2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Moving Object Reconstruction in Monocular Video Data Using Boundary Generation

Sebastian Bullinger, Christoph Bodensteiner, Sebastian Wuttke and Michael Arens
Department Object Recognition
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, 76275 Ettlingen, Germany
Email: {sebastian.bullinger, christoph.bodensteiner, sebastian.wuttke, michael.arens}@iosb.fraunhofer.de

***Abstract*—We present a method to reconstruct the three-dimensional shape of a moving instance of a known object category in video data. We exploit state-of-the-art semantic segmentation techniques to extract the object's two-dimensional shape in each frame. Therefore, our method is robust to occlusion, handles stationary objects and extends naturally to multiple video sequences. We apply Structure from Motion (SfM) to previously generated object images in order to compute a three-dimensional representation of the object. Our approach allows us to remove outliers in SfM reconstructions and to compute clean object meshes by leveraging previously computed semantic segmentations and virtual camera positions. We evaluate the accuracy of our method using a multi-view dataset of a moving vehicle. A laser scan serves as ground truth. We applied our algorithm on publicly available video data and on 25 sequences from our dataset. The algorithm achieves an average point distance of 3.3 cm evaluated on seven trajectories contained in the dataset.**

I. Introduction

A. Moving Object Reconstruction

The reconstruction of moving objects in monocular video data recorded by moving cameras is a challenging task. This is due to the fact that objects usually cover a minority of pixels in video frames. Thus, conventional Structure from Motion or Visual SLAM (VSLAM) techniques treat moving objects as outliers and reconstruct background structures instead. Previous works, e.g. [1] and [2], tackle this problem by considering multiple video frames simultaneously to determine moving parts in the video. They use motion segmentation or keypoint tracking to detect moving objects. Thus, their approach can not handle stationary objects and is vulnerable to occlusion and limited to single shots. In contrast, we exploit state-of-the-art semantic segmentation Convolutional Neural Networks (ConvNets), see for example [3] and [4], to extract the object's two-dimensional shape in each frame. Thus, our method is robust to occlusion, handles stationary objects and extends naturally to multiple video sequences. We refine the ConvNet segmentations by applying the GrabCut [5] algorithm. Next, we apply SfM to all images, even from different shots. Therefore, we leverage the full available three-dimensional shape information. Our approach works with any standard SfM pipeline. We propose a method to remove outliers in object reconstructions using semantic information computed by the corresponding ConvNet. However, SfM point clouds of moving objects with reflecting surfaces show high outlier ratios and varying point densities. Meshes built on top of these point clouds show irregular surface properties. For that reason, we present an algorithm generating points consistent to the semantic information provided by the ConvNet. The resulting point cloud consists of uniformly distributed points with consistent normal vectors. We compute clean object representations superior to Structure from Motion based meshes using these point clouds.

B. Contribution

We propose a pipeline to reconstruct the three-dimensional shape of moving instances of rigid object categories in video data. We present a method to compute object point clouds consistent to constraints derived from semantic segmentations. Because the resulting point clouds have high point densities they are suitable for computing watertight meshes.
Due to the lack of publicly available video data of moving objects with corresponding three-dimensional shape ground truth we created an appropriate dataset. This dataset consists of videos capturing a car performing several maneuvers. As ground truth serves a laser scan of the vehicle present in the video sequences. We evaluate the accuracy of our algorithm using this dataset which is publicly available on our website[1]. In addition, we compare our method visually to previous works. We achieve superior results on public available Internet video data.

C. Related Work

Our method builds on top of recent semantic segmentation as well as 3D reconstruction techniques.
Semantic segmentation is the task of providing semantic information at pixel-level. Early works, like Rother et al. [5], require rough human annotated fore- and background information to compute an exact fore- and background segmentation. In contrast, semantic segmentation approaches using pre-trained ConvNets do not require any supervision at operating time. Early semantic segmentation approaches using ConvNets, e.g. Farabet et al. [6], exploit patchwise training. Long et al. [3] proposed a new architectural style of ConvNets, so called Fully Convolutional Networks, which are trained end-to-end. These ConvNets inspired many state-of-the-art semantic segmentation approaches. Different works

[1]Project page: http://s.fhg.de/boundarygeneration

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

[4], [7], [8] combine Convolutional Networks with Conditional Random Fields. Recently, Dai et al. [9] proposed an instance-aware semantic segmentation approach.

