https://doi.org/10.2352/ISSN.2470-1173.2019.3.SDA-657

Semi-automatic post-processing of multi-view 2D-plus-depth video

Braulio Sespede [1], Florian Seitner [2], Margrit Gelautz [1]
[1] Institute of Visual Computing and Human-Centered Technology, TU Wien; Vienna, Austria
[2] emotion3D GmbH; Vienna, Austria

Abstract

We propose a post-processing framework based on multi-view interactive video segmentation for correcting 2D-plus-depth video footage. The suggested approach uses user-made scribbles to guide the multi-view segmentation process, which is based on an efficient cost-volume filtering algorithm. We extend the 2D algorithm to 3D and propose several improvements that increase precision and recall while also decreasing the need for user input. Our semi-automatic approach is supported by an interactive visualization tool that integrates both 2D and 3D views of the footage, allowing the user to explore novel views coherently and grasp a better understanding of the underlying data. We integrate our post-processing framework into a workflow for generating dynamic meshes from footage recorded by multiple stereo cameras, demonstrating the applicability of the technique.

Introduction

In recent years, immersive 3D experiences such as virtual reality and augmented reality [1] have increased the demand for real-world 3D assets. In order to capture 3D models from real-world scenes, active and passive depth sensing techniques are frequently used. Depth sensing techniques usually lead to 3D models with geometric imperfections, as the depth reconstruction techniques have certain limitations (e.g., due to uniform color in stereo analysis algorithms or background light in the case of time-of-flight cameras). As a consequence, correction by a rotoscoping artist is usually required during post-production in order to prepare the footage for real-world usage.

In this context, we propose a technique that can be used during post-production to improve the geometric accuracy of multi-view 2D-plus-depth video by using sparse user cues to correct specific parts of a scene. The semi-automatic correction algorithm requires (i) a fast video segmentation algorithm capable of understanding user-given cues and propagating them in a temporally and spatially consistent manner, (ii) minimal user input and interactive response times, and (iii) a visualization tool capable of rendering multi-view 2D-plus-depth videos to provide feedback and guide the correction process. In order to achieve these goals, we propose a semi-automatic algorithm capable of tracking movement in multi-view video, while achieving interactive response times for high resolutions. The user is then able to apply different filters to the underlying depth of specific parts of the scene. To this end, we adapt the video segmentation algorithm proposed in [2] to multi-view scenes. The reasons behind this choice are: (a) we can achieve high-quality results with a moderate amount of user input, (b) the algorithm supports further refining of the results in an interactive manner, and (c) real-time performance can be achieved by making use of an efficient cost-volume filtering approach implemented on the GPU [3].

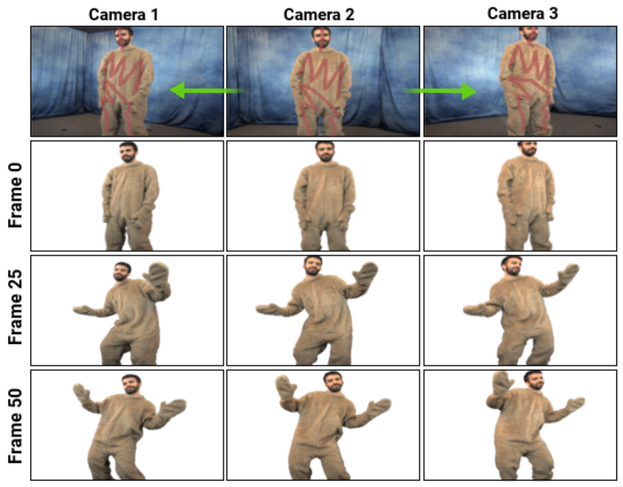


Figure 1. *Examples of segmentation results. The first row shows the user input over the center camera (camera 2) and propagated scribbles on nearby cameras (camera 1 and camera 3). The second to fourth row show the results of the proposed multi-view post-processing algorithm with scribbles on the first frame alone (the scribbles are drawn in red in the first row).*

