A Tensor Voting Approach for Multi-view 3D Scene Flow Estimation and Refinement

Jaesik Park, Tae Hyun Oh, Jiyoung Jung, Yu-Wing Tai, and In So Kweon

Korea Advanced Institute of Science and Technology, Daejeon, South Korea
{jspark,thoo,jyjung}@rcv.kaist.ac.kr, yuwing@cs.kaist.ac.kr,
iskweon@kaist.ac.kr

Abstract. We introduce a framework to estimate and refine 3D scene flow which connects 3D structures of a scene across different frames. In contrast to previous approaches which compute 3D scene flow that connects depth maps from a stereo image sequence or from a depth camera, our approach takes advantage of full 3D reconstruction which computes the 3D scene flow that connects 3D point clouds from multi-view stereo system. Our approach uses a standard multi-view stereo and optical flow algorithm to compute the initial 3D scene flow. A unique two-stage refinement process regularizes the scene flow direction and magnitude sequentially. The scene flow direction is refined by utilizing 3D neighbor smoothness defined by tensor voting. The magnitude of the scene flow is refined by connecting the implicit surfaces across the consecutive 3D point clouds. Our estimated scene flow is temporally consistent. Our approach is efficient, model free, and it is effective in error corrections and outlier rejections. We tested our approach on both synthetic and real-world datasets. Our experimental results show that our approach outperforms previous algorithms quantitatively on synthetic dataset, and it improves the reconstructed 3D model from the refined 3D point cloud in real-world dataset.

Keywords: Scene Flow, Tensor Voting, Multi-view.

1 Introduction

3D scene flow [1] is a dense 3D motion vector field which describes the non-rigid motion of objects in 3D world. Over the last decade, various approaches [1–11] have been proposed for 3D scene flow estimation. Among the proposed approaches, most of them [2, 3, 5, 6, 8–11] were developed based on stereoscopic inputs such that the estimated 3D scene flow is defined up to the depth map ambiguities [11] where scene flow in occluded areas was undetermined. Also, since depth map provides incomplete information about the 3D world, the estimated scene flow can be easily biased. In this work, we introduce a 3D scene flow algorithm which utilizes the 3D point cloud from multi-view stereo. Our estimated scene flow, thus, describes the full 3D motion vector field of objects' motions in 3D world.

Our approach is built on top of recent advances in 3D multi-view reconstruction and in 2D optical flow estimation. In 3D multi-view reconstruction, recent works such as [12] can reconstruct accurate 3D model with metric error less than

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part IV, LNCS 7575, pp. 288–302, 2012.

1mm in the standard middlebury multi-view stereo dataset [13]. Their techniques are well suited for illumination changes, occlusion and low-textured regions. In optical flow estimation, recent algorithms such as [14] have demonstrated less than 1 pixel average end-point error and less than 5 degree average angular error in the standard middlebury optical flow dataset [15]. However, when combining them together for the scene flow estimation through back projecting the estimated optical flow onto the estimated 3D model [1], the accuracy of the estimated 3D scene flow is far from satisfactory. This is because such a naive approach can easily amplify errors from both the estimated 3D model and the estimated optical flow. In addition, since optical flow is estimated individually in 2D image domain, errors caused by aperture problem, occlusion/disocclusion from certain view point can be easily propagated to the estimated 3D scene flow. There are various approaches [16–18, 4] which jointly estimate the 3D model together with scene flow through mesh deformation and/or 3D points tracking. However, such approaches usually involve complicated data structures to handle deformed meshes or they require rich textures on the reconstructed surface for accurate and dense 3D points tracking. In contrast, our approach computes the scene flow across 3D point clouds which is model free, simple in data representation, and it is computationally efficient.

Our approach starts with an initial scene flow computed by back projecting 2D optical flow from different view points onto the 3D point cloud [1]. Inspired by the work from Wu *et al.* [19] which uses the closed form tensor voting (CFTV) to reject outliers and to estimate surface normals without explicit computation of surface mesh, we modify the CFTV to handle scene flow data. Since CFTV only provides us the scene flow direction, but not the magnitude of the scene flow, we utilize implicit surface representation to estimate the scene flow magnitude by connecting the scene flow of the 3D point cloud at time t with the implicit surface of the 3D point cloud at time $t+1$. One major advantage of our approach is that it can effectively reduce computation and number of parameters by converting the direction and magnitude estimation of the scene flow into a two-stage process that is optimized sequentially. By adopting the CFTV framework in scene flow direction estimation, our scene flow refinement algorithm is structure-aware and we gain the advantage on outlier rejections and error corrections of the 3D point cloud. Our experimental results show that our approach is effective in improving the accuracy of the estimated scene flow from its initialization. Also, our scene flow refinement algorithm is ready to be adopted to other scene flow estimation algorithms as post-processing to further enhance their performances.

2 Related Work

Since the concept of scene flow estimation introduced by Vedula *et al.* [1], various scene flow estimation algorithms have been introduced. In [1], Vedula *et al.* first introduced a scene flow estimation algorithm which projects optical flow onto a known 3D model with photometric consistent constraint from each of the input images to regularize the estimated scene flow. There are other studies on scene

flow estimation in multi-view camera setup as well, such as [7, 20]. Zhang *et al.* [7] combine 2D motion from optical flow and stereo view constraint in scene flow estimation. Pons *et al.* [20] estimate scene flow after scene reconstruction by matching images at time t and images at time $t+1$ to refine scene flow over multiple cameras. Wedel *et al.* [6] take stereo image sequences as input. They use stereo matching algorithm to estimate depth map and computes the scene flow across the estimated depth map.

