in the experiment are displayed. This camera array is mounted on the top of a surgical training box, as shown from the side in Fig. 1(c). The electronics for transmitting the multiview video streams to a close-by laptop are shown in Fig. 1(d). For this kind of embedded system application, trade-offs between the image quality of rendered 3D stereo video and corresponding computation costs become very important.

The technical contribution of the RT3DV system is the development of a low-complexity 3D reconstruction algorithm for 3D visualization of dynamic scenes. It does not require an expensive depth sensor (e.g., Kinect) and can be port to internet of things (IoT) devices.

In the rest of this paper, related works will be reviewed in Sect. 2. The RT3DV will be presented in detail in Sect. 3. Experiments with acquired multiview videos and results are presented in Sect. 4. Discussions and future works are in Sect 5.

## 2 Related Work

### 2.1 Image-based 3D Reconstruction

A typical image-based 3D reconstruction is the process of rebuilding the 3D shape of the original scene captured by multiview images. Existing 3D reconstruction algorithms aim to estimate a dense point cloud 3D representation of the scene. The objective is to enhance details of the estimated 3D surface model while conforming to visibility evidence. Computation complexity and computing time are of lesser concern. In this work, a typical image-based 3D reconstruction pipeline will be implemented as a baseline algorithm, of which the performance will be compared against the RT3DV algorithm.

The typical image-based 3D reconstruction pipeline consists of 3 steps: (a) 3D dense point cloud estimation, (b) 3D surface reconstruction, and (c) texture mapping. Before applying these three steps, a set of feature points (keypoints) will be detected at each view (image) using a feature detection algorithm such as SIFT [9], FAST [10], or SURF [11], etc. Then, a feature matching algorithm and RANSAC will be applied to jointly calibrate the camera intrinsic parameters as well as camera extrinsic parameters relative to a reference world coordinate. The baseline image-based 3D reconstruction pipeline is summarized in Fig. 2.

#### 2.1.1 Multiview Stereo (3D Dense Point Cloud Reconstruction)

Furukawa et al. [1] proposed a point-growing multiview stereo algorithm (PMVS) that iteratively grows the point cloud by adding new feature points according to the epipolar geometry while not violating visibility constraints. Bleyer et al. [12] introduced the PatchMatch algorithm [13, 14] for stereo matching. Initially, each pixel is assigned to a 3D plane randomly. A good plane that reduces a cost function will be propagated diagonally to neighboring pixels in an iterative manner. This PatchMatch algorithm has been adopted and extended in other works [2–5, 15]. Shen [15] employs PatchMatch stereo [12] for multiview reconstruction to generate a depth map for each image and imposes depth consistency over neighboring images. Based on the PatchMatch propagation scheme, Zheng et al. [5] propose a probabilistic graphical model for jointly view selection and depth estimation for each pixel without considering slanted 3D planes. Galliani et al. develop *Gipuma* [2] in which they use a diffusion-like propagation scheme to efficiently propagate good planes to half the amount of pixels at once, which utilizes the parallel computation of Graphic Process Unit (GPU). Xu et al. [4] adopt an asymmetric checkerboard propagation scheme based on the confidence of current neighbor hypotheses and jointly selects a subset of views for cost aggregation.

#### 2.1.2 3D Surface Reconstruction from Dense Point Clouds

When the dense point cloud is estimated using the PatchMatch method [16], the surface reconstruction problem may be posed as an energy minimization problem using the Delaunay triangulation. The energy cost function measures the agreement of inside/outside labeling of Delaunay tetrahedra based on the visibility constraints. A globally optimal tetrahedra labeling can be obtained by solving a graph S-T minimum cut problem. The method

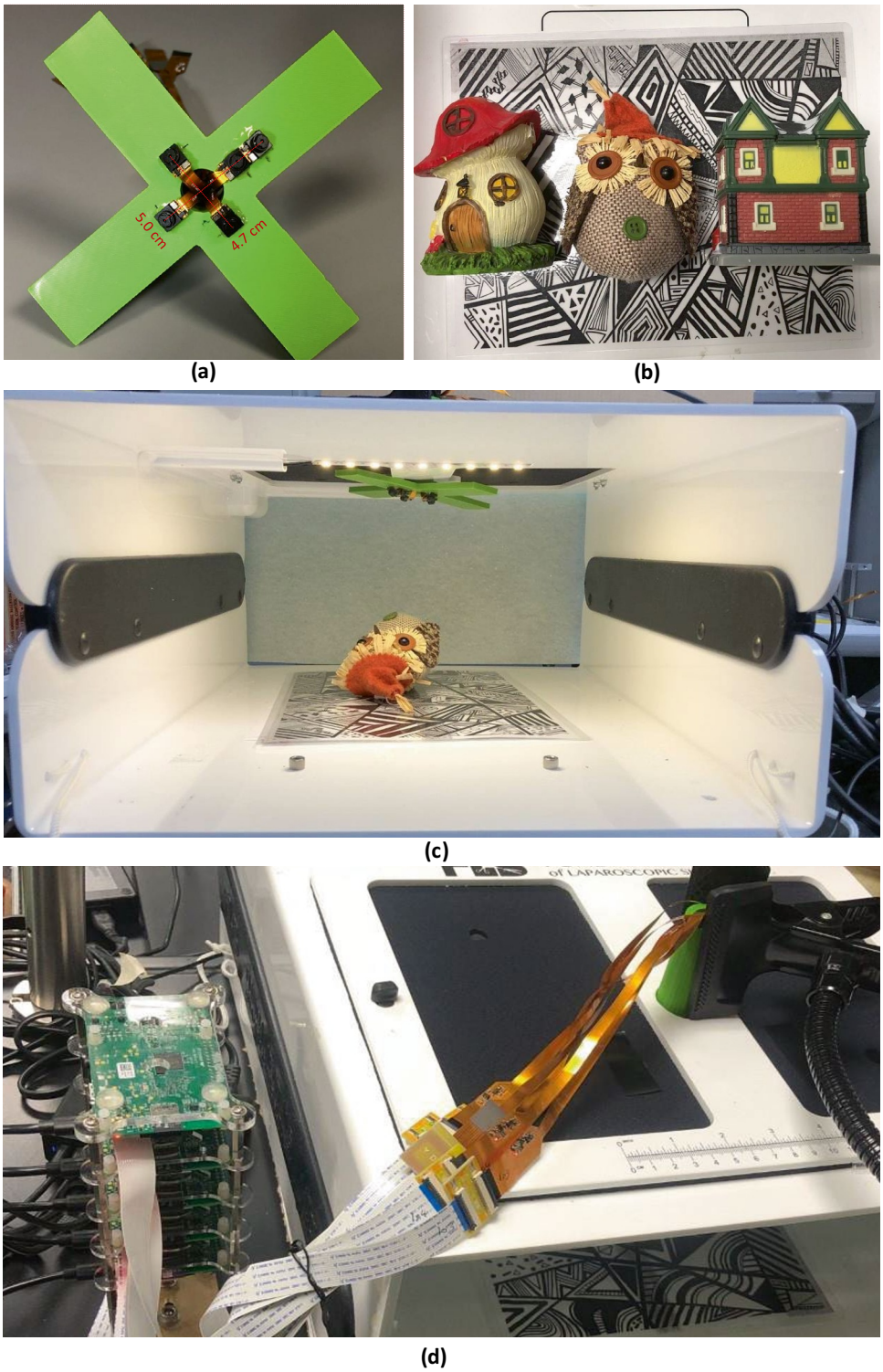



**Figure 1** (**a**) Micro-camera Array used to collect data (**b**) (**c**) Objects to reconstruct are placed in a Fundamentals of Laparoscopic Surgery (FLS) laparoscope trainer box while the camera array is recording. (**d**) Each Pi camera is connected to a Raspberry Pi.

described in [16], however, assumes a strong geometrical prior and may fail for weakly-supported surfaces well. Improvements were proposed in [17] and [18] which yield a more complete 3D surface at additional computation cost.

