Vision, Modeling, and Visualization (2010)

Reconstructing Shape and Motion from Asynchronous Cameras

Felix Klose[1], Christian Lipski[1], Marcus Magnor[1]

[1]Computer Graphics Lab, TU Braunschweig

Abstract

We present an algorithm for scene flow reconstruction from multi-view data. The main contribution is its ability to cope with asynchronously captured videos. Our holistic approach simultaneously estimates depth, orientation and 3D motion, as a result we obtain a quasi-dense surface patch representation of the dynamic scene. The reconstruction starts with the generation of a sparse set of patches from the input views which are then iteratively expanded along the object surfaces. We show that the approach performs well for scenes ranging from single objects to cluttered real world scenarios.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Stereo, Time-varying imagery

1. Introduction

With the wide availability of consumer video cameras and their ever increasing quality at lower prices, multi-view video acquisition has become a widely popular research topic. Together with the large amount of processing power readily available today, multiple views are used as input data for high quality reconstructions. While the traditional two-view stereo reconstruction extends well to a multi-view scenario for static scenes, the complexity increases for scenes with moving objects. The most common way of approaching this problem is the use of synchronized image acquisition.

To loose the limitations that synchronized acquisition setups impose, we present our multi-view reconstruction approach that takes asynchronous video as input. Hence, no custom and potentially costly hardware with synchronized shutters is needed.

Traditional reconstruction algorithms rely on synchronous image acquisition, so that they can exploit the epipolar constraint. We eliminate this limitation and furthermore benefit from the potentially higher temporal sampling due to the different shutter times. With our approach, scene flow reconstruction with rolling shutters as well as heterogeneous temporal sampling, i.e. cameras with different framerates, is possible.

In Sect. 2 we give a short overview of the current research. Sect. 3 then gives an overview our algorithm. A detailed description of our approach is then given in Sect. 5-8, followed by our experimental results in Sect. 9, before we conclude in Sect. 10.

2. Related Work

When evaluating static multi-view stereo (MVS) algorithms, Seitz et al. [SCD*06] differentiated the algorithms by their basic assumptions. Grouping algorithms by their underlying model provides four categories: The volumetric approaches using discrete voxels in 3D space [KKBC07, SZB*07], the algorithms that evolute a surface [FP09b], reconstructions based on depth map merges [MAW*07, BBH08] and algorithms are based on the recovery of 3D points that are then used to build a scene model [FP09a, GSC*07].

While all the MVS approaches recover a scene model from multiple images, the limitations on the scene shown on the images vary. Algorithms that are based on visual hulls or require a bounding volume are more suited for multiple views of a single object. The mentioned point based methods on the other hand perform well on single objects and cluttered scenes.

Regarding the objective of scene motion recovery, the

DOI: 10.2312/PE/VMV/VMV10/171-177

term scene flow was coined by Vedula [VBR*99]. The 3D scene flow associates a motion vector with each input image point, corresponding to its velocity in scene space. The existing approaches to recover scene flow can be split into three groups based in their input data. The first group utilizes multiple precomputed optical flow fields to compute the scene flow [ZK01, VBR*05]. The second uses static 3D reconstructions at discrete timesteps and recovers the motion by registering the data [ZCS03, PKF05, PKF07]. A third family of algorithms uses spatio-temporal image derivatives as input data [NA02, CK02].

Besides the obvious connection between the structure and its motion, in current research the recovery largely remains split into two disjunct tasks. Wang et al. [WSY07] proposed an approach to cope with asynchronously captured data. However, their two-step algorithm relies on synthesizing synchronized intermediate images, which are then processed in a traditional way.

Our holistic approach simultaneously recovers geometry and motion without resampling the input images. We base the pipeline of our approach on the patch-based MVS by Furukawa et al. [FP09a], which showed impressive results for the reconstruction of static scenes. While Furukawa et al. explicitly remove non-static objects, i.e., spatially inconsistent scene parts, from scene reconstruction, we create a dynamic scene model where both object geometry and motion are recovered. Although we adapt the basic pipeline design, our requirement to cope with dynamic scenes and to reconstruct motion make fundamental changes necessary. E.g., our initialization and optimization algorithms have to take individual motion of a patch into account.