Our main contribution is the adaptation of [2] to enable the correction of multi-view 2D-plus-depth video, and a visualization approach to complement it for more intuitive usage. In this work, we focus on disparity maps resulting from a stereo analysis algorithm [12], but it can be applied to any other sources of depth data in principle. We extend the ideas proposed in [2] in several ways: (i) we use depth as an additional cue to reduce ambiguity during segmentation, (ii) we incorporate ideas from background subtraction techniques to reduce false positives in our scene model, (iii) we extend the algorithm to multi-view settings by propagating user input between different camera views, thus effectively turning a 2D video segmentation technique into a 3D video segmentation technique. The proposed modifications result in reduced user input and improve the precision and recall of the resulting 3D masks.

Related Work

Several papers have been published with regards to the use of user input to account for errors in stereo matching algorithms (e.g. [4][5][6]). Instead of making corrections on the resulting disparity maps, these methods loop between user input and stereo correspondence calculation. While some of these approaches give good results (e.g. overcoming frequent stereo matching errors such as along discontinuities in color ambiguous scenes), they do not address video content and multi-view scenes at the same time.

Doro et al. [4] propose incorporating smoothness, discontinuity and depth ordering constraints into a multi-view global stereo algorithm. These constraints aim to correct common pitfalls of stereo algorithms through simple brush-based annotations. Even though global stereo algorithms tend to be slower compared to local stereo algorithms, a GPU implementation makes this approach feasible at interactive frame-rates. As a weakness, the approach does not account for video footage and requires a lot of user input, making it unfeasible for dynamic scenes.

Following a similar approach, Zhang et al. [5] also incorporate a smoothness and discontinuity brush by modifying the energy function, but only for binocular stereo content. Additionally, they use a segmentation tool based on GraphCut [7] to bias the matching procedure towards the segmented objects in the left and right view. Their algorithm also allows the user to fit specific geometric models such as planes and spheres into the selection, which allows to easily correct common surfaces such as walls.

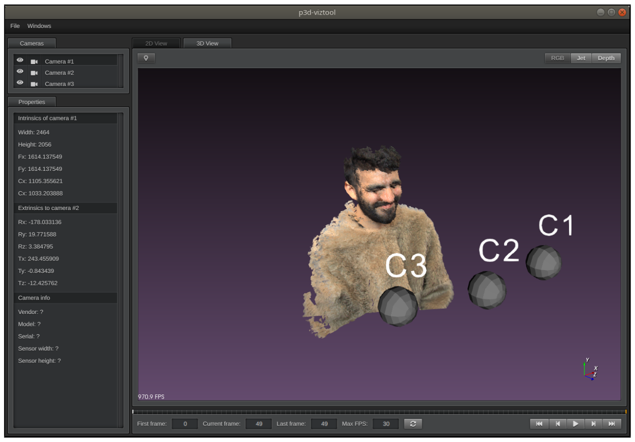
Ruhl et al. [6] incorporate user input into a local stereo algorithm due to the inherently faster computation speed of local methods. To correct stereo content, the authors propose setting constraints to the cost-volume directly, in order to restrict the possible labels. The correction is achieved by selecting a region on the disparity maps and letting the user define a valid cost range for said selection. To set a valid cost range, the user interacts with the point cloud and shifts the selection along the z-axis in the form of a cost-block, indicating a valid range of disparities. Even though this approach is more general than the work of Doro et al. [4] and Zhang et al. [5] in terms of the corrections it can perform, it requires precise selections, and does not account for dynamic content.

Method Description

Below, we describe the different components of our post-processing pipeline, which is composed by: propagation of user-input to multiple views, translation of the user input into a cost-volume which represents areas of interest for sections of the video (with our modifications to [2]), the CUDA based spatio-temporal filtering of the cost-volume, and visualization and correction of the segmentation masks.