#### 2.1.3 Texture Mapping

Texture mapping [19–21] is the process of painting the triangular surface mesh with realistic color, texture, and shade using images acquired from one or more

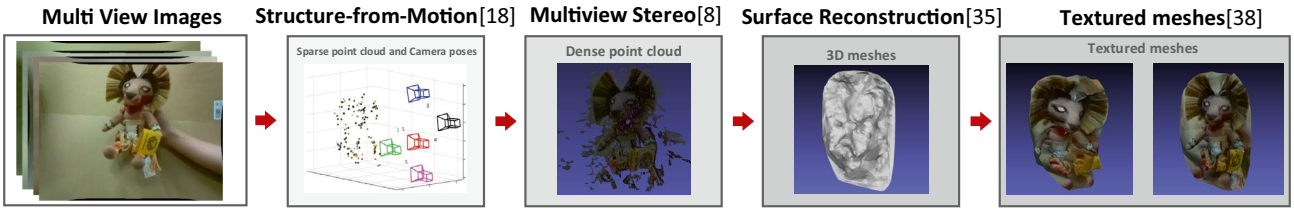



**Figure 2** Baseline image-based 3D reconstruction pipeline.

appropriated cameras. The selection of camera(s) for texture mapping is formulated as a multi-label Markov random filed energy optimization problem. Each 3D triangular mesh will be assigned to a close-by view so that its appearance can be warped from a triangular area in the video frame with matching vertices. In [20], the selection criterion is to align the surface normal of the triangular mesh to the optical axis of the view. A global color adjustment and a local Poisson editing are applied to minimize the seam line along the boundary of the triangular mesh. In [19], instead of one view (camera), the corresponding 2D triangular regions in multiple views (cameras) are collected and blended to yield the final texture of the mesh. It reduces the blurring and ghosting artifacts due to blending but cannot mitigate texture bleeding due to geometric registration error and camera calibration error. In [21], post-processing efforts are introduced to ensure color consistency and geometry consistency of textures in adjacent surface meshes.

## 2.2 Free-Viewpoint Video

Free-viewpoint video (FVV), a.k.a 4D video [22, 23] refers to a 3D video service that allows viewers to choose their preferred viewing angles freely. A 4D video, represented by a 3D surface model, associated texture maps, and the evolution of this 3D model over time (hence 4D), is generated to achieve this goal.

The MVS algorithm is the basis of FVV for developing and updating the 3D surface model. In [24], an initial dense correspondence is established to compute depth for each pixel. The estimations of depths are then filtered and used to refine the correspondence estimation in turn. Rendering from a given view angle is performed using both refined depths and updated correspondence. In [25] and [26], the shapes and the segmentation of dynamic objects are jointly computed and optimized. Many of the existing efforts focus on encoding and transmitting FVV streams, assuming the models have been obtained offline. The online acquisition of FVV has yet to be explored in depth. RT3DV developed in this work is perhaps the first effort to generate free-viewpoint video in real-time.

## 2.3 Real-time 3D Reconstruction with RGB-D Camera

A real-time template-based reconstruction method of dynamic scenes is demonstrated in [21], in which an online template was deformed to fit the depth data from an RGB-D camera. The template is non-rigidly tracked to provide a detail layer to account for high-frequency details. However, a rigid template must be captured at the beginning [21]. DynamicFusion [27] is the pioneering work for real-time and template-free 3D reconstruction of dynamic scenes using RGB-D cameras, where a canonical reference model is updated incrementally by unwarping depth measurements returned with a single RGB-D camera at a real-time rate. Several follow-ups improved the quality of reconstruction via additional constraints. For example, VolumeDeform [28] combines depth correspondences with robust sparse correspondences (SIFT) to avoid drift. Fusion4D [29] extend [27, 28] to a multiview scenario in which 8 RGB-D cameras capture depth data simultaneously, and multiple GPUs are used to compute the deformation field and the fusion of all data frame. However, the examples shown in [27–29] are limited to the reconstructed scene only undergoing slow motion and minor topological changes.

KillingFusion [30] estimates a dense deformation field in the TSDF space constrained by a damped Killing motion via a variational formulation, capturing more free movements. SobolevFusion [31] proposes to use Sobolev gradient flow to compute the deformation field and determine the voxel correspondences by matching the low-dimensional signatures of their Laplacian eigenfunctions, allowing large motion and topological changes of the scene. The recent work [32] uses the dual back RGB cameras of a VR device to achieve real-time 3D rendering. In [32], a video encoder is used to find a sparse $70 \times 70$ depthmap by block matching over a pair of rectified images, and then a fast Laplacian solver is used to smooth the depthmap. These methods all take in as input the depthmaps return by RGB-D cameras at a real-time rate. In our work, we tackle the problem of real-time 3D rendering using multiview RGB cameras, where depth information is derived from pure RGB images.

**Table 1** Summary of related 3D reconstruction and rendering algorithms.

| Algorithm | Input | Stationary cameras | Dynamic Scene | Hardware | Real-time | Output |
|---|---|---|---|---|---|---|
| Multi-view Stereo(MVS) [1–5, 15] | Multi-view images | √ | × | RGB cameras | × | Dense point cloud |
| Free-Viewpoint Video(FVV) [22–26] | Multi-view videos | × | √ | RGB Cameras | × | Rendered videos |
| Volumetric Methods based on RGB-D cameras [27–31] | Depthmaps | √ | √ | RGB-D camera | √ | 3D Volumetric Surface |
| RT3DV(ours) | Multi-view videos | √ | √ | RGB camera | √ | Rendered videos |

Table 1 summarizes the related 3D reconstruction algorithms and their characteristics compared to the RT3DV pipeline.

## 3 The RT3DV Algorithm

### 3.1 Overview

The inputs to the RT3DV algorithm are video streams acquired synchronously by cameras on a camera array. The outputs are a 3D surface model consisting of connected triangular meshes and a texture map (color, texture, and shade) for each triangular mesh. These outputs will be forwarded to a 3D rendering engine (Unity [33] in this work) to display a stereopsis video from given viewpoints. The RT3DV algorithm performs the following tasks for each video frame: (a) identifying 2D distinct feature points at each view (camera), (b) establishing correspondence of 2D feature points across all pairs of views, (c) estimating the 3D world coordinate of corresponding 2D feature points, (d) applying the Delaunay graph cut algorithm [34] to reconstruct the 3D triangular mesh surface model using the estimated 3D feature points as its vertices, and (e) estimating corresponding appearance map (texture, color, shade) for each triangle surface pigment.

When the algorithm is initiated (initiation mode), the cameras need to be calibrated to estimate their intrinsic parameters (focal length, pixel scaling, etc.) and extrinsic parameters (positions and poses). If the camera array remains stationary throughout the video, the camera parameters will be assumed available, and the initiation mode will not be executed anymore. Once the cameras are calibrated, the RT3DV algorithm will be executed in either a feature detection mode or a feature tracking mode. In the feature detection mode, 2D feature points will be detected at each camera's current frame. In the feature tracking mode, existing 2D features from the previous frame will be tracked. Leveraging the temporal correlations between successive video frames, the feature detection mode will be executed periodically with the feature tracking mode executed in between. The pipeline and block diagram of RT3DV are shown in Fig. 3.

### 3.2 Initiation

To initiate the RT3DV algorithm, camera calibration will be performed. An incremental structure from motion (iSfM) algorithm [35–38] will be applied to jointly estimate the camera intrinsic and extrinsic parameters and 3D coordinates of feature points.

First, the speeded-up robust feature (SURF) [11] detection algorithm is applied to the first video frames of all cameras to detect local features. Each detected local feature is represented by its 2D image coordinate within the video frame and a feature descriptor characterizing its local appearance. A fast matching algorithm FLANN [39] will then be applied to find figures across neighboring views having similar feature descriptors (appearance consistency). This appearance-based matching results will be verified using the epipolar geometry constraints. The RANSAC [40] algorithm will be applied to select a subset of matching 2D feature points of two views to estimate the corresponding fundamental matrix [41]. If the majority of remaining 2D feature points of both views also meet the epipolar constraints with the estimated fundamental matrix, the relative positions (extrinsic parameters) between this pair of cameras then may be determined. Matching 2D feature points that fail this geometric consistency check will be deemed as outliers and discarded. Based on the estimated positions and poses of cameras, 3D coordinates of the matching 2D feature points may be determined. Given these estimated 3D coordinates, one may proceed to refine the camera calibration. And then, the 3D coordinates will be refined further. This iterative Bundle Adjustment [42] process will converge as no further changes are observed. The iSfM repeats the above steps for one camera at a time until all cameras are processed. On completion of iSfM, the calibrated camera parameters and estimated 3D coordinates of feature points will be made available for subsequent frames.