3. Overview

We assume that the input video streams show multiple views of the same scene. Since we aim to reconstruct a geometric model, we expect the scene to consist of opaque objects with mostly diffuse reflective properties.

In a preprocessing step the in- and extrinsic camera parameters for all images are estimated by sparse bundle adjustment [SSS06]. Additionally the sub-frame time offsets between the cameras have to be determined. Different methods have been explored in recent research to automatically obtain the sub-frame offset [MSMP08, HRT*09].

The algorithm starts by creating a sparse set of seed points in an initialization phase, and grows the seeds to cover the visible surface by iterating expansion, optimization and filter steps.

Our scene model represents the scene geometry as a set of small tangent plane *patches*. The goal is to reconstruct a tangent patch for the entire visible surface. Each patch is described by its position, normal and velocity vector.

The presented algorithm processes an *image group* at a time, which consists of images chosen by their respective temporal and spatial parameters. All patches extracted from an image group collectively form a dynamic model of the scene, that is valid for the timespan of the image group. The *image group timespan* is the time interval ranging from the acquisition of the first image of the group to the time the last selected image was recorded.

Since the scene model has a three dimensional velocity vector for each surface patch, linear motion in the scene space is reconstructed. The motion only needs to be linear for the image group timespan.

The result of our processing pipeline is a patch cloud. Each patch is described by its position, orientation and (linear) motion. While it is unordered in scene space, each pixel in image space (of each reference image) is assigned to a single patch or no patch. A visualization of our quasi-dense scene reconstruction is shown in Fig. 1.

4. Image Selection and Processing Order

To reconstruct the scene for a given time t' a group of images is selected from the input images. The image group G contains three consecutive images $I^{-1,0,1}$ from each camera, where the middle image I^0 is the image from the camera taken closest to t' in time.

The acquisition time $t(I) = c_{\text{offset}} + \frac{n}{c_{\text{fps}}}$ of an image from the camera c is determined by the camera time offset c_{offset}, the camera framerate c_{fps} and the frame number n.

During the initialization step of the algorithm, the processing order of the images is important and it is favorable to use the center images first. For camera setups where the cameras roughly point at the same scene center the following heuristic is used to sort the image group in ascending order:

$$s(I) = \sum_{I' \in G} |\Phi(I) - \Phi(I')| \tag{1}$$

Where $\Phi(I)$ is the position of the camera that acquired the image I. When at least one camera is static, $s(I)$ can evaluate to identical values for different images I. These the images with identical values $s(I)$ are ordered by the distance of their acquisition time from t'.

5. Initialization

To reconstruct the initial set of patches it is necessary to find pixel correspondences within the image group. In classical stereo vision, two pixel coordinates in two images are sufficient to triangulate the 3D position. Since our reconstruction process does not only determine the position, but also the velocity of a point in the scene, more correspondences are needed.

The search for correspondences is further complicated by the nature of our input data. One of the implications of the

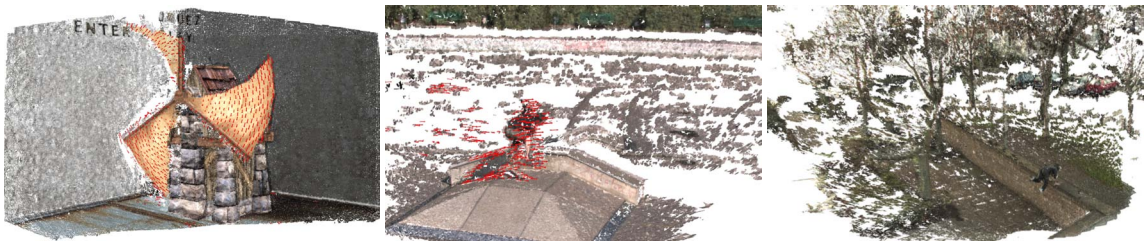


Figure 1: *Visualization of reconstructed scenes. The patches are textured according to their reference image. Motion is visualized by red arrows.*

asynchronous cameras is, that no epipolar geometry constraints can be used to reduce the search region for the pixel correspondence search.

We compute a list of interest points for each image $I' \in G$. An Harris Corner detector is used to select the points of interest. The intention is to select points which can be identified across multiple images. A local maximum suppression is performed, i.e., only the strongest response within a small radius is considered. Every interest point is then described by a SURF [BETG08] descriptor. In the following, an interest point and its descriptor is referred to as a feature.