Visualization The first step to correcting the 2D-plus-depth footage is to identify areas that need to be corrected. To enable this process, we project the information from all available views into the same 3D space by using the camera set-up's calibration information and the stereo-derived depth data, thus building a common point cloud representation. The implemented 3D visualization also supports dynamic scenes, allowing the detection of errors such as insufficiently synchronized cameras, which may be more difficult to detect when visualizing 2D disparity maps alone. Figure 2 shows an example of the visualization tool and its user interface. Following the idea of [8], every step of the semi-automatic post-processing pipeline gives the user a corresponding visual feedback.

Propagating User-input Between Views The user annotates one of the views with a scribble S_{2D}, which identifies a foreground object throughout the video. Then, the annotated scribble



Camera #1
Camera #2
Camera #3

Properties

Intrinsics of camera #1
Width: 2464
Height: 2056
Fx: 1614.137549
Fy: 1614.137549
Cx: 1105.355621
Cx: 1033.203888

Extrinsics to camera #2
Rx: -178.033136
Ry: 19.771588
Rz: 3.384795
Tx: 243.455909
Ty: -0.843439
Tz: -12.425762

Camera info
Vendor: ?
Model: ?
Serial: ?
Sensor width: ?
Sensor height: ?

970.9 FPS

First frame: 0 Current frame: 49 Last frame: 49 Max FPS: 30

Figure 2. *Screenshot of the visualization tool used to support the post-processing of dynamic scenes. In this case the scene was recorded using 3 stereo camera (indicated by gray spheres).*

is projected into 3D space using the camera parameters and the depth data from the annotated reference camera C_o, forming a point cloud of the scribble, S_{3D}. Afterwards, we project the point cloud back into the 2D spaces of the other cameras using their respective camera parameters, forming a projected scribble $S^{'}_{2D}$ for the other views. Assuming the target camera C_t is modeled after an undistorted pinhole camera, and if $p^{'} = (u,v) \in S^{'}_{2D}$ and $p = (x,y,z) \in S_{3D}$, then:

$$x^{aux} = x/z$$

$$y^{aux} = y/z$$

$$u = f^{'}_x * x^{aux} + c^{'}_x$$

$$v = f^{'}_y * y^{aux} + c^{'}_y$$

where $f^{'}_x$ and $f^{'}_y$ are the horizontal and vertical focal lengths of C_t, and $c^{'}_x$ and $c^{'}_y$ coordinates of its principal point. The projected pixels $p^{'}$ are used to guide the segmentation process in their respective views, thus reducing the need for user annotation.

Building the Cost-volume Once the foreground scribbles have been propagated to other views, we use the RGBD information from the underlying pixels to build a foreground model F based on a histogram, H_f. In extension of the approach proposed by [2], each bucket of the histogram identifies a combination of color and depth values. The user can specify how many buckets are used to describe the depth of the scene, as well as its color. Regarding the background model B, we noticed that the original approach presented in [2] introduced foreground data into the background model, often requiring further editing of the final mask in our application. It is frequent in background subtraction algorithms to construct a reference image representing the background. Following this idea, we use the RGBD data of an empty scene to build the background histogram, H_b. Other methods such as calculating the median of an RGBD scene could also be used to build the background reference [11]. Once both histograms have been created, we build the cost-volume $P(x,y,t)$ which contains the probabilities $p_i \in [0,1]$ that the pixel $i = (x,y,t)$ belongs to F:

$$p_i = \frac{H_f(i)}{H_f(i) + H_b(i)} \tag{1}$$

In order to obtain the segmentations masks we could simply threshold the cost-volume by keeping pixels whose p_i exceeds 0.5 in (1), but this would result in noisy spatio-temporal masks due to an incomplete color model. To account for this, an optimization framework that smooths the cost-volume while preserving edges [3] is used in the next processing step.

Spatio-temporal Filtering Following the idea proposed in [3], we filter the cost-volume by applying an edge preserving filtering algorithm. In our case, we use the fast guided filter [9], as it can increase the computation speed of the original algorithm [10] significantly. The main property of the guided filter is that pixels are filtered using a weighted average of pixels that are similar in color and are spatio-temporally nearby. The main difference to several other algorithms that use guidance images is that the guided filter has a complexity independent of the windows size. Once the cost-volume has been filtered, the cost-volume is thresholded so that only the pixels i with $p_i > 0.5$ remain. Finally, we ensure that the resulting masks are temporally connected to the input scribbles.