Other approaches use joint estimation framework [3, 5, 9]. These approaches estimate depth map and scene flow simultaneously under different camera configurations. For instance, Huguet *et al.* [3] couples optical flow estimations for each camera with dense stereo matching. Rabe *et al.* [5] present a real time implementation of a variational optical flow algorithm with Kalman filter for temporal smoothness. Basha *et al.* [2] estimates 3D structure and scene flow simultaneously in a unified variational framework. Their approach is based on 3D parameterizations of depth map and 3D scene flow. Vogel *et al.* [8] improves the work from Basha *et al.* [2] by introducing a rigid motion prior into optimization function for scene flow estimation. Wedel *et al.* [11] compute 3D scene flow from hand held video cameras through stereoscopic analysis of video depth to estimate 3D motions of camera and scene flow. Recently, Hadfield *et al.*[10] introduce scene flow particle filter which operate directly on the depth map captured by Xbox Kinect.

Besides optimization based methods, there are methods which rely on deformation and 3D points tracking. In [17], Devernay *et al.* tracks the poses of surfels [21] in video sequence to estimate scene flow. They represent the surfel as a small planar square region. Through analyzing the deformation of surfel in each video frame, they can estimate the translation and rotation parameters of surfel across different frames and hence obtain the scene flow in terms of surfel deformation parameters. Furukawa *et al.* [18] start with a 3D mesh representation, and track the projected motion trajectory of vertex coordinates of the 3D mesh in a scene through tracking the optical flow of feature points. Since 3D topology is known, the scene flow is obtained through computing the deformed mesh which agrees with the projected optical flow.

There are several approaches in scene flow estimation employing various 3D representations that should also be mentioned. Carceroni and Kutulakos [22] represent scene flow as dynamic surfel which encodes the instantaneous local shape, reflectance, and motion of a small region in the scene under known illumination conditions. Wand *et al.* [23] operate on point cloud inputs, but also infer the topology of the point cloud. Courchay *et al.* [24] use animated mesh to simultaneously estimate 3D shape and 3D motion of a dynamic scene from multiple-viewpoint calibrated videos.

Comparing our approach with previous works, our approach also uses regularization to estimate and to refine the scene flow. A major difference between our approach and the previous approaches is that we process our algorithm on the 3D point cloud while many of the previous algorithms are processed on the depth map in 2D image plane. Comparing 3D point clouds with depth maps, 3D point

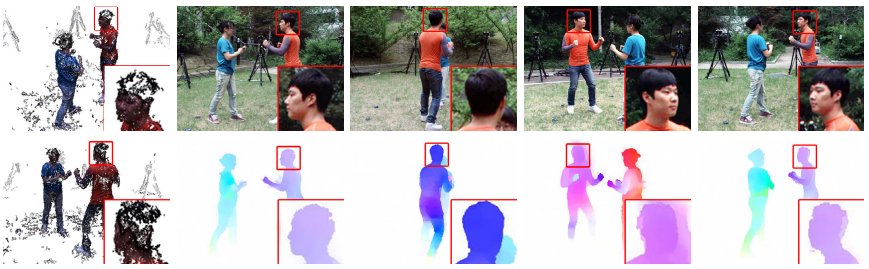


Fig. 1. Initial inputs of our approach. Our data is captured by 20 synchronized cameras which were arranged uniformly around the target objects in an outdoor environment. (a) The 3D point cloud reconstructed by using PMVS [12]. (b) to (e) The estimated optical flow from different cameras by using method in [14]. Note the errors of the 3D point cloud and optical flow around the head area caused by color ambiguity.

clouds are unstructured, and its sampling can be highly non-uniform. These factors cause additional challenges in computing scene flow for 3D point clouds. In addition, neighborhood regularization in 3D point clouds without mesh reconstruction is not as trivial as the neighborhood regularization for scene flow working on depth map where the spatial neighborhood is well defined and well sampled in 2D image domain. For this reason, we adopt the CFTV [19] which defines neighborhood in 3D not only based on the Euclidean distance, but also based on the relative location and normal orientations of 3D points. We choose to work on an unstructured 3D point cloud since this is a model free representation, and it provides the maximum freedom of deformation for scene flow estimation comparing to previous works using surfel/mesh representation. Also, data structure for an unstructured 3D point cloud is simple, and hence it is computationally efficient.

3 Our Approach

3.1 Data Acquisition

Our system for data acquisition consists of 20 cameras uniformly distributed around the target objects. These cameras are calibrated and synchronized with image resolution 1600×1200. Instead of putting our system in a well controlled indoor environment, we capture our data in outdoor environment with uncontrolled lighting and complex background to demonstrate a more general application of our approach. Figure 1 shows the setting and initial inputs of our approach.

3.2 Initial Scene Flow Estimation

Our approach starts with an initial estimation of the scene flow from multi-view stereo and optical flow triangulation. For our convenience, we use the source code from the patch based multi-view stereo (PMVS)[12] to obtain our initial

3D point cloud for each frame individually. We also use the source code provided by Sun *et al.* [14] to compute the optical flow for every image sequence from each camera individually.

We denote by $\mathbf{x} = \{u, v\}$ the image plane coordinates, $\mathbf{X} = \{x, y, z\}$ the 3D world coordinates, $\mathbf{f} = \{f_u, f_v\}$ the optical flow in image plane, $\mathbf{F} = \{F_x, F_y, F_z\}$ the scene flow in 3D world, and $\mathbf{P}$ the 3×4 camera projection matrix. We use subscript index $i \in [1, M]$ to represent the i-th point in the 3D point cloud, subscript index $j \in [1, N]$ to represent the j-th camera in the system setting, and superscript index $t \in [1, T]$ to represent time (the t-th frame) in input image sequences where M is total number of points in the 3D point cloud, $N = 20$ is the total number of cameras, and T is the total number of frames respectively. Hence, $\mathbf{x}_{ij}^t = \mathbf{P}_j \mathbf{X}_i^t$, and $\mathbf{f}_{ij}^t = \mathbf{P}_j \mathbf{F}_i^t$.