For each image I', every feature f extracted from that image is serially processed. A given feature f_0 is matched against all features from the other images. The best match for each image is added into a candidate set C.

The candidate set C may contain outliers. This is due to wrong matchings and the fact, that the object on which f_0 is located may not be visible in all camera images. A subset for reconstructing the surface patch has to be selected. To find such a subset a RANSAC based method is used:

First a set S of $\Theta - 1$ features is randomly sampled from C. Then the currently processed feature f_0 is added to the set S. The value of $|S| = \Theta$ can be varied depending on the input data. For all our experiments we chose $\Theta = 6$.

The sampled features in S are assumed to be evidence of a single surface. Using the constraints from feature positions and camera parameters and assuming a linear motion model, a center position $\vec{c}$ and a velocity $\vec{v}$ are calculated. The details of the geometric reconstruction are given later (section 5.1).

The vectors $\vec{c}$ and $\vec{v}$ represent the first two parameters of a new patch P. The next RANSAC step is to determine which features from the original candidate set C consent to the reconstructed patch P. The patch is reprojected into the images $I' \in G$ and the distance from the projected position to the feature position in I' is evaluated. After multiple RANSAC iterations the largest set $T \subset C$ of consenting features found is selected.

Although the reconstruction equation system is already overdetermined by the $|T|$ matched features, the data tends to be degenerated and leads to unsatisfying results. The degeneration is caused by too small baselines along one or multiple of the spatial axes of the camera positions, as well as the temporal axis. As a result of the insufficient information in the input data, patches with erroneous position and velocity are reconstructed.

Under the assumption that sufficient information is present in the candidate set C to find the correct patch, the initialization algorithm enriches the set T, using a greedy approach.

To find more information that is coherent with the current reconstruction more features $f' \in C \setminus T$ need to be added to T. Each feature f' is accepted into T if the patch reconstructed from $T' = T \cup \{f'\}$ has at least T' as consenting feature set.

After the enrichment of T the final set of consenting features is used to calculate the position and velocity for the patch P'. To fully initialize P', two more parameters need to be set. The first is the *reference image* of the patch, which has two different uses. If I is the reference image of P' than the acquisition time $t_r = t(I)$ marks the point when the patch P' is observed at the reconstructed center position $\vec{c}$. As a result the scene position $pos(P', t')$ of P' at any given time t' is:

$$pos(P', t') = \vec{c} + (t' - t_r) \cdot \vec{v}. \tag{2}$$

Furthermore, the reference image is used in visibility calculations, where a normalized cross correlation is used. The correlation template for a patch P' is extracted from its reference image. The reference image for P' is the image the original feature f_0 was taken from. The last parameter for P' is the surface orientation represented by the patch normal. The normal of P' is coarsely approximated by the vector pointing from $\vec{c}$ to the center of the reference image camera. When the patch has been fully initialized, it is added to the initial patch generation.

After all image features have been processed the initial patch generation is optimized and filtered once before the expand and filter iterations start.

5.1. Geometric Patch Reconstruction

Input for the geometric patch reconstruction is a list of corresponding pixel positions in multiple images combined with the temporal and spatial position of the cameras. The result is a patch center $\vec{c}$ and velocity $\vec{v}$.

Assuming a linear movement of the scene point, its position $\vec{x}(t)$ at the time t is specified by a line

$$\vec{x}(t) = \vec{c} + t \cdot \vec{v}. \tag{3}$$

To determine $\vec{c}$ and $\vec{v}$, a linear equation system is formulated. The line of movement (3) must intersect the viewing rays $\vec{q}^i$ that originate from the camera center $\Phi(I^i)$ and are cast through the image plane at the pixel position where the patch was observed in image $t^i = t(I^i)$:

$$\begin{pmatrix} \mathrm{Id}^{3\times3} & \mathrm{Id}^{3\times3}\cdot t^0 & -\vec{q}_0{}^T & 0 & 0 \\ \vdots & \vdots & & \ddots & \\ \mathrm{Id}^{3\times3} & \mathrm{Id}^{3\times3}\cdot t^i & 0 & 0 & -\vec{q}_i{}^T \end{pmatrix} \cdot \begin{pmatrix} \vec{c}^T \\ \vec{v}^T \\ a_0 \\ \vdots \\ a_j \end{pmatrix} = \begin{pmatrix} \Phi(I^0)^T \\ \vdots \\ \Phi(I^i)^T \end{pmatrix} \tag{4}$$

The variables a^0 to a^j give the scene depth in respect to the camera center $\Phi(I^{\lfloor \frac{i}{3} \rfloor})$ and are not further needed. The overdetermined linear system is solved with a SVD solver.