GPU Implementation As the authors of the guided filter mention in the original publication [9], the most computational expensive part of the algorithm is the box filter, which is used for auxiliary operations. Fortunately, as described in [3], the box filter can be implemented using the sliding window technique. This technique works by applying a sequence of lower-dimensional box filters in each direction while still obtaining the same result (i.e., first filtering in the x direction, then y, and finally t). Using the sliding window technique makes the box filters $O(N)$ operations, where N is the number of pixels in the image. Moreover, every slice of each directional box filter can be filtered in a parallel manner, making use of the large amount of cores provided by modern GPUs.

Another important aspect for the GPU implementation is how memory is accessed and allocated in the GPU memory, as uncoalesced memory accesses can decrease the performance significantly. For this reason, in our implementation the guidance image and the cost-volume are represented as an array where each element contains the RGB information from the guidance image followed by the value of the cost-volume for that pixel. The main constraint in our implementation was found to be the GPU memory, as only a limited number of frames can be uploaded to the GPU when resolutions are high, resulting in a lot of transactions between CPU and GPU. To address this issue, the video was divided into chuncks that fit in GPU memory and are processed sequentially.

Experimental Results

Our post-processing pipeline was implemented and evaluated using an Intel Core i7-7800X CPU at 3.5GHz. The GPU used is an NVIDIA GeForce GTX 1080 with 4GBs of RAM. In order to test the algorithm we used a self-recorded data set, which was captured using 3 stereo cameras and has a resolution of 2464x2056. To evaluate the segmentation algorithm, we set the parameters to constant values: $\{r_s, r_t, \varepsilon, bins_{rgb}, bins_{depth}\} = \{20, 1, 400, 50, 10\}$, where r_s and are r_t are the spatial and temporal radius of the filtering algorithm, respectively, ε the smoothing factor of the filtering algorithm, and $bins_{rgb}$ and $bins_{depth}$ the number of buckets representing RGB and depth information, respectively.

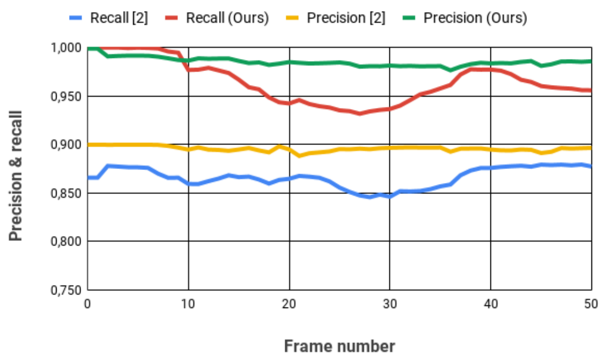


Figure 3. *Precision and recall over time between [2] and our improved version of the algorithm.*

Multi-view Video Segmentation

The video used for evaluation consists of 50 frames. In the evaluation we compare the results delivered by our modified approach to those achieved by the original algorithm [2]. To provide a fair comparison we use the same scribbles, which are used to annotate only the first frame. Qualitative results can be seen in Figure 1. During our evaluation we noticed that the original algorithm was prone to segmentation bleeding over time due to similar colors in background and foreground. While we overcome this problem and obtain more precise masks, our algorithm requires more annotated frames when the video contains frequent changes in depth.

The first experiment to quantify the results was to evaluate the precision and recall of the segmentation masks over time, using manually annotated data [16]. The precision measures the percentage of samples out of the proposed mask that were actually foreground samples. The recall measures the percentage of correct samples out of the total correct foreground samples:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

where TP stands for true positives, FP for false positives, and FN for false negatives. Observing Figure 3, we can see that our approach has higher precision and recall compared to [2]. This was particularly noticeable in color ambiguous scenes where the additional depth cue can produce a less ambiguous model. Another aspect that can be observed in Figure 3 is that there is a dip in terms of precision for the proposed algorithm between frames 15 to 35 (red curve). After analyzing the scene we concluded that this happened due to a change in depth of the segmented object.