### 3.3 Fast Reconstruction with Feature Tracking

Given the calibrated camera parameters, the set of matching 2D feature points, and corresponding 3D coordinates, one may leverage the temporal correlation of videos to

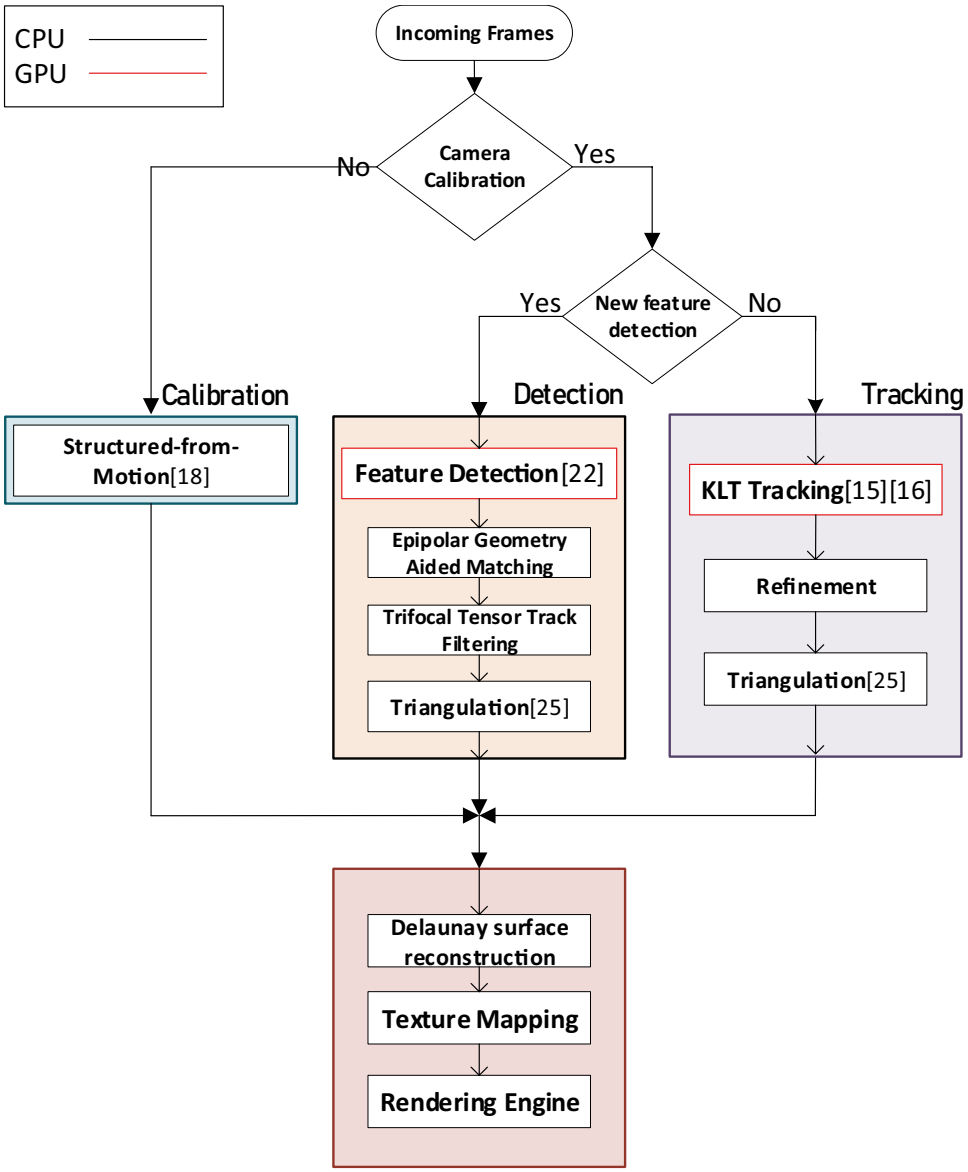



**Figure 3** Block diagram of the proposed pipeline RT3DV.

update the 2D feature positions using feature tracking instead of the time-consuming feature detection.

The Kanade–Lucas–Tomasi (KLT) feature tracker [43] will be used to track local movement of an existing 2D feature available from the previous frame. The outcome will further be refined by applying a block matching algorithm using the Sum of Absolute Differences (SAD) similarity metric.

$$\mathrm{SAD}(k,l) = \sum_{(i,j)\in N} |E_p(i,j) - E_c(i+k, j+l)| \tag{1}$$

where $E_p, E_c$ denote the previous and current frame and $N$ denotes a template window in a feature point's local neighborhood.

Since only existing 2D feature points from the previous frame are tracked, the feature correspondence relationship will remain unchanged unless a 2D feature disappears (cannot be tracked) due to dynamic scene change, in which case the track will be discarded. If a matching 2D feature point changes its position after tracking, its corresponding 3D coordinates will also be updated by triangulation using the stander DLT method for [41]. Otherwise, the previously estimated 3D coordinates will remain unchanged. This on-demand update strategy saves lots of computation when only a small fraction of feature points move between successive frames.

### 3.4 Fast Reconstruction based on Feature Detection

Feature tracking will capture movements of existing features in a dynamic scene. It does not, however, detects the presence of new features. Thus feature detection will

be performed periodically in the RT3DV algorithm. The period between two feature detection frames depends on prior knowledge of the dynamics of the scene and can be adjusted. We assume that the camera calibration parameters are available. Therefore, after new feature detection is performed, the robust feature matching process can be accelerated by enforcing the epipolar geometry [41] and aided by trifocal tensor [41]. Specifically, with calibrated cameras, the essential matrix $\mathbf{E}$ between any two cameras in the camera array is available. If a 2D feature point $\mathbf{x}$ in the video frame of one camera and another 2D feature point $\mathbf{x}'$ in the video frame of another camera correspond to the same 3D point, then

$$\mathbf{x}'^T\mathbf{E}\mathbf{x} = \mathbf{x}'^T\boldsymbol{\ell} = 0 \tag{2}$$

where $\boldsymbol{\ell} = \mathbf{E}\mathbf{x}$ is the *epipolar line*.

Instead of using Eq. (2) to verify the matches, we only retain the matches if the corresponding point is within 3 pixels from the epipolar line. This verification procedure only takes constant operations for each match, and thus the complexity is $O(N_m)$, where $N_m$ is the number of initial matches returned by FLANN. Then, we build feature tracks (2D feature correspondences across all views) from the remaining matches. The above procedure is called epipolar-geometry-aided matching. These tracks are then further refined by trifocal tensors.

Since all camera poses are available, we can quickly calculate the epipolar line for each 2D feature. A simple extension of epipolar geometry can help us find outliers: given a pair of matched points $(\mathbf{x}_1, \mathbf{x}_2)$, the third corresponding point $\mathbf{x}_3$ must pass both epipolar lines $\boldsymbol{\ell}_{13}$ and $\boldsymbol{\ell}_{23}$, where

$$\boldsymbol{\ell}_{13} = \mathbf{K}_3^{-T}\hat{\mathbf{T}}_{13}\mathbf{R}_{13}\mathbf{K}^{-1}\,\mathbf{x}_1 \tag{3}$$

$$\boldsymbol{\ell}_{23} = \mathbf{K}_3^{-T}\hat{\mathbf{T}}_{23}\mathbf{R}_{23}\mathbf{K}_2^{-1}\,\mathbf{x}_3 \tag{4}$$

In principle, we can determine $\mathbf{x}_3$ by intersecting $\boldsymbol{\ell}_{13}$ and $\boldsymbol{\ell}_{23}$:

$$\mathbf{x}_3 = \boldsymbol{\ell}_{13} \times \boldsymbol{\ell}_{23} \tag{5}$$

However, this approach fails when $\boldsymbol{\ell}_{13}$ and $\boldsymbol{\ell}_{23}$ are parallel and will be inaccurate if they are nearly colinear. This happens if the 3D point $\mathbf{X}$ lies on or near the trifocal plane defined by the three camera centers.