In order to obtain the initial scene flow from the 3D point cloud and the 2D optical flow, we utilize the approach from Ruttle *et al.* [16] which minimizes the least square reprojection errors of the scene flow projected onto the image plane from each camera. This is achieved by solving $A_i(\mathbf{X}_i + \mathbf{F}_i) = 0$ for each point in the 3D point cloud individually where A_i is equal to:

$$A_i = \begin{bmatrix} (u_{i1} + f_{u,i1})\mathbf{P}_{3,1} - \mathbf{P}_{1,1} \\ (v_{i1} + f_{v,i1})\mathbf{P}_{3,1} - \mathbf{P}_{2,1} \\ \vdots \\ (u_{iN} + f_{u,iN})\mathbf{P}_{3,N} - \mathbf{P}_{1,N} \\ (v_{iN} + f_{v,iN})\mathbf{P}_{3,N} - \mathbf{P}_{2,N} \end{bmatrix}, \tag{1}$$

where $\{u_{ij}, v_{ij}\}$ is the image coordinate of the i-th point projected on the j-th camera, and $\{f_{u,ij}, f_{v,ij}\}$ is the 2D optical flow of the i-th point on the image plane of the j-th camera. $\mathbf{P}_{1,j}$, $\mathbf{P}_{2,j}$, and $\mathbf{P}_{3,j}$ are the first, the second and the third rows of the j-th camera projection matrix $\mathbf{P}_j$ respectively.

3.3 Closed Form Tensor Voting

Since our approach for scene flow refinement is built on top of the closed form tensor voting (CFTV) framework[25, 19], we briefly review it here. CFTV improves the reconstruction accuracy of Furukawa *et al.*[12] by refining noisy surface normal vectors. The tensor voting field considers normal orientation and minimum curvature connections between local neighborhood in 3D to define the neighborhood connectivity prior.

For each point $\mathbf{X}_i$ in the 3D point cloud, a vote is received from each of its neighborhood points $\mathbf{X}_{\tilde{i}}$ as tensor $\mathbf{T}_{i\tilde{i}}$:

$$\mathbf{T}_{i\tilde{i}} = c_{i\tilde{i}} R_{i\tilde{i}} K_{\tilde{i}} R'_{i\tilde{i}} \tag{2}$$

where $R_{i\tilde{i}} = \mathbf{I} - 2\mathbf{r}_{i\tilde{i}}\mathbf{r}_{i\tilde{i}}^{\mathsf{T}}$, $R'_{i\tilde{i}} = (\mathbf{I} - \frac{1}{2}\mathbf{r}_{i\tilde{i}}\mathbf{r}_{i\tilde{i}}^{\mathsf{T}})R_{i\tilde{i}}$, I is 3×3 identity matrix, $\mathbf{r}_{i\tilde{i}}$ is a unit vector whose direction is the same as $\mathbf{X}_{\tilde{i}} - \mathbf{X}_i$, $K_{\tilde{i}} = \mathbf{n}_{\tilde{i}}\mathbf{n}_{\tilde{i}}^{\mathsf{T}}$ is a kernel function which encodes the most likely normal direction, $\mathbf{n}_i$, of $\mathbf{X}_i$ casted by $\mathbf{X}_{\tilde{i}}$ defined by the osculating arc connecting $\mathbf{X}_i$ and $\mathbf{X}_{\tilde{i}}$, and $c_{i\tilde{i}} = \exp(-\frac{\|\mathbf{X}_i - \mathbf{X}_{\tilde{i}}\|^2}{\sigma_d})$

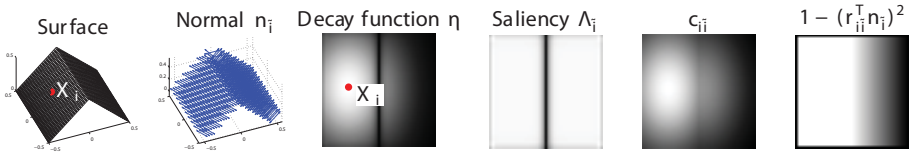


Fig. 2. An example to illustrate the effects of each component (titled with its variable names) in the decay function η in Equation (4). Our decay function combines the surface saliency, Euclidean distance, and normal orientation to evaluate the influence of neighborhood to $\mathbf{X}_i$. A larger weight is given to a point with larger surface saliency, with closer distance, and with better agreement on normal orientations.

is a decay function controlled by a parameter σ_d which penalizes neighborhood points that are further away from $\mathbf{X}_i$.

After collecting votes from neighborhood, the refined normal direction is obtained by using eigen-decomposition to get the eigenvector with the largest eigenvalue. Note that CFTV does not need to reconstruct mesh model in order to estimate and refine normal direction, and this is a major benefit which fits to our problem for scene flow refinement with an unstructured 3D point cloud. In addition, through analyzing the eigenvalue of tensor sum, $\sum_{\tilde{i}} \mathbf{T}_{i\tilde{i}}$, CFTV rejects outliers which received very limited amount of votes from its neighborhood. It can also be used to correct 3D point location by searching a position along the normal direction of a 3D point which receives the maximum amount of vote from its neighborhood. In our implementation, we adopt CFTV as a second step to improve the accuracy of the 3D point cloud from PMVS before scene flow estimation and refinement.

3.4 Scene Flow Refinement

Scene Flow Direction Refinement. We extend the CFTV framework to handle our scene flow data. The basic idea is to define a "voting field" of scene flow which collects the most likely scene flow direction from neighborhood such that structures of 3D points are considered without explicit mesh reconstruction in order to avoid complicated data structure and to make the scene flow refinement process efficient. Also, by incorporating the scene flow and the normal direction propagation together in CFTV, we can further enhance the outlier rejection and error correction ability of CFTV.