5.2. Patch Visibility Model

There are two sets of visibilities associated with every patch P. The set of images where P might be visible $V(P)$ and the set of images where P is considered truly visible $V^t(P) \subset V(P)$. The two different sets exist to deal with specular highlights or not yet reconstructed occluders.

During the initialization process the visibilities are determined by thresholding a normalized cross correlation. If $\nu(P,I)$ is the normalized cross correlation calculated from the reference image of P to the image I, then $V(P) = \{I|\nu(P,I) > \alpha\}$ and $V^t(P) = \{I|\nu(P,I) > \beta\}$. The threshold parameters used in all our experiments are $\alpha = 0.45$ and $\beta = 0.8$. The correlation function ν takes the patch normal into account when determining the correlation windows.

In order to have a efficient lookup structure for patches later on, we overlay a grid of cells over every image. In every grid cell all patches are listed, that when projected to the image plane, fall into the given cell and are considered possibly or truly visible in the given image.

The size of the grid cells λ and the resulting resolution determines the final resolution of our scene reconstruction as only one truly visible patch in each cell in every image is calculated. We experienced that it is a valid strategy to start with a higher λ (e.g. $\lambda \geq 2$) for an initial quasi-dense reconstruction, followed by a reconstruction at pixel level ($\lambda = 1$).

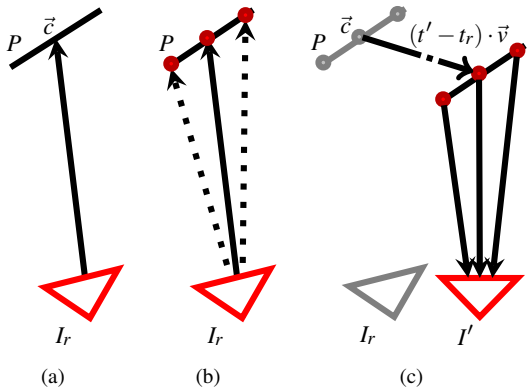


Figure 2: *Computing cross correlation of moving patches. (a) A patch P is described by its position $\vec{c}$, orientation, recording time t_r and its reference image I_r. (b) Positions of sampling points are obtained by casting rays through the image plane (red) of I_r and intersecting with plane P. (c) According to the difference in recording times $(t' - t_r)$ and the motion $\vec{v}$ of the patch, the sampling points are translated, before they are projected back to the image plane of I'. Cross correlation is computed using the obtained coordinates in image space of I'.*

The grid structure is also used to perform the visibility tests during the expand and filter iterations.

The visibility of P is estimated by a depth comparison within the grid cells. All images, for which P is closer to the camera than the currently closest patch in the cell, are added to $V(P)$. The images $I' \in V^t(P')$, where the patch is considered truly visible, are determined using the same method of comparing ν against β as before, except that the threshold is lowered with increasing expansion iteration count to cover poorly textured regions.

6. Expansion phase

The initial set of patches is usually very sparse. To incrementally cover the entire visible surface, the existing patches are expanded along the object surfaces. The expansion algorithm processes each patch from the current generation.

In order to verify if a given patch P should be expanded, all images $I \in V^t(P)$ where P is truly visible are considered. Given the patch P and a single image I, the patch is projected into the image plane and the surrounding grid cells are inspected. If a cell is found where no truly visible patch exists yet, a surface expansion of P to the cell is calculated.