The degeneracy of the epipolar method can be avoided by using the trifocal tensor in three views which is analogous to fundamental matrix in two [41] views. The idea is to construct a homograpy by finding a plane defined by the back-projection of a line in the second view using the trifocal tensor. The homography and $\mathbf{x}_3$ are [41],

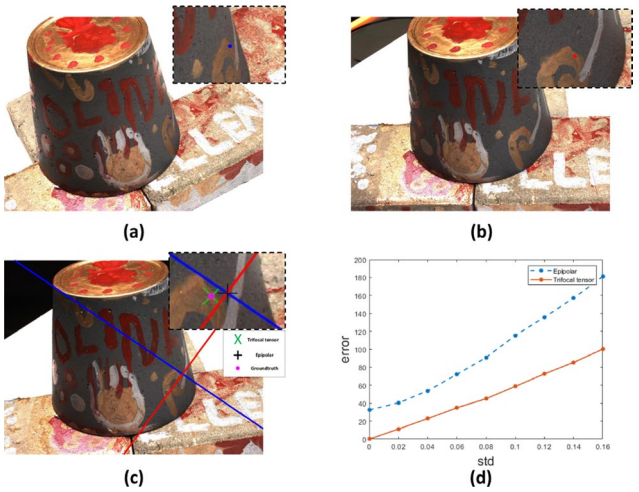



**Figure 4** (**a**) A feature point is shown in the first view. (**b**) The corresponding point in the second view. (**c**) The corresponding point in the third view can be found by intersecting two epipolar lines or trifocal tensor. The epipolar method is prone to error while trifocal tensor method is more robust. (**d**) Average transfer errors for the epipolar method and trifocal tensor method over DTU dataset. Gaussian noise with different standard deviation values are added to the corresponding image points.

$$h_i^k = l_2^j \mathscr{T}_i^{jk} \tag{6}$$

$$x_3^k = \sum_{i=1}^{3}\sum_{j=1}^{3} h_i^k x_1^i \tag{7}$$

where $\mathbf{x}_1 = (x_1^1, x_1^2, x_1^3)$, $\mathbf{x}_3 = (x_3^1, x_3^2, x_3^3)$, the line in the second view is $\mathbf{l}_2 = (l_2^1, l_2^2, l_2^3)$, and $\mathscr{T}_i^{jk} = a_i^j b_4^k - a_4^j b_i^k$ is the trifocal tensor, and $a_i^j, b_i^j$ are the $(i, j)$ element of the camera projection matrices $\mathbf{P}_2$ and $\mathbf{P}_3$ for the second and third view. A good chioce for $\mathbf{l}_2$ is the line that is through $\mathbf{x}_2$ and perpendiculat to the epipolar line. A comparison of accuracy for the epopolar method and the trifocal tensor method is shown in Fig. 4.

We couple the above procedure with RANSAC to filter outliers in a track and find the largest support set of corresponding points. Because each view has at most five elements (thus ten pairs possible), we can quickly iterate the ten possible pairs for each track. Once the outliers are filtered out, the stander DLT method [41] is used to triangulate for the 3D position for each track. As shown in Fig. 5, a better rendering result is achieved with the proposed trifocal tensor fitlering procedure. More discussion about Fig. 5 can be found in Experiment and Discussion section.

The Trifocal-Tensor-based Track Filtering is summarized as follow:

**Algorithm 1** Trifocal-Tensor-based Track Filtering

```
procedure TRACK FILTERING(Tracks, k)
    for each track T in Tracks do
        for each possible element pair t_i, t_j in T do
            Maintain set C for each pair
            for each t_r in T\{t_i, t_j} do
                Compute t̂_r by Equation 7
                if dist(t_r, t̂_r) < k then
                    Add t_r to C
                end if
            end for
        end for
        T = arg max_C |C|                ▷ keep the largest support set
    end for
end procedure
```

### 3.5 Surface Reconstruction and Texture Mapping

Once we have an accurate point cloud, the next step is to reconstruct a 3D surface model out of it. The desired surface reconstruction algorithm should not only work well with the sparse nature of our reconstructed point cloud but also be very computationally efficient to satisfy the real-time requirement. We use the Delaunay graph cut algorithm by Labatut et al. [34] because it is fast and robust to changes in point density, which helps to reconstruct difficult surface parts. In their work, they also showed that their approach is very robust against outliers.

Once we obtain the 3D surfaces, each surface is projected to all views that observe it. If the 3D surface is viewable by multiple views, we retrieve the texture from the view whose viewing angle is smallest with the normal of the 3D surface:

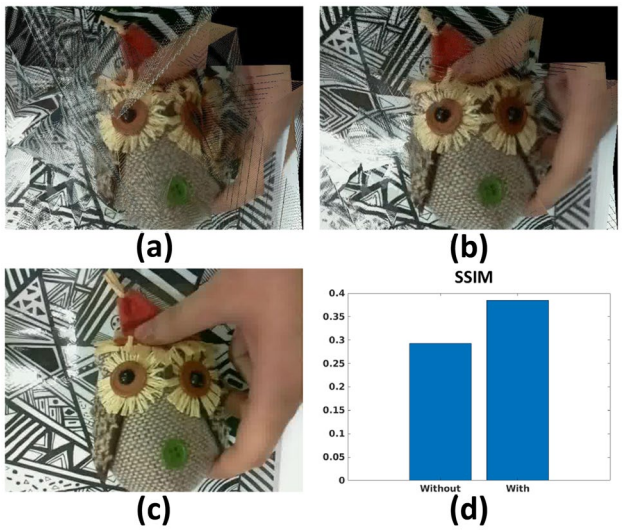



**Figure 5** Rendering result. (**a**) rendered view without trifocal tensor filtering (**b**) with trifocal tensor filtering (**c**) the ground truth (**d**) Structural similarity (SSIM) index for rendering result with and without trifocal tensor filtering.

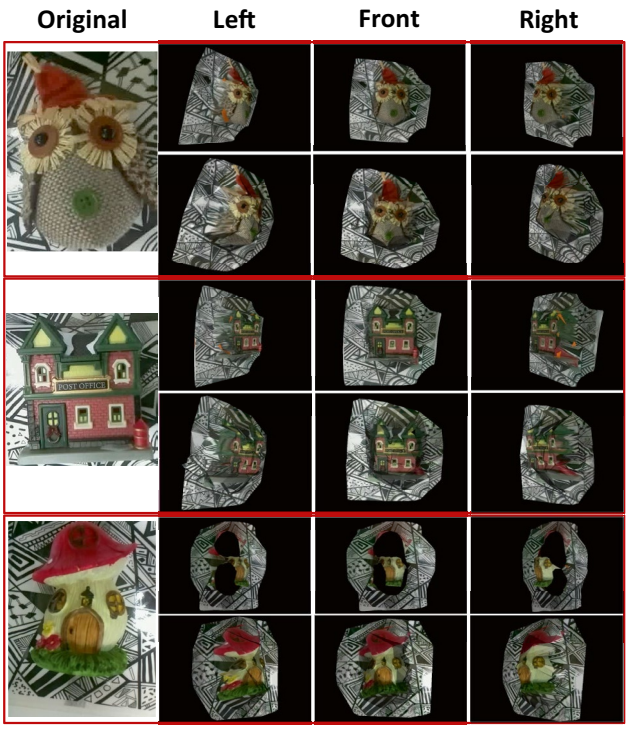



**Figure 6** Virtual views using textured 3D surfaces for the objects Owl, Post Office, and Mushroom at a given frame. The virtual view is generated by 3D rendering engine. The 1st, 2nd, 3rd columns are the left, front, and right virtual view. The 1st row is the result generated by the baseline pipeline, which takes around 5 s. The 2nd row is the result of the proposed pipeline, which takes around 42 ms on average.

$$\arg\min_{C} \langle \mathbf{v}_C, \mathbf{n} \rangle \tag{8}$$

where $\mathbf{v}_C$ is the viewing angle of view $C$, and $\mathbf{n}$ is the normal of the 3D surface.