We define our scene flow tensor as follow:

$$\mathbf{S}_{i\tilde{i}} = \eta(\mathbf{X}_i, \mathbf{X}_{\tilde{i}}, \mathbf{n}_i, \mathbf{n}_{\tilde{i}}) \overrightarrow{\mathbf{F}}_{\tilde{i}} \overrightarrow{\mathbf{F}}_{\tilde{i}}^{\mathsf{T}}, \tag{3}$$

where $\overrightarrow{\mathbf{F}}_{\tilde{i}}$ is the normalized scene flow direction of $\mathbf{X}_{\tilde{i}}$ and $\eta(\mathbf{X}_i, \mathbf{X}_{\tilde{i}}, \mathbf{n}_i, \mathbf{n}_{\tilde{i}})$ is a structure aware decay function defined as:

$$\eta(\mathbf{X}_i, \mathbf{X}_{\tilde{i}}, \mathbf{n}_i, \mathbf{n}_{\tilde{i}}) = c_{i\tilde{i}} \left[\Lambda_i (1 - (\mathbf{r}_{\tilde{i}i}^{\mathsf{T}} \mathbf{n}_i)^2) + \Lambda_{\tilde{i}} (1 - (\mathbf{r}_{i\tilde{i}}^{\mathsf{T}} \mathbf{n}_{\tilde{i}})^2) \right], \tag{4}$$

where $\Lambda_{\tilde{i}} = \lambda_{1,\tilde{i}} - \lambda_{2,\tilde{i}}$ and $\Lambda_i = \lambda_{1,i} - \lambda_{2,i}$ are the surface saliency of $\mathbf{X}_{\tilde{i}}$ and $\mathbf{X}_i$ respectively, $\lambda_{1,i}$ and $\lambda_{2,i}$ are the first and second largest eigenvalues of structure

tensor sum $\sum_j \mathbf{T}_{ij}$ in Equation (2), and $1-(\mathbf{r}_{\tilde{i}i}^{\mathsf{T}}\mathbf{n}_i)^2$ and $1-(\mathbf{r}_{i\tilde{i}}^{\mathsf{T}}\mathbf{n}_{\tilde{i}})^2$ measures how likely $\mathbf{X}_i$ and $\mathbf{X}_{\tilde{i}}$ are connected to each other given the configurations of normal directions $\mathbf{n}_i$ and $\mathbf{n}_{\tilde{i}}$ respectively [25]. Hence, if two points $\mathbf{X}_i$ and $\mathbf{X}_{\tilde{i}}$ are physically close to each other but the normal configurations of the two points do not match with each other, $\eta(\mathbf{X}_i, \mathbf{X}_{\tilde{i}}, \mathbf{n}_i, \mathbf{n}_{\tilde{i}})$ returns a small weight in propagation since $\mathbf{X}_i$ and $\mathbf{X}_{\tilde{i}}$ are unlikely to connect with each other on the same surface. On the other hand, we give a large weight to $\mathbf{X}_i$ and $\mathbf{X}_{\tilde{i}}$ if they are physically closed and the normal configurations agree with each other. Figure 2 illustrates the effects of $\eta(\mathbf{X}_i, \mathbf{X}_{\tilde{i}}, \mathbf{n}_i, \mathbf{n}_{\tilde{i}})$ and each individual component of $\eta(\mathbf{X}_i, \mathbf{X}_{\tilde{i}}, \mathbf{n}_i, \mathbf{n}_{\tilde{i}})$ where $\mathbf{X}_i$ receives a larger weight from a point that lies on the same surface with the same normal direction and smaller weight from a point that lie on another surface with different normal directions.

After we collect the second order moment of the scene flow from neighborhood, we can obtain the scene flow direction through eigen-decomposition to find the direction of eigenvector with the largest eigenvalue. The sign ambiguity is resolved by choosing the sign that provides smaller angular difference between the refined scene flow direction and the initial scene flow direction. The scene flow CFTV can estimate the scene flow direction, but it does not provide the magnitude of scene flow since magnitude of scene flow is not encoded in the second order moment. Although we can also propagate the scene flow magnitude individually similar to the scene flow direction propagation, the estimated scene flow magnitude might not be physically correct. For instance, it does not guarantee that the estimated scene flow will touch the surface of the 3D model in the next frame.

Scene Flow Magnitude Refinement. With the estimated scene flow direction, $\overrightarrow{\mathbf{F}}$, we estimate the scene flow magnitude, $\mathbf{m}$, by exploiting the physical property of the scene flow which connects the surface of the 3D model of the two consecutive frames. The energy function we want to minimize is defined as follow:

$$E(m_i)=\frac{1}{\mathbf{X}_{\sigma,i}^2}\parallel \mathbf{X}_{\mu,i}-(\mathbf{X}_i+m_i\overrightarrow{\mathbf{F}}_i)\parallel^2+\kappa\sum_{\tilde{i}}\eta(\mathbf{X}_i,\mathbf{X}_{\tilde{i}},\mathbf{n}_i,\mathbf{n}_{\tilde{i}})\parallel m_i-m_{\tilde{i}}\parallel^2 \quad (5)$$

where $\mathbf{X}_{\mu,i}$ is the predicted 3D point location of $\mathbf{X}_i$ at time $t+1$ evaluated by the surface saliency of the 3D point cloud at time $t+1$ along the direction $\overrightarrow{\mathbf{F}}_i$ connecting $\mathbf{X}_i$, $\mathbf{X}_{\sigma,i}^2$ is the variance of surface saliency, $\kappa=0.5$ is the parameter balancing the data term in the first half of Equation (5) and the smoothness term in the second half of Equation (5).

$\mathbf{X}_{\mu,i}$ and $\mathbf{X}_{\sigma,i}^2$ can be estimated by sampling several (7 in our implementation) discrete locations along $\overrightarrow{\mathbf{F}}_i$ to collect votes and to evaluate the variation of surface saliency. A Gaussian distribution is fitted to the evaluated surface saliency and hence $\mathbf{X}_{\mu,i}$ and $\mathbf{X}_{\sigma,i}^2$ correspond to the mean and variance of the fitted Gaussian distribution. Note that the estimation of $\mathbf{X}_{\mu,i}$ and $\mathbf{X}_{\sigma,i}^2$ can be computed individually for each $\mathbf{X}_i$ without explicit surface reconstruction. Equation (5) is then optimized after initial estimation of the scene flow magnitude of all $\mathbf{X}_i$.