A viewing ray is cast through the center of the empty cell and intersected with the plane defined by the patches position at $t(I)$ and its normal. The intersection point is the center position for the newly created patch P'. The velocity and normal of the new patch are initialized with the values from

the source patch P. At this stage, P' is compared to all other patches listed in its grid cell and is discarded if another similar patch is found. To determine whether two patches are similar in a given image, their position $\vec{x}_0, \vec{x}_1$ and normals $\vec{n}_0, \vec{n}_1$ are used to evaluate the inequality

$$(\vec{x}_0\vec{x}_1)\cdot\vec{n}_0 + (\vec{x}_1\vec{x}_0)\cdot\vec{n}_1 < \kappa. \quad (5)$$

The comparison value κ is calculated from the pixel displacement of λ pixels in image I and corresponds to the depth displacement which can arise within one grid cell. If the inequality holds, the two patches are similar.

Patches that are not discarded are processed further. The reference image of the new patch P' is set to be the image I in which the empty grid cell was found. The visibility of P' is estimated by a depth comparison as described in 5.2. Because the presence of outliers may result in a too conservative estimation of $V(P')$, the visibility information from the original patch is added $V(P') = V(P') \cup V(P)$ before calculating $V^t(P')$.

After the new patch is fully initialized, it is handed into the optimization process. Finally, the new patch is accepted into the current patch generation, if $|V^t(P')| \geq \phi$. The least number of images to accept a patch is dependent on the camera setup and image type. With increasing ϕ less surface can be covered with patches on the outer cameras, since each surface has to be observed multiple times. Choosing ϕ too small may result in unreliable reconstruction results.

7. Patch Optimization

The patch parameters calculated from the initial reconstruction or the expansion are the starting point for a conjugate gradient based optimization. The function ρ maximized is a visibility score of the patch. To determine the visibility score a normalized cross correlation $v(P,I)$ is calculated from the reference image of P to all images $I \in V(P)$ where P is expected to be visible:

$$\rho(P) = \frac{1}{|V(P)| + a\cdot|V^t(P)|}\left(\sum_{I\in V(P)} v(P,I) + \sum_{I\in V^t(P)} a\cdot v(P,I)\right) \quad (6)$$

The weighting factor a accounts for the fact that images from $V^t(P)$ are considered reliable information, while images from $V(P)\setminus V^t(P)$ might not actually show the scene point corresponding to P. The visibility function $\rho(P)$ is then maximized with a conjugate gradient method.

To constrain the optimization, the position of P is not changed in three dimensions, but in a single dimension representing the depth of P in the reference image. The variation of the normal is specified by two rotation angles and at last the velocity is left as three dimensional vector. The resulting problem has six dimensions.

8. Filtering

After the expansion step the set of surface patches possibly contains visual inconsistencies. These inconsistencies can be put in three groups. The outliers outside the surface, outliers that lie inside the actual surface and patches that do not satisfy a regularization criterion. Three distinct filters are used to eliminate the different types of inconsistencies.

The first filter deals with outliers outside the surface. To detect an outlier a support value s and a doubt value d is computed for each patch P. The support is the patch score Eq. (6) multiplied by the number of images where P is truly visible $s = \rho(P)\cdot|V^t(P)|$. Summing the score of all patches P' that are occluded by P gives a measure for visual inconsistency introduced by P and is the doubt d. If the doubt outweighs the support $d > s$ the patch is considered an outlier and removed.

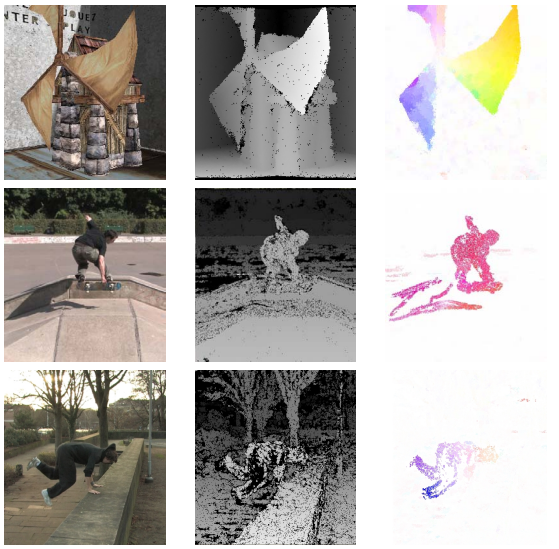
Patches lying inside the surface will be occluded by the patch representing the real surface, therefore the visibilities of all patches are recalculated as described in 5.2. Afterwards, all patches that are not visible in at least ϕ images are discarded as outliers.