## 4 Experiment and Discussion

### 4.1 Setup and Protocol

#### 4.1.1 Hardware and Software Platform

We evaluate the proposed RT3DV algorithm using the hardware platform shown in Fig. 1. Five synchronous video

**Table 2** Timing for baseline pipeline.

| Objects | Dense [2] | Surface | Texture | Total |
|---|---|---|---|---|
| Mushroom | 1.4 s | 1.2 s | 1.4 s | 4.0 s |
| Post Office | 1.3 s | 2.0 s | 2.1 s | 5.4 s |
| Owl | 1.4 s | 1.9 s | 2.4 s | 5.7 s |

**Table 3** Timing for RT3DV. Feature detection is performed every 10 frames.

| Objects | Detect | KLT | Tri | Surface | Texture | Avg |
|---|---|---|---|---|---|---|
| Mushroom | 66 ms | 9 ms | 10 ms | 10 ms | 13ms | 43 ms |
| Post Office | 57 ms | 8 ms | 9 ms | 8ms | 11ms | 38 ms |
| Owl | 63 ms | 10 ms | 11 ms | 10 ms | 12 ms | 45 ms |

streams are acquired from five micro-cameras (dark squares in Fig. 1(a)) simultaneously. The displacements between cameras are around 5 cm, as shown in Fig. 1(a). The camera array assembly is mounted on the top of an FLC laparoscope trainer box. The objects are placed at the bottom of the box and will be moved manually during the video capture to emulate a dynamic scene.

This platform is a prototype 3D visualization system developed to enhance the visualization of traditional laparoscope [6–8]. Each camera has a resolution of $640 \times 480$ pixels and has a frame rate of 30 frames per second (fps). Each camera is attached to a Raspberry Pi video capturing board, which compresses the video into Mpeg-4 format. The compressed video is then transmitted through Ethernet cables to a desktop PC to be processed. The PC is equipped with an 8-core 4.00 Hz i7-6770k CPU, a GeForce GTX 2080 Ti GPU, 16 GB main memory running Ubuntu 16.04 operating system.

For the baseline pipeline, we used the C++ implementation as SfM [35]. We chose the Cuda implementation of MVS [2]. The surface reconstruction [34] is implemented in C++ by [36]. RT3DV is implemented in C++ using OpenCV with CUDA 9 enabled, where SURF [11] and FLANN [39] matching run in GPU.

#### 4.1.2 Data Sets

We generated three sets of multiview video streams, one for each object, using the experiment platform described above. Besides, we perform the same experiment on the public available DTU MVS dataset [44], where underlying point clouds, the camera poses, and the images for each camera are all available. Since the objects are stationary, we first translate the underlying point cloud and then back-project it to all cameras to generate the multiview videos of moving objects.

#### 4.1.3 Protocols

We ran the baseline pipeline and RT3DV on all the multiview video frames in both datasets. We chose the running time to be the time interval between the completion of texture mapping between the successive frames. For the baseline pipeline, we excluded SfM and only measured the running time between the end of SfM to the end of texture mapping because SfM is only performed once as the initiation step in RT3DV. The processing time per frame is the average running time of all successive frames for each video. We conducted three trials of the experiment and reported the average processing time per frame of the three trials.

### 4.2 Results

#### 4.2.1 Timing

For the three multiview videos collected in the FLC laparoscope trainer box, we remove the tracked features whose error is greater than five and triangulate the rest. The algorithm is set to re-detect features per 10 frames. The running time is related to the number of feature points being processed. With a large number of features, the processing time for tracking, epipolar-geometry-aided matching, and trifocal-tensor-based track filtering and surface reconstruction can worsen.The baseline pipeline tries to find the dense feature point cloud. However, for scenes that have few feature points, it fails to compute the scene geometry, which introduces holes in the reconstruction, as shown in the last row of Fig. 6. The timing for RT3DV and the baseline pipeline are summarized in Tables 2 and 3. The number of feature points and 3D triangles can be found in Table 4.

**Table 4** Comparison of number of features and triangles and PSNR.

| | Features | | Triangles | | PNSR | |
|---|---|---|---|---|---|---|
| | Baseline | Ours | Baseline | Ours | Baseline | Ours |
| Mushroom | 14733 | 370 | 29418 | 715 | 14.9 dB | 14.4 dB |
| Post Office | 15710 | 281 | 31383 | 585 | 17.1 dB | 16.1 dB |
| Owl | 17637 | 303 | 35240 | 622 | 16.9 dB | 16.0 dB |

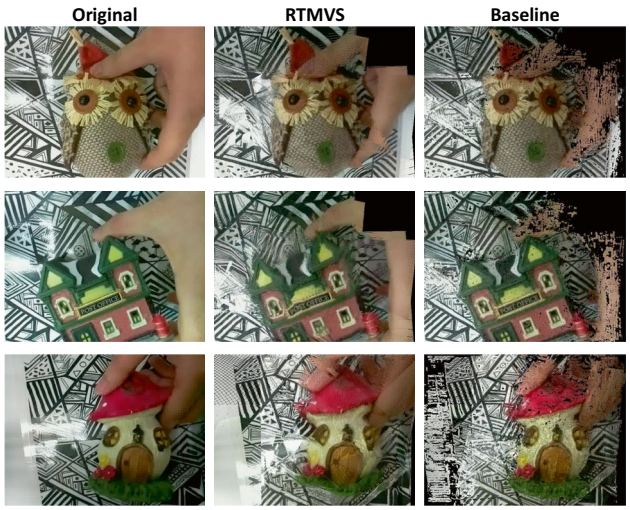



**Figure 7** Rendered view to an original view using depthmaps only. Images in the 1st column are original images. The 2nd column are generated by RT3DV. The 3rd column are generated by the Patchmatch-based MVS implemented by Galliani et al. [2].

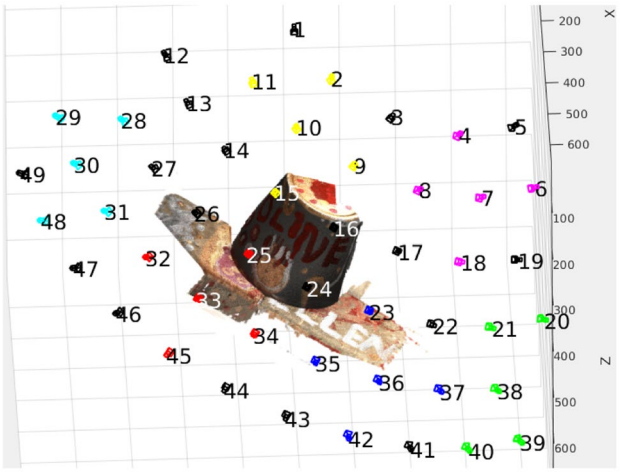


**Figure 9** Camera configuration for computing PNSR for virtual view. Each camera configuration has five cameras. We reconstruct the middle view and compute the PSNRs with the real image of the middle view using RT3DV and the baseline pipeline. Config 1-6 correspond to camera group of Green, Blue, Red, Cyan, Magenta, Yellow.

#### 4.2.2 Quality Metric and Evaluation

A key result of this work is that the visual quality of rendered images using a sparse point cloud is comparable to that using a dense point cloud. To facilitate objective visual quality evaluation, we adopt a protocol similar to the *leave one out* cross-validation method: we render a view at a viewing angle that coincides with one of the cameras and compute the peak signal to noise ratio (PSNR) between the rendered video frame and the acquired video frame (ground truth) without using any video frames from that validation camera.

Figure 7 shows the original image and the rendered view generated by our method and the baseline pipeline. In our experiment, we track feature points for ten frames and re-detect new feature points. Figure 8 shows the tracking result of our method as opposed to the classic pipeline. The averaged PSNR of Fig. 8 is recorded Table 4. The experiment is repeated with and without the epipolar geometry aided matching and the proposed trifocal tensor based filtering procedure. The rendered view and the structural similarity (SSIM) are computed and shown in Fig. 5. A qualitatively and quantitatively better result is obtained with the proposed matching and filtering procedure.

For the DTU dataset, the camera configurations are shown in Fig. 9. We test the baseline pipeline and RT3DV on four objects (object 1, 5, 6, 45) for six camera configurations (Green, Blue, Red, Cyan, Magenta, Yellow), as shown in Fig. 9. For each camera configuration, we reconstruct the image of the center view and computer the PSNRs with the original image using RT3DV and the baseline pipeline. Figure 10 shows the generated view. Table 5 and Fig. 11 show the PNSR and averaged processing time of our pipeline and the baseline pipeline, from which we see that the results of RT3DV and the baseline pipeline have similar PSNR but RT3DV is orders of magnitude faster than the baseline.