As a second experiment we evaluated the flickering error. The goal of this experiment was to assess the spatio-temporal consistency of the proposed algorithm, which is measured by the pixel-wise flicker error FE of pixel i:

$$FE_i = \sum \frac{|a_i - a_j|}{|I_i - I_j| + 1} \qquad (2)$$

where subscripts i and j denote the indices of temporally neighboring pixels, a is the label value (either background or foreground), and I represents the hue of said pixels. According to

(2), the flickering error is higher when the change in label corresponds to pixels of similar color. Table 1 shows the flickering error when using different segmentation cues for our approach and the results of the initial algorithm [2]. We can observe that the additional depth data not only improves the precision and recall but also reduces flicker under the same conditions.

Table 1: Flickering error

Segmentation cue	Total flickering error
RGB, random samples [2]	35365.44
RGBD, random samples (Ours)	30698.93
RGBD, using empty scene (Ours)	18184.70

As a last experiment we measure the performance improvement from porting the algorithm to the GPU. Table 2 shows the average computational speed for the different parts of the algorithm for a video with 100 frames of 2464x2056 resolution. It can be observed that the time consumption of the cost-volume filtering can be significantly reduced by the GPU implementation. As a result, the total time per frame decreases by a factor of 5.1, achieving an average processing time of 196.6ms per frame.

Table 2: Computation time

Step	Average runtime per frame and camera (in seconds)
Cost-volume computation	0.0202
Cost-volume filtering (CPU)	0.9745
Cost-volume filtering (GPU)	0.1686
Cost-volume thresholding	0.0011
Temporal connected components	0.0067

Applications in Dynamic 3D Reconstruction

We found several applications for the proposed framework, the first one being (i) the evaluation of multi-view dynamic stereo systems [13] and the second (ii) post-processing of point clouds for mesh reconstruction [15].

Regarding the first, it is common for 3D reconstruction algorithms to perform evaluations using ground truth datasets, but for systems using diverging sensor configurations the utility of such publicly available datasets is restricted. In this case, it is common practice to use evaluation systems based on novel views, such as [14]. Precise masks are necessary for multi-view reconstruction systems that require such type of evaluation, being a good candidate for the proposed method.

Regarding the second use, mesh reconstruction algorithms usually apply high degrees of regularization to deal with noise in the input point clouds, which results in the loss of high frequency details. In this context, the proposed technique can be used to remove spurious points introduced by incorrect depth estimations, resulting in higher-quality meshes. Figure 4 shows an example of such a reconstruction.

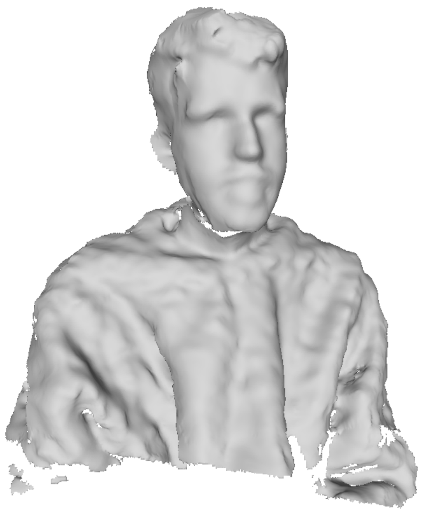


Figure 4. *Mesh-based reconstruction using the post-processed point cloud from Figure 2.*

Conclusion

We propose a post-processing approach for multi-view 2D-plus-depth videos based on a semi-automatic segmentation algorithm. The proposed approach can provide interactive performance when implemented on the GPU. Our main contribution was to extend a 2D semi-automatic video segmentation algorithm to 3D multi-view settings. Our experiments demonstrated that the addition of depth as a cue, as well as the use of background references as a model can improve the precision and recall of the resulting segmentation masks. Furthermore, these additions can reduce temporal flicker by up to 50 percent. Finally, we introduced some possible applications in the domain of 3D reconstruction.