The regularization is done with the help of the patch similarity defined in Eq. (5). In the images where a patch P is visible all surrounding c patches are evaluated. The quotient of the number c' of patches similar to P in relation to the total surrounding patches c is the regularization criterion: $\frac{c'}{c} < z$. The quotient of the similarly aligned patches was $z = 0.25$ in all our experiments.

9. Results

To test the capabilities of our algorithm we used the synthetic scene shown in Fig. 3 (top row). The scene is a textured model of a windmill with rotating wings. As input we generated images from six viewpoints at a resolution of 480×270 pixels. The time offset between the six cameras is spread equally over one frame. The grid cell size is set to $\lambda = 2$ for the first 18 iterations and then decreased to $\lambda = 1$ for the next 50 iterations. The total runtime on the test machine a AMD Athlon 64 X2 6400+ was 12 hours. In the resulting depth map Fig. 3b), continuous depth changes as the floor plane or the walls are remarkably smooth while the discontinuities on the wing edges are retained. The small irregularities where no patch was created stem from the conservative filtering step. How well the motion of the wings is reconstructed can be seen in the flow visualization Fig. 3c). The outline of the wings is clearly visible and the motion decreases towards the rotation center.

In addition the synthetic test scene, we used two outdoor sequences. The resolution for both scenes was 960×540 pixels. The skateboarder scene, Fig. 3 (middle row), was filmed with six unsynchronized cameras and chosen because it has a large depth range and fast motion. The skateboarder



(a) (b) (c)

Figure 3: *(a) Input views, (b) quasi-dense depth reconstruction and (c) optical flow to the next frame. For the synthetic windmill scene, high-quality results are obtained. When applied to the more challenging real-world scenes (skateboarder scene, middle, parkours scene, bottom), robust and accurate results are still obtained. The conservative filtering prevents the expansion to ambiguous regions. E.g., most pixels in the asphalt region in the skateboarder scene are not recovered. All moving regions except the untextured body of the parkours runner were densely reconstructed, while some motion outliers remain in the background.*

and the ramp in the foreground as well as the trees in the background are reconstructed in great detail, Fig. 3b). The asphalt area offers very little texture. Due to our restrictive filtering, it is not fully covered with patches. The motion of the skater and that of his shadow moving on the ramp is visible in 3c). The shown results were obtained after 58 iterations starting with $\lambda = 2$ and using $\lambda = 1$ from iteration 55 onward. The total computation time was 95 hours.

The second real world scene Fig. 3 (bottom row) features a setup of 16 cameras showing a parkours runner jumping into a handstand. The scene has a highly cluttered background geometry. Similar to the skateboard scene, regions with low texture are not covered with patches. However, details of the scene are clearly visible in the depth map and the motion reconstructed for the legs and the back of the person is estimated very well. Due to the cluttered geometry and the large number of expansion steps, the reconstruction took 160 hours. For visual assessment of our approach, we would like to refer to our accompanying video.

To demonstrate the static reconstruction capabilities we show the results obtained from the Middlebury "ring"

(a) (b) (c)

Figure 4: *Reconstruction results from the Middlebury MVS evaluation datasets. (a) Input views. (b) Closed meshes from reconstructed patch clouds. (c) Textured patches. While allowing the reconstruction of all six degrees of freedom (including 3D motion), our approach still reconstructs the static geometry faithfully.*

datasets [Mid] in Fig. 4. We used the Poisson surface reconstruction [KBH06] to create the closed meshes. The static object is retrieved, although no prior knowledge about the dynamics of the scene was given, i.e., we used all six degrees of freedom for reconstruction. Computation time for these datasets was 24 hours each.

10. Conclusion

The results produced by our algorithm show promising potential. We successfully reconstructed depth, orientation and motion in several challenging scenes. To stimulate further research, we plan to publish our synthetic data along with ground truth information on-line.

We do not yet use the temporal coherence within a video or a dynamic regularization. Both concepts are expected to further improve the robustness of our approach. Sophisticated regularization techniques could also help to reconstruct texture-less areas, e.g., the asphalt area in the skateboarder sequence.