## 5 Discussion

**Number of features and processing time.** The processing time of the proposed pipeline heavily relies on the accuracy and the number of feature points extracted from the scene.

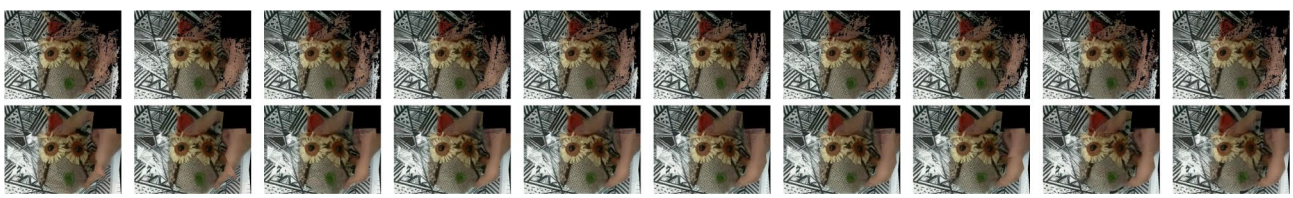


**Figure 8** From left to right are the rendered view for the moving objects. The 1st row are the results of the baseline pipeline which takes more than 5s per frame. The 2nd row are the rendered results for RT3DV that takes 44ms per frame on average.

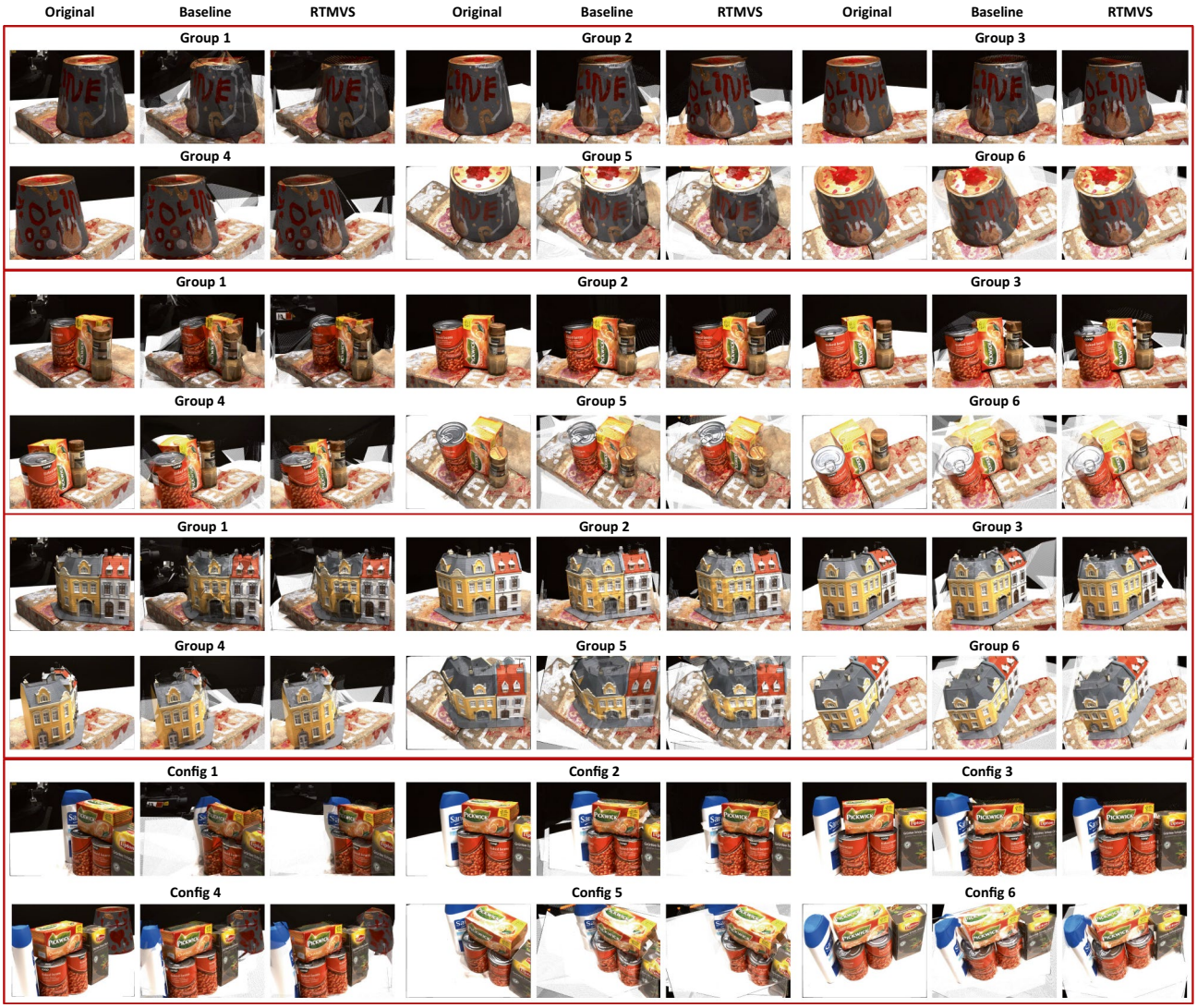



**Figure 10** Generated virtual center views for object 1, object 5, object 6, object 45 (in order) of the DTU dataset [44] by baseline pipeline and RT3DV.

Our virtual view is generated by rendering the 3D model computed by the sparse point cloud. If the number of points in the point cloud is too few, the point cloud will not capture the 3D geometry of the scene well. On the other hand, if the number of features is too large with similar accuracy, the reconstruction would take too long to complete, as the number of features has a direct relation to each stage of the pipeline. In our experiment of the laparoscope training box, the number of features is between 280 to 400, and the averages reconstruction time per frame is around 42 ms.

**Stationary camera poses.** The proposed pipeline assumes the relative camera poses are stationary. If the relative camera poses have changed, then we need to calibrate the cameras, which can be done by running SfM from scratch.

**Table 5** PSNR and Processing time per frame for reconstructing the middle view of different camera configurations in Fig. 9.

| (PSNR,time) | | Config 1 | Config 2 | Config 3 | Config 4 | Config 5 | Config 6 |
|---|---|---|---|---|---|---|---|
| Object1 | RT3DV | (14.28 dB, 215 ms) | (17.06 dB, 251ms) | (16.86 dB, 242ms) | (16.76 dB, 212 ms) | (15.96 dB, 245 ms) | (17.78 dB, 199 ms) |
| | Baseline | (15.49 dB, 14929 ms) | (17.12 dB, 17745 ms) | (16.54 dB, 16329 ms) | (17.02 dB, 16936 ms) | (18.41 dB, 17789 ms) | (19.90 dB, 17622 ms) |
| Object5 | RT3DV | (13.85 dB, 328 ms) | (16.46 dB, 342 ms) | (15.57 dB, 383 ms) | (15.48 dB, 359 ms) | (15.57 dB, 461 ms) | (17.76 dB, 518 ms) |
| | Baseline | (14.18 dB, 14993 ms) | (16.45 dB, 15643 ms) | (15.69 dB, 15867 ms) | (15.71 dB, 15709 ms) | (17.34 dB, 16878 ms) | (18.14 dB, 17752 ms) |
| Object6 | RT3DV | (14.34 dB, 383 ms) | (15.75 dB, 405 ms) | (16.71 dB, 377 ms) | (15.73 dB, 318 ms) | (14.96 dB, 440 ms) | (19.43 dB, 457 ms) |
| | Baseline | (13.60 dB, 15845 ms) | (16.62 dB, 16871 ms) | (15.76 dB, 17025 ms) | (16.86 dB, 16470 ms) | (13.31 dB, 17129 ms) | (18.28 dB, 17241 ms) |
| Object45 | RT3DV | (13.48 dB, 382 ms) | (17.59 dB, 339 ms) | (18.17 dB, 375 ms) | (18.84 dB, 366 ms) | (15.26 dB, 351 ms) | (18.45 dB, 371 ms) |
| | Baseline | (15.08 dB, 13863 ms) | (19.01 dB, 14967 ms) | (17.87 dB, 15598 ms) | (19.52 dB, 15475 ms) | (15.84 dB, 14614 ms) | (18.62 dB, 15736 ms) |