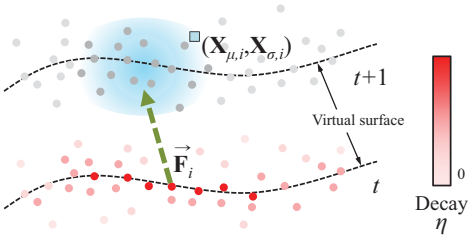


Fig. 3. We estimate the magnitude of the scene flow by utilizing the physical property of the scene flow that connects the 3D structures of the consecutive frames. Since our approach is mesh free, we compute a "virtual surface" by evaluating the variation of the surface saliency by collecting tensor votes along the scene flow direction. The optimal scene flow magnitude is the one which touches the virtual surface at $t+1$ with structure aware neighbor smoothness regularization defined by Equation (5).

Note that we use the same weighting function $\eta(\mathbf{X}_i, \mathbf{X}_{\tilde{i}}, \mathbf{n}_i, \mathbf{n}_{\tilde{i}})$ in Equation (3) to define the neighborhood smoothness term which embeds the normal configuration of $\mathbf{X}_i$ and $\mathbf{X}_{\tilde{i}}$ and the neighborhood smoothness prior defined by CFTV into account. Figure 3 illustrates this process of finding the scene flow magnitude that connects the virtual surface of two point clouds at time t and $t+1$. We can also further reject outliers by evaluating the scene flow saliency collected by Equation (3).

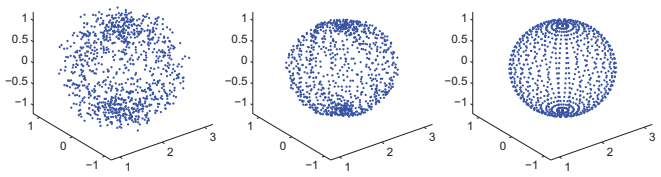
3.5 Temporally Consistent 3D Scene Flow

The above process refines the scene flow of 3D point cloud for each frame individually. In order to fully utilize the correlation between consecutive frames, and to improve the temporal coherency of the estimated scene flow in a long range, we redefine the set of nearest neighbor along the temporal domain for frame $t-1$ and $t+1$. For temporal neighborhood, we can again propagate the scene flow direction, but now we need to modify the voting field such that the voting direction is along the tangential direction of osculating arc defined by the scene flow direction of 3D points at time $t-1$ and $t+1$. Our modified scene flow tensor for temporal neighborhood is defined as follow:

$$\mathcal{S}_{i\tilde{i}^{t+1}} = c_{i\tilde{i}^{t+1}} \mathcal{R}_{i\tilde{i}^{t+1}} \mathcal{K}_{\tilde{i}^{t+1}} \mathcal{R}'_{i\tilde{i}^{t+1}}, \tag{6}$$

where $\tilde{i}^{t+1}$ is the index of 3D point neighbor at time $t+1$, $\mathcal{R}_{i\tilde{i}} = 2\mathbf{r}_{i\tilde{i}}\mathbf{r}_{i\tilde{i}}^{\mathsf{T}}$, $\mathcal{R}'_{i\tilde{i}} = (\frac{1}{2}\mathbf{r}_{i\tilde{i}}\mathbf{r}_{i\tilde{i}}^{\mathsf{T}})\mathcal{R}_{i\tilde{i}}$, and $\mathcal{K}_{\tilde{i}} = \mathbf{F}_{\tilde{i}}\mathbf{F}_{\tilde{i}}^{\mathsf{T}}$ is the kernel function that encodes the most likely scene flow direction, $\mathbf{F}_i^t$, of $\mathbf{X}_i^t$ casted by $\mathbf{X}_{\tilde{i}}^{t+1}$. The scene flow tensor for $\mathcal{S}_{i\tilde{i}^{t-1}}$ is defined similarly. The decay function $c_{i\tilde{i}^{t+1}}$ is same as in Equation (2).

After including temporal neighborhood and the scene flow direction propagation from temporal neighborhood, our estimated scene flow is temporally consistent. At the same time, the scene flow structures/discontinuities can be well preserved in the structure aware scene flow propagation by Equation (3) and Equation (5). In our implementation, we alternate the refinement processes



(a) (b) (c)

Fig. 4. *Sphere* dataset. The 3D points on the sphere are translated with (a) initial noisy scene flow, (b) our refined scene flow and (c) the ground truth scene flow.

Table 1. Mean square error of Figure 4. Our approach achieves the lowest MSE after magnitude regularization.

	MSE
Noisy 3D scene scene flow	0.239
Ours without magnitude regularization	0.130
Ours with magnitude regularization	**0.077**

described in Section 3.4 and Section 3.5 iteratively. In real world dataset, we observe two iterations are sufficient to converge. From our experiments, we found that including temporal neighborhood from time $t+1$ and $t-1$ is sufficient to guarantee temporal coherence without over smoothing the scene flow structures.