The conceivable applications reach from free viewpoint applications over markerless motion capture to image segmentation tasks, that can distinguish foreground from background by using depth and velocity cues. One obstacle for most application are the long run-times of our approach. A fully GPU-based implementation might help to reduce this problem significantly.

References

[BBH08] BRADLEY D., BOUBEKEUR T., HEIDRICH W.: Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008* (2008), pp. 1–8.

[BETG08] BAY H., ESS A., TUYTELAARS T., GOOL L. V.: Surf: Speeded up robust features. *Computer Vision and Image Understanding 110*, 3 (2008), 346–359.

[CK02] CARCERONI R., KUTULAKOS K.: Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance. *International Journal of Computer Vision 49*, 2 (2002), 175–214.

[FP09a] FURUKAWA Y., PONCE J.: Accurate, dense, and robust multi-view stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2009).

[FP09b] FURUKAWA Y., PONCE J.: Carved visual hulls for image-based modeling. *International Journal of Computer Vision 81*, 1 (2009), 53–67.

[GSC*07] GOESELE M., SNAVELY N., CURLESS B., HOPPE H., SEITZ S.: Multi-view stereo for community photo collections. In *IEEE International Conference on Computer Vision (ICCV)* (2007).

[HRT*09] HASLER N., ROSENHAHN B., THORMÄHLEN T., WAND M., GALL J., SEIDEL H.-P.: Markerless Motion Capture with Unsynchronized Moving Cameras. In *Proc. of CVPR'09* (Washington, June 2009), IEEE Computer Society, p. to appear.

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (2006), Eurographics Association, p. 70.

[KKBC07] KOLEV K., KLODT M., BROX T., CREMERS D.: Propagated photoconsistency and convexity in variational multiview 3d reconstruction. In *Workshop on photometric analysis for computer vision* (2007).

[MAW*07] MERRELL P., AKBARZADEH A., WANG L., MORDOHAI P., FRAHM J., YANG R., NISTÉR D., POLLEFEYS M.: Real-time visibility-based fusion of depth maps. In *Proceedings of International Conf. on Computer Vision* (2007).

[Mid] MIDDLEBURY MULTI-VIEW STEREO EVALUATION: http://vision.middlebury.edu/mview/.

[MSMP08] MEYER B., STICH T., MAGNOR M., POLLEFEYS M.: Subframe Temporal Alignment of Non-Stationary Cameras. In *Proc. British Machine Vision Conference* (2008).

[NA02] NEUMANN J., ALOIMONOS Y.: Spatio-temporal stereo using multi-resolution subdivision surfaces. *International Journal of Computer Vision 47*, 1 (2002), 181–193.

[PKF05] PONS J., KERIVEN R., FAUGERAS O.: Modelling dynamic scenes by registering multi-view image sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005* (2005), vol. 2.

[PKF07] PONS J., KERIVEN R., FAUGERAS O.: Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision 72*, 2 (2007), 179–193.

[SCD*06] SEITZ S., CURLESS B., DIEBEL J., SCHARSTEIN D., SZELISKI R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2006), vol. 1.

[SSS06] SNAVELY N., SEITZ S., SZELISKI R.: Photo tourism: exploring photo collections in 3D. In *ACM SIGGRAPH 2006 Papers* (2006), ACM, p. 846.

[SZB*07] SORMANN M., ZACH C., BAUER J., KARNER K., BISHOF H.: Watertight multi-view reconstruction based on volumetric graph-cuts. *Image Analysis 4522* (2007).

[VBR*99] VEDULA S., BAKER S., RANDER P., R P., YZ E., COLLINS R., KANADE T.: Three-dimensional scene flow.

[VBR*05] VEDULA S., BAKER S., RANDER P., COLLINS R., KANADE T.: Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2005).

[WSY07] WANG H., SUN M., YANG R.: Space-Time Light Field Rendering. *IEEE Trans. Visualization and Computer Graphics* (2007), 697–710.

[ZCS03] ZHANG L., CURLESS B., SEITZ S.: Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2003), vol. 2.

[ZK01] ZHANG Y., KAMBHAMETTU C.: On 3D scene flow and structure estimation. In *Proc. of CVPR'01* (2001), vol. 2, IEEE Computer Society, pp. 778–785.