Acknowledgments

This work has been funded by the Austrian Research Promotion Agency (FFG) and the Austrian Ministry BMVIT under the program ICT of the Future (project ”Precise3D”, grant no. 855442).

References

[1] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L. Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip A. Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi, Holoportation: Virtual 3D teleportation in real-time, Proc. UIST, pg. 741754, (2016).

[2] Nicole Brosch, Asmaa Hosni, Christoph Rhemann, and Margrit Gelautz, Spatio-temporally coherent interactive video object segmentation via efficient filtering, Proc. DAGM, pg. 418427, (2012).

[3] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz, Fast cost-volume filtering for visual correspondence and beyond, IEEE Trans. Pattern Anal. Mach. Intell., 35, 2, (2013).

[4] Yotam Doron, Neill DF Campbell, Jonathan Starck, and Jan Kautz, User directed multi-view-stereo, Proc. ACCV, pg. 299313, (2014).
[5] Chenxi Zhang, Brian Price, Scott Cohen, and Ruigang Yang, High-quality stereo video matching via user interaction and space-time propagation, Proc. IC3DV, pg. 7178, (2013).
[6] Kai Ruhl, Martin Eisemann, and Marcus Magnor, Cost volume-based interactive depth editing in stereo post-processing, Proc. CVMP, pg. 16, (2013).
[7] Yuri Boykov and Vladimir Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, IEEE Trans. Pattern Anal. Mach. Intell., 26, 9, (2004).
[8] Jia He, Chang-Su Kim, and C-C Jay Kuo, Interactive Segmentation Techniques: Algorithms and Performance Evaluation, Springer Science & Business Media, 2013.
[9] Kaiming He, Jian Sun, and Xiaoou Tang, Guided image filtering, Proc. ECCV, pg. 114, (2010).
[10] Kaiming He and Jian Sun. Fast guided filter, CoRR, abs/1505.00996, (2015).
[11] Massimo Piccardi, Background subtraction techniques: a review, Proc. ICSMC, pg. 30993104, (2004).
[12] Florian Seitner, Matej Nezveda, Margrit Gelautz, George Braun, Christian Kapeller, Werner Zellinger, Bernhard Moser, Trifocal system for high-quality inter-camera mapping and virtual view synthesis, Proc. IC3D, pg. 1-8, (2015).
[13] Christian Kapeller, Evaluation of a 3d reconstruction system comprising multiple stereo cameras, Masters thesis, (2018).
[14] Michael Waechter, Mate Beljan, Simon Fuhrmann, Nils Moehrle, Johannes Kopf, and Michael Goesele, Virtual rephotography: Novel view prediction error for 3D reconstruction, ACM Trans. Graphics, 36, 1, (2017).
[15] Christian Kapeller, Braulio Sespede, Matej Nezveda, Matthias Labschutz, Simon Flory, Florian Seitner, and Margrit Gelautz, A workflow for 3D model reconstruction from multi-view depth acquisitions of dynamic scenes, Proc. OAGM, pg. 116119, (2018).
[16] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski, High-quality video view interpolation using a layered representation, ACM Trans. Graphics, 23, 3, (2004).

Author Biography

Braulio Sespede received his joint MSc in Software Engineering from UAS Technikum Wien and Buenos Institute of Technology in 2018. His research interests lies in the domain of interactive computer graphics techniques and 3D imaging.

Florian Seitner received his M.S. and Dr. Techn. (PhD) degrees in Computer Science from the Vienna University of Technology in 2006 and 2013, respectively. Since 2009 he is leading emotion3D's R&D team in the areas of 3D range imaging, mobile vision and 3D displaying technologies.

Margrit Gelautz received her PhD in Telematics from Graz University of Technology (1997). She is an associate professor at Vienna University of Technology with focus on image and video analysis & synthesis techniques, with special interest in 3D film applications.