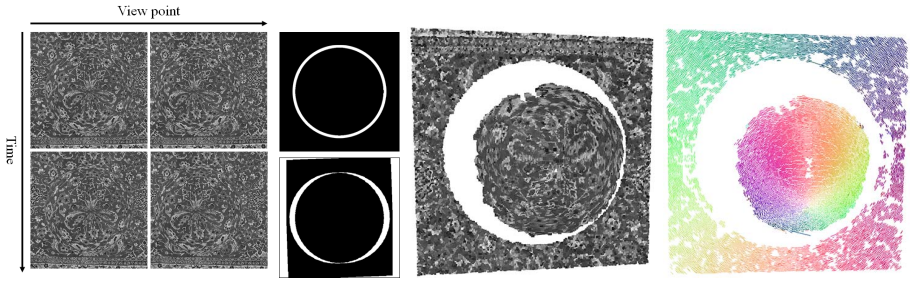
4 Experimental Results

Our first experiments consists of a synthetic dataset, *sphere*, as shown in Figure 4. The radius of the sphere is 1, and the ground truth scene flow is $[1, 0, 0]^\mathsf{T}$ uniformly moving in x-direction. To simulate a noisy scene flow from optical flow projection, we add uniform noise with range [-0.25,0.25] in random directions, but the ground truth locations of 3D points are unaltered. We apply our approach to refine the noisy scene flow. As shown in Figure 4(a) and (b), our approach can successfully refine the noisy scene flow. After shifting the 3D points according to our refined scene flow, the shape of sphere is still preserved. Table 1 shows quantitative evaluation on the mean square error of scene flow comparing to ground truth. It shows that both the direction refinements by scene flow CFTV, and the magnitude refinement by scene flow magnitude regularization are effective.

We compare our results with the results of Huguet *et al.* [3] and Basha *et al.* [2] using *ball* dataset in [2] as shown in Figure 5, which is publicly available with ground truth. This dataset consists of 10 rendered images which represent 5 different view points at time t and $t+1$. The textured sphere and background plane are rotated differently, and the mask for the occluded region and the discontinuity region are provided. In this example, we have also provided an additional baseline comparison to illustrate the relative performance between ours and [18]. Approach in [18] is a mesh deformation approach which refines

Table 2. Quantitative evaluation on the synthetic *ball* dataset. Our results are compared with results from Huguet *et al.* [3] and Basha *et al.* [2]. Ours(Baseline) shows initially estimated scene flow results without any refinement. From left to right are the evaluation metrics for the NRMS errors of the estimated 3D point location ($NRMS_{\mathbf{X}}(\%)$), the NRMS errors of the estimated scene flow ($NRMS_{\mathbf{F}}(\%)$), and the AEE of the estimated scene flow ($AAE_{\mathbf{F}}(deg)$). For each row of each compared methods, the first row and the second row shows the results where the errors in the discontinuities mask (w/o Discontinuities) and occlusion mask (w/o Occlusions) were excluded, the last row shows the results where all pixels are included (All pixels) for evaluation. Best quantitative results were underlined.

Method	Measurement	$NRMS_{\mathbf{X}}(\%)$	$NRMS_{\mathbf{F}}(\%)$	$AAE_{\mathbf{F}}(deg)$
Huguet *et al.* [3]	w/o Discontinuities	9.82	15.96	7.17
	w/o Occlusions	1.19	11.04	6.66
	All pixels	10.43	19.09	9.20
Basha *et al.* [2]	w/o Discontinuities	0.65	2.94	1.32
	w/o Occlusions	1.99	5.63	2.09
	All pixels	4.39	9.71	3.39
Ours (Baseline)	w/o Discontinuities	0.25	6.43	4.74
	w/o Occlusions	0.26	6.99	4.98
	All pixels	1.12	7.89	5.28
Ours (After Refinement)	w/o Discontinuities	0.23	4.88	2.73
	w/o Occlusions	0.24	5.07	2.72
	All pixels	0.57	5.42	2.83



(a) (b) (c) (d)

Fig. 5. Synthetic *ball* dataset from [2]. (a) The synthetic rendered images for inputs. (b) Discontinuity map and occlusion map. (c) Our estimated 3D structure. (d) Our estimated scene flow (color coded for scene flow direction).

the initial mesh from [12] and scene flow through matching the correspondents between time t and $t+1$. In our baseline comparison, we also start with initial 3D point location from [12]. We skip the CFTV refinement of location so that our 3D point location is more closed to the input of [12]. In scene flow refinement, instead of using implicit surface, we found the correspondent at $t+1$ that is closest to the estimated scene flow end-point to get the refined scene flow. We also skip the temporal consistent scene flow refinement described in Section 3.5. We use the evaluation metrics in [2] to measure the quality of our results in term

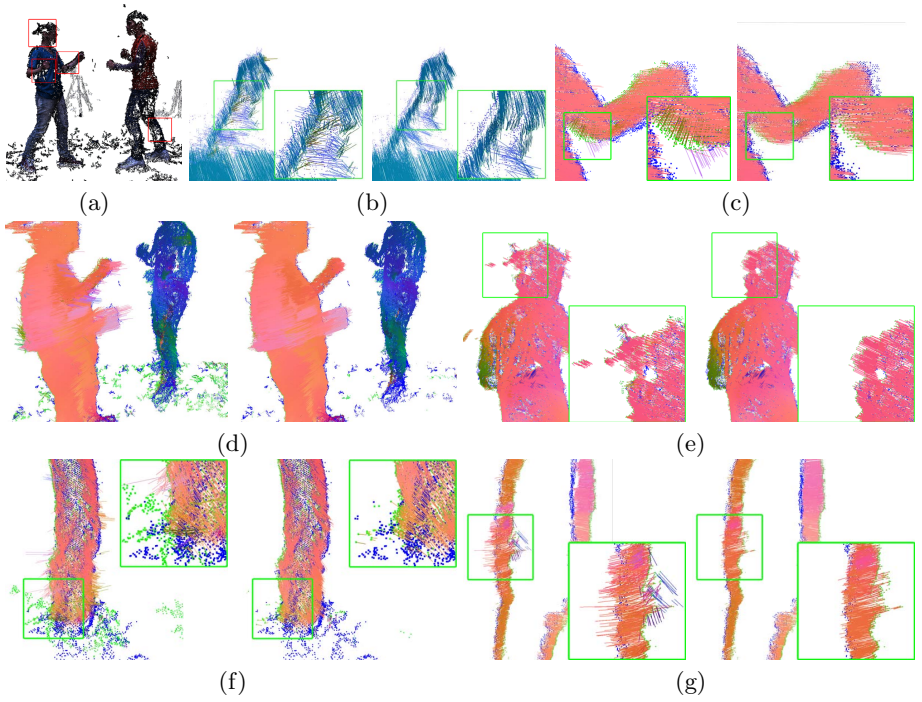


Fig. 6. Scene flow refinement results on *outdoor* dataset. (a) The reconstructed 3D structure at t. For better visualization and evaluation, we zoom-in the scene flow for different body parts: (b) arm part viewed from above, (c) elbow part, (d) back part, (e) head part, (f) ankle part and (g) vertical cross section of human's torso are presented. The scene flow before and after the refinement are shown for comparison.

of the normalized root mean square (NRMS) error for the estimated 3D point locations and for the estimated scene flow, and the absolute angular error (AAE) for the estimated scene flow. Table 2 shows the quantitative comparisons.