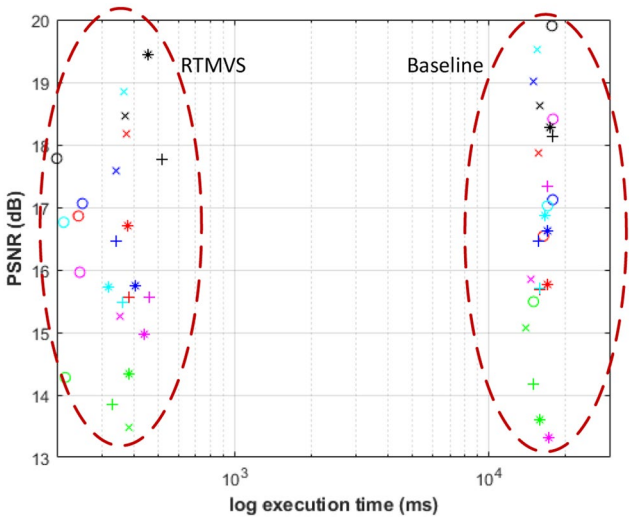



**Figure 11** Comparison of processing time per frame and PSNR between RT3DV and the baseline pipeline. Config 1-6 correspond to Green, Blue, Red, Cyan, Magenta, Yellow. Object 1, 5, 6, 45 correspond to markers of cycle, plus, star, cross.

## 6 Conclusion

In this work, we propose RT3DV for near-field scenes. The proposed algorithm utilizes the temporal and spatial correlation of multiview videos and is faster than the state-of-the-art pipeline in order of magnitude. While the state-of-the-art pipeline reconstructs fine details on parts of the scene, it introduces holes on the part that has fewer features. Our efficient and straightforward pipeline can preserve the integrity of the scene and provide an adequate visualization result.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

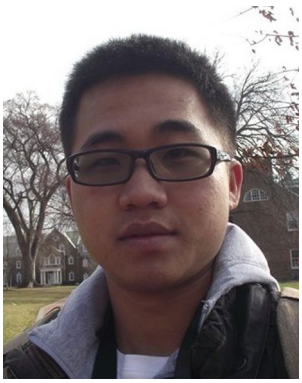
## References

1. Furukawa, Y., & Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(8), 1362–1376. https://doi.org/10.1109/TPAMI.2009.161
2. Galliani, S., Lasinger, K., & Schindler, K. (2015). Massively parallel multiview stereopsis by surface normal diffusion. In *2015 IEEE International Conference on Computer Vision (ICCV)* (pp. 873–881). https://doi.org/10.1109/ICCV.2015.106
3. Shen, S. (2013). Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. *IEEE Transactions on Image Processing, 22*(5), 1901–1914. https://doi.org/10.1109/TIP.2013.2237921
4. Xu, Q., & Tao, W. (2018). Multi-view stereo with asymmetric checkerboard propagation and multi-hypothesis joint view selection.
5. Zheng, E., Dunn, E., Jojic, V., & Frahm, J. M. (2014). Patchmatch based joint view selection and depthmap estimation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1510–1517). https://doi.org/10.1109/CVPR.2014.196
6. Kim, J. J., Watras, A., Liu, H., Zeng, Z., Greenberg, J. A., Heise, C. P., Hu, Y. H., & Jiang, H. (2018). Large-field-of-view visualization utilizing multiple miniaturized cameras for laparoscopic surgery. *Micromachines, 9*(9). https://doi.org/10.3390/mi9090431. https://www.mdpi.com/2072-666X/9/9/431
7. Watras, A., Ke, J., Zeng, Z., Kim, J. J., Liu, H., Jiang, H., & Hu, Y. H. (2017). Parallax mitigation for real-time close field video stitching. In *2017 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 568–571). https://doi.org/10.1109/CSCI.2017.349

8. Watras, A.J., Kim, J. J., Liu, H., Hu, Y.H., & Jiang, H. (2018). Optimal camera pose and placement configuration for maximum field-of-view video stitching. *Sensors, 18*(7). https://doi.org/10.3390/s18072284. https://www.mdpi.com/1424-8220/18/7/2284
9. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94
10. Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. In A. Leonardis, H. Bischof, & A. Pinz (Eds.), *Computer Vision - ECCV 2006* (pp. 430–443). Berlin, Heidelberg: Springer Berlin Heidelberg.
11. Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In A. Leonardis, H. Bischof, & A. Pinz (Eds.), *Computer Vision - ECCV 2006* (pp. 404–417). Berlin, Heidelberg: Springer Berlin Heidelberg.
12. Michael Bleyer, C.R., & Rother, C. (2011). Patchmatch stereo - stereo matching with slanted support windows. In *Proceedings of the British Machine Vision Conference* (pp. 14.1–14.11). BMVA Press. http://dx.doi.org/10.5244/C.25.14
13. Barnes, C., Shechtman, E., Finkelstein, A., & Goldman, D. B. (2009). Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics, 28*(3). https://doi.org/10.1145/1531326.1531330
14. Besse, F., Rother, C., Fitzgibbon, A., & Kautz, J. (2014). PMBP: Patchmatch belief propagation for correspondence field estimation. *International Journal of Computer Vision, 110*(1), 2–13. https://doi.org/10.1007/s11263-013-0653-9
15. Besse, F., Rother, C., Fitzgibbon, A., & Kautz, J. (2014). PMBP: Patchmatch belief propagation for correspondence field estimation. *International Journal of Computer Vision, 110*(1), 2–13. https://doi.org/10.1007/s11263-013-0653-9
16. Jancosek, M., & Pajdla, T. (2011). Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011* (pp. 3121–3128). https://doi.org/10.1109/CVPR.2011.5995693
17. Jancosek, M., & Pajdla, T. (2014). Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces. *International Scholarly Research Notices, 2014*, 798595. https://doi.org/10.1155/2014/798595
18. Waechter, M., Moehrle, N., & Goesele, M. (2014). Let there be color! large-scale texturing of 3d reconstructions. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision - ECCV 2014* (pp. 836–850). Cham: Springer International Publishing.
19. Fu, Y., Yan, Q., Yang, L., Liao, J., & Xiao, C. (2018). Texture mapping for 3d reconstruction with RGB-d sensor. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4645–4653). https://doi.org/10.1109/CVPR.2018.00488
20. Zhou, Q. Y., & Koltun, V. (2014). Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics, 33*(4). https://doi.org/10.1145/2601097.2601134
21. Zollhöfer, M., Nießner, M., Izadi, S., Rehmann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., Theobalt, C., & Stamminger, M. (2014). Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics, 33*(4). https://doi.org/10.1145/2601097.2601165
22. Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., Hoppe, H., Kirk, A., & Sullivan, S. (2015). High-quality streamable free-viewpoint video. *ACM Transactions on Graphics, 34*(4). https://doi.org/10.1145/2766945
23. Lee, C. C., Tabatabai, A., & Tashiro, K. (2015). Free viewpoint video (FVV) survey and future research direction. *APSIPA Transactions on Signal and Information Processing, 4*. https://doi.org/10.1017/ATSIP.2015.18
24. Lipski, C., Klose, F., & Magnor, M. (2014). Correspondence and depth-image based rendering a hybrid approach for free-viewpoint video. *IEEE Transactions on Circuits and Systems for Video Technology, 24*(6), 942–951. https://doi.org/10.1109/TCSVT.2014.2302379
25. Mustafa, A., Kim, H., Guillemaut, J. Y., & Hilton, A. (2016). Temporally coherent 4d reconstruction of complex dynamic scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4660–4669). https://doi.org/10.1109/CVPR.2016.504
26. Mustafa, A., Kim, H., Guillemaut, J. Y., & Hilton, A. (2015). General dynamic scene reconstruction from multiple view video. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.
27. Newcombe, R. A., Fox, D., & Seitz, S. M. (2015). Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 343–352). https://doi.org/10.1109/CVPR.2015.7298631
28. Innmann, M., Zollhöfer, M., Nießner, M., Theobalt, C., & Stamminger, M. (2016). VolumeDeform: Real-time Volumetric Non-rigid Reconstruction.
29. Dou, M., Khamis, S., Degtyarev, Y., Davidson, P., Fanello, S. R., Kowdle, A., Escolano, S. O., Rhemann, C., Kim, D., Taylor, J., Kohli, P., Tankovich, V., & Izadi, S. (2016). Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics, 35*(4). https://doi.org/10.1145/2897824.2925969
30. Slavcheva, M., Baust, M., Cremers, D., & Ilic, S. (2017). Killingfusion: Non-rigid 3d reconstruction without correspondences. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5474–5483). https://doi.org/10.1109/CVPR.2017.581
31. Slavcheva, M., Baust, M., & Ilic, S. (2018). Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2646–2655). https://doi.org/10.1109/CVPR.2018.00280
32. Chaurasia, G., Nieuwoudt, A., Ichim, A. E., Szeliski, R., & Sorkine-Hornung, A. (2020). Passthrough+: Real-time stereoscopic view synthesis for mobile mixed reality. *Proceedings of the ACM in Computer Graphics and Interactive Techniques, 3*(1). https://doi.org/10.1145/3384540
33. Technologies, U. (2019). Unity. https://unity.com/
34. Labatut, P., Pons, J., & Keriven, R. (2009). Robust and efficient surface reconstruction from range data. *Computer Graphics Forum, 28,* 2275–2290. https://doi.org/10.1111/j.1467-8659.2009.01530.x
35. Moulon, P., Monasse, P., & Marlet, R. (2013). Adaptive structure from motion with a contrario model estimation. In K. M. Lee, Y. Matsushita, J. M. Rehg, & Z. Hu (Eds.), *Computer Vision - ACCV 2012* (pp. 257–270). Berlin, Heidelberg: Springer Berlin Heidelberg.
36. Schnberger, J. L., & Frahm, J. M. (2016). Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4104–4113). https://doi.org/10.1109/CVPR.2016.445
37. Snavely, N., Seitz, S. M., & Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics, 25*(3), 835–846. https://doi.org/10.1145/1141911.1141964
38. Wu, C. (2013). Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision - 3DV 2013* (pp. 127–134). https://doi.org/10.1109/3DV.2013.25
39. Muja, M., & Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP International Conference on Computer Vision Theory and Applications* (pp. 331–340).
40. Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image