Our baseline results in Table 2 is worse than the result of [2]. Comparing the results of ours after refinement with those of our baseline, it is shown that our approach works reasonably and keeps 3D point positions accurately. In addition, the positions of 3D points are further improved by the outlier rejection step of our approach. While the results from [2] show a better performance when the pixels of discontinuity regions are excluded, our approach shows a better performance when all pixels are evaluated due to the usage of CFTV that can handle outliers and discontinuities implicitly in the framework. The result of [2] can be biased due to smoothness assumption on the occlusion region. Also, note that our approach is designed for scene flow refinement for general 3D point clouds, not limited to stereoscopic inputs while both Huguet *et al.* [3] and Basha *et al.* [2] are designed to handle stereoscopic data.

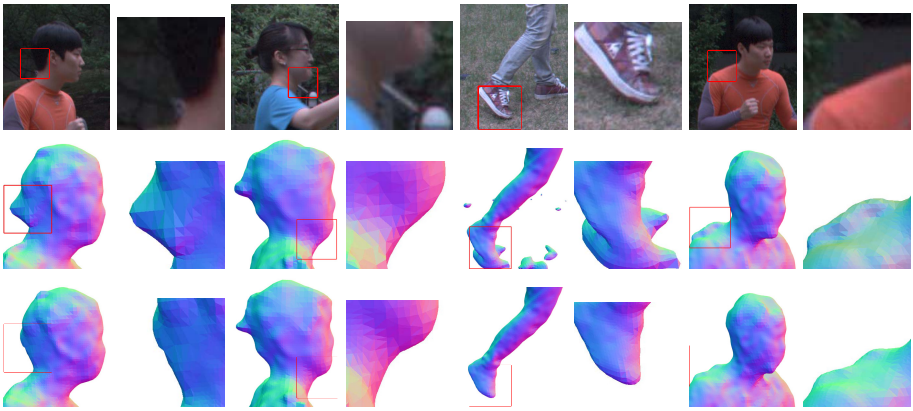


Fig. 7. Our approach improves the 3D reconstruction accuracy. First row: reference input image. Second row: reconstructed 3D model using the 3D point cloud from PMVS [12]. Third row: reconstructed 3D model using our refined 3D point locations of the 3D point cloud.

We evaluate our approach qualitatively on real-world outdoor dataset shown in Figure 1. This dataset consists of around 350,000 points on average per frame. We run our algorithm on a machine with Intel(R) Core (TM) i7 3Ghz PC with 8Ghz RAM memory. The PMVS [12] takes around 2.5 minutes to reconstruct the 3D point cloud per frame, and the optical flow implementation from [14] takes around 15 seconds to compute the optical flow per each image per frame. Our C++ implementation with approximate nearest neighbor data structure [26] (it takes around 5 minutes for initialization and building the ANN data structure) takes around 3 minutes for the direction refinement and around 5 seconds for the magnitude refinement for all points per frame. We use 10 neighbor points in implementation. For each frame of filtered 210,000 points, our un–optimized C++ implementation takes about 250 seconds from Section (3.5) to Section (3.3). Therefore, overall time for temporal consistency takes around 500 seconds.

For better visualization of our results, we show the zoomed-in regions on several parts of the human body in Figure 1. We show the scene flow before and after refinement for comparisons. Our approach is successful in improving the accuracy of scene flow via scene flow direction refinement and scene flow magnitude refinement. Due to the usage of CFTV, our approach can reject outliers and correct locations of 3D points. To visualize the effect of outliers rejection, we show the reconstructed 3D model using Poission surface reconstruction [27] in Figure 7. We have also translated the 3D points at time t to time $t+1$ according to the estimated scene flow in Figure 8. As we can observe in Figure 8, the translated 3D points at t (orange points) match with the 3D points at $t+1$ (cyan points).

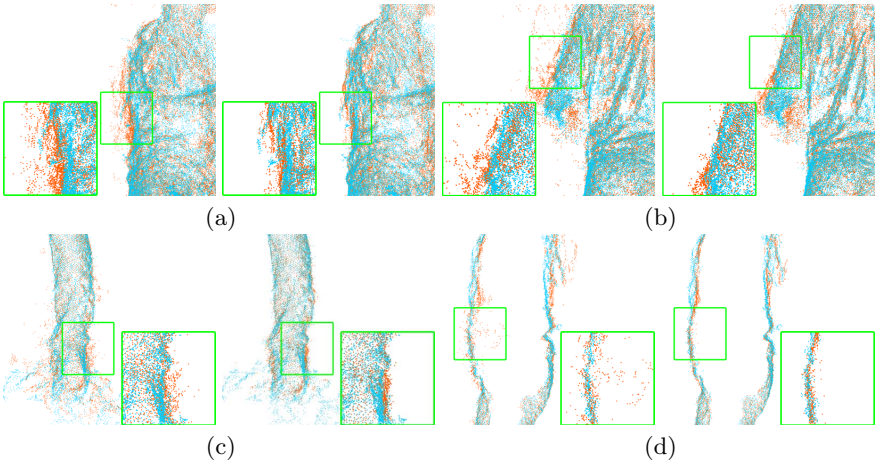


Fig. 8. We translate the point cloud at t (orange points) to match with the point cloud at $t+1$ (cyan points) according to the estimated scene flow. We show the zoom-in regions for: (a) man's back (b) arm part (c) ankle part. (d) Vertical cross section of human's torso. As we can see from the figure, the orange points align well with the cyan points which shows that our estimated scene flow is accurate.