analysis and automated cartography. *Communications of the ACM, 24*(6), 381–395. https://doi.org/10.1145/358669.358692
41. Hartley, R., & Zisserman, A. (2003). *Multiple View Geometry in Computer Vision* (2nd ed.). USA: Cambridge University Press.
42. Zhang, J., Boutin, M., & Aliaga, D. G. (2006). Robust bundle adjustment for structure from motion. In *2006 International Conference on Image Processing* (pp. 2185–2188). https://doi.org/10.1109/ICIP.2006.312973
43. Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI'81* (vol. 2, pp. 674–679). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
44. Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., & Aans, H. (2014). Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 406–413). https://doi.org/10.1109/CVPR.2014.59

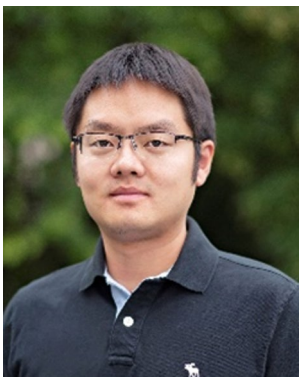




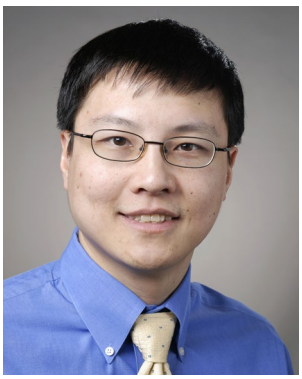
**Jianwei Ke (No photo)** received the B.S. Degree in Electrical Engineering from the University of Delaware with the highest distinction in 2015. He is a Ph.D. candidate in the ECE department at the University of Wisconsin-Madison since 2016. In 2020, he completed M.S. degrees in Electrical Engineering and Computer Science at the University of Wisconsin-Madison. His research interests span general computer vision and signal/image processing. He is currently working on image-based 3D reconstruction and real-time 3D reconstruction and rendering.

**Alex Watras (No photo)** was born in Minocqua, WI, in 1993. He received the B.S. in electrical engineering with a secondary major in Applied Mathematics from the University of Wisconsin - Madison, Madison, Wisconsin, in 2015. In 2018 He completed the M.S. degree in Electrical Engineering at the University of Wisconsin-Madison, and he is currently pursuing a Ph.D. in electrical engineering at the same time. His research focuses on real-time video stitching, optimal camera placement for small camera arrays, and parallax mitigation methods in video mosaics.

**Jae-Jun Kim (No photo)** received his Ph.D. in Bio and Brain Engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2015 after a B.S. and M.S. degree in Bio and Brain Engineering from KAIST in 2008 and 2010, respectively. In 2015, he was a postdoctoral researcher in Information and Electronics Research Institute at KAIST. He is currently a postdoctoral researcher at the Department of Electrical and Computer Engineering of University of Wisconsin-Madison, USA. His research interests are in biomedical engineering, bioinspired photonic devices, and smart and functional materials.



**Hewei Liu (No photo)** is an assistant scientist at the Department of Electrical and Computer Engineering in University of Wisconsin-Madison. His research interest including development of MEMS and sensors, micro- and nanofabrication, optics and photonic.



**Hongrui Jiang (No photo)** received the B.S. degree in physics from Peking University, Beijing, China, and the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, USA, in 1999 and 2001, respectively. From 2001 to 2002, he was a Post-Doctoral Researcher at the Berkeley Sensor and Actuator Center, University of California at Berkeley. He is currently the Vilas Distinguished Achievement Professor and the Lynn H. Matthias Professor in engineering with the Department of Electrical and Computer Engineering, and a Faculty Affiliate with the Department of Biomedical Engineering, the Department of Materials Science and Engineering, and the Department of Ophthalmology and Visual Sciences, and a member of the McPherson Eye Research Institute, University of Wisconsin–Madison. His research interests are in microfabrication technology, biological and chemical microsensors, microactuators, optical microelectromechanical systems, smart materials and micro-/nanostructures, lab-on-chip, and biomimetics, and bioinspiration. He is a Member of the Editorial Board of the IEEE/ASME JOURNAL OF MICROELECTROMECHANICAL SYSTEMS. Dr. Jiang is a fellow of the Institute of Physics, the Royal Society of Chemistry, and the American Institute for Medical and Biological Engineering. He was a recipient of the National Science Foundation CAREER Award and

the Defense Advanced Research Projects Agency Young Faculty Award in 2008, the H. I. Romnes Faculty Fellowship of the University of Wisconsin–Madison in 2011, the National Institutes of Health Director's New Innovator Award in 2011, the Vilas Associate Award of the University of Wisconsin in 2013, and the Research to Prevent Blindness Stein Innovation Award in 2016.

**Yu Hen Hu (No photo)** received BSEE from National Taiwan University, Taiwan ROC in 1976, and MSEE and PhD degrees from University of Southern California, Los Angeles, CA, USA in 1982. He was a member of faculty in the Electrical Engineering Department of Southern Methodist University, Dallas, Texas from 1983 to 1987. Since 1987, he has been with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison where he is currently a professor. In 1999, Dr. Hu is a visiting researcher at Bell Laboratories, Holmdel NJ, and at Microsoft Research – China, Beijing, China. He has been a visiting professor at National Taiwan University, Graduate Institute of Electronics, Taipei, Taiwan in 2007 and 2015. Dr. Hu's has broad research interests ranging from design and implementation of signal processing algorithms, computer aided design and physical design of VLSI, pattern classification and machine learning algorithms, and image and signal processing in general. He has published more than 380 technical papers, edited, and co-authored four books and many book chapters in these areas. Dr. Hu has served as an associate editor for the IEEE Transaction of Acoustic, Speech, and Signal Processing, IEEE signal processing letters, European Journal of Applied Signal Processing, Journal of VLSI Signal Processing, and IEEE Multimedia magazine. He has served as the secretary and an executive committee member of the IEEE signal processing society, a board of governor of IEEE neural network council representing the signal processing society, the chair of signal processing society neural network signal processing technical committee, and the chair of IEEE signal processing society multimedia signal processing technical committee. He has also served as a member of IEEE Jack S. Kilby Signal Processing Medal committee, and a steering committee member of the international conference of Multimedia and Expo on behalf of IEEE Signal processing society. Dr. Hu is a fellow of IEEE.