5 Conclusion

In this paper, we have presented a framework to refine the scene flow estimated by multi-view stereo and optical flow back projection from images to 3D points. Our approach extends the CFTV framework to handle scene flow data in addition to the normal estimation problem tackled by the original CFTV framework. To estimate the scene flow magnitude, we exploit the physical property of the scene flow to connect the implicit surface of 3D point clouds in the consecutive frames. We have also introduced the scene flow temporal neighborhood and described how to propagate the scene flow direction from temporal neighborhood. Our approach is model free, and it is robust to handle outliers and noisy scene flow. As part of our future work, we shall study how to include other motion priors, such as rigid motion prior [8], to further enhance the performance of our framework. We will also study how to use the estimated scene flow data for different video editing, segmentation, view synthesis and recognition tasks.

Acknowledgement. This research was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No.2012-0000986 and No. 2012-0003359) and partially by Microsoft Research Asia under KAIST-Microsoft Research Collaboration Center(KMCC). We thank ETRI for capturing the multi-view dataset.

References

1. Vedula, S., Baker, S., Rander, P., Collins, R., Kanade, T.: Three-dimensional scene flow. IEEE Trans. on PAMI 27(3), 475–480 (2005)
2. Basha, T., Aviv, T., Moses, Y., Kiryati, N.: Multi-view scene flow estimation: A view centered variational approach. In: CVPR (2010)
3. Huguet, F., Devernay, F.: A variational method for scene flow estimation from stereo sequences. In: ICCV (2007)
4. Neumann, J., Aloimonos, Y.: Spatio-temporal stereo using multi- resolution subdivision surfaces. IJCV 47(1-3), 181–193 (2002)
5. Rabe, C., Müller, T., Wedel, A., Franke, U.: Dense, Robust, and Accurate Motion Field Estimation from Stereo Image Sequences in Real-Time. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 582–595. Springer, Heidelberg (2010)
6. Wedel, A., Rabe, C., Vaudrey, T., Brox, T., Franke, U., Cremers, D.: Efficient Dense Scene Flow from Sparse or Dense Stereo Data. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 739–751. Springer, Heidelberg (2008)
7. Zhang, Y., Kambhamettu, C.: On 3-d scene flow and structure recovery from multiview image sequences. IEEE Trans. on Sys. Man Cyber. B 33(4), 592–606 (2003)
8. Vogel, C., Schindler, K., Roth, S.: 3D scene flow estimation with a rigid motion prior. In: ICCV (2011)
9. Valgaerts, L., Bruhn, A., Zimmer, H., Weickert, J., Stoll, C., Theobalt, C.: Joint Estimation of Motion, Structure and Geometry from Stereo Sequences. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 568–581. Springer, Heidelberg (2010)
10. Hadfield, S., Bowden, R.: Kinecting the dots: Particle based scene flow from depth sensors. In: ICCV (2011)
11. Wedel, A., Brox, T., Vaudrey, T., Rabe, C., Franke, U., Cremers, D.: Stereoscopic scene flow computation for 3d motion understanding. IJCV 95(1), 29–51 (2011)
12. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. IEEE Trans. on PAMI 32(8), 1362–1376 (2010)
13. Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: CVPR (2006)
14. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: CVPR (2010)
15. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. IJCV 92(1), 1–31 (2011)
16. Ruttle, J., Manzke, M., Dahyot, R.: Estimating 3d scene flow from multiple 2d optical flows. In: International Machine Vision and Image Processing Conference, IMVIP 2009 (2009)
17. Devernay, F., Mateus, D., Guilbert, M.: Multi-camera scene flow by tracking 3-d points and surfels. In: CVPR (2006)
18. Furukawa, Y., Ponce, J.: Dense 3D motion capture from synchronized video streams. In: CVPR (2008)
19. Wu, T.P., Yeung, S.K., Jia, J., Tang, C.K.: Quasi-dense 3D reconstruction using tensor-based multiview stereo. In: CVPR (2010)
20. Pons, J., Keriven, R., Faugeras, O.: Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. IJCV 72(2), 179–193 (2007)

21. Pfister, H., Zwicker, M., van Baar, J., Gross, M.: Surfels: Surface elements as rendering primitives. SIGGRAPH (2000)
22. Carceroni, R., Kutulakos, K.: Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape and reflectance. IJCV 49(2-3), 175–214 (2002)
23. Wand, M., Jenke, P., Huang, Q., Bokeloh, M., Guibas, L., Schilling, A.: Reconstruction of deforming geometry from time-varying point clouds. In: Eurographics Symposium on Geometry Processing (2007)
24. Courchay, J., Pons, J.-P., Monasse, P., Keriven, R.: Dense and Accurate Spatio-temporal Multi-view Stereovision. In: Zha, H., Taniguchi, R.-I., Maybank, S. (eds.) ACCV 2009, Part II. LNCS, vol. 5995, pp. 11–22. Springer, Heidelberg (2010)
25. Wu, T.P., Yeung, S.K., Jia, J., Tang, C.K., Medioni, G.: A closed-form solution to tensor voting: Theory and applications. IEEE Trans. on PAMI 34(8), 1482–1495 (2012)
26. Mount, D.M., Arya, S.: ANN: A library for approximate nearest neighbor searching (2010), `http://www.cs.umd.edu/~mount/ANN/`
27. Michael Kazhdan, M.B., Hoppe, H.: Poission surface reconstruction. In: Eurographics Symposium on Geometry Processing (2006)