The field of Structure from Motion can be divided into iterative and global approaches. Iterative or sequential SfM methods [10]–[13] are more likely to find reasonable solutions than global SfM approaches [12], [13]. However, the latter are less prone to drift. SfM point clouds provide only sparse geometric information. Thus, we are interested in dense point clouds [14], [15] and watertight meshes [15], [16]. Mesh colorization can be performed using [17].

Previous moving object reconstruction approaches using video data usually exploit some kind of color or motion detection. Feng et al. [18] present a color-based segmentation to achieve 3D monocular tracking. Yuan et al. [19] and Kundu et al. [1] use motion segmentation to distinguish objects and background. Yuan et al. of [19] use this information to apply SfM to single objects, where as Kundu et al. [1] perform multibody VSLAM. In contrast to previous methods, Lebeda et al. [2] use feature tracking in order to extract moving objects in unstructured video data. The object shape is visualized using watertight meshes.

D. Paper Overview

The paper is organized as follows. In section II-A we describe the computation of object images, i.e. pictures where non-object pixels are masked out. We explain the reconstruction of objects using object images in section II-B. In section II-C we describe the removal of outliers exploiting semantic segmentations. We present in section II-D an approach to compute an approximation of the object boundary by combining information derived from SfM and semantic segmentation computations. In section III we evaluate the presented method on our dataset and compare it visually to state-of-the-art results using publicly available video data. Section IV concludes the paper.

II. Methods

A. Computation of Object Images

Given a frame of an input video sequence, we extract image parts showing only the object category of interest using the semantic segmentation approach presented by Zheng et al. in [4]. The ConvNet computes for a given input image the corresponding semantic segmentation at pixel level. However, the result of the ConvNet yields no smooth edges along the object's contour, i.e. a large number of pixels along the objects contour are misclassified. We refine the segmentation by applying GrabCut [5]. The algorithm requires a mask containing possible and obvious foreground as well as possible and obvious background pixels as input. We apply a dilation and an erosion operation to the semantic object mask computed by the ConvNet, simultaneously. This produces two new masks. Let M_{conv}, M_{dil} and M_{ero} be pixel sets corresponding to the masks generated by the ConvNet, the dilation and the erosion operation, respectively. We set the obvious foreground to M_{ero}, the possible foreground to $M_{conv} \setminus M_{ero}$, the possible background to $M_{dil} \setminus M_{conv}$ and the obvious background to ${M_{dil}}^C$ as shown in Fig. 1. Thus, GrabCut determines the object contour along the segmentation provided by the ConvNet within the image area $M_{dil} \setminus M_{ero}$. Finally, we use the GrabCut segmentations to compute object images, i.e. pictures where non-object pixels are masked out.

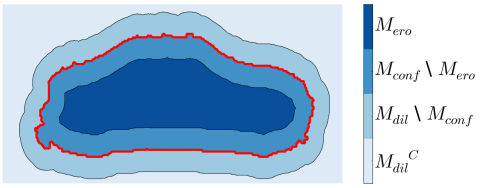


Fig. 1. Illustration of the pixel sets used to refine the original ConvNet segmentation. The red line shows the shape of the original ConvNet segmentation.

B. 3D Object Reconstruction and Virtual Camera Filtering

In contrast to the original frames, object images contain only camera pose information relative to the object. The scene is equivalent to one, where the virtual camera is moving and the object is stationary. To reconstruct the object any off-the-shelf Structure from Motion tool chain like [11], [12], [13] or [15] can be used.

The SfM computation produces a point cloud representing the object and parameters of corresponding virtual cameras. Let n be the number of virtual cameras. n may be smaller than the number of input frames due to failed image registrations. SfM reconstructions contain sometimes single isolated virtual cameras. However, valid virtual camera positions extracted from the same video sequences possess usually similar distances to their respective closest virtual camera, since the camera as well as the object move and rotate gradually from frame to frame in each scene. In order to detect isolated cameras, we compute for each camera i the distance d_i to the respective nearest neighbor. We assume that there are less than 25 % isolated cameras. If there are more than 25 % isolated cameras it is likely that the reconstruction is degenerated and not useful at all. Thus, we consider the 75th percentile p_{75} of all distances $\{d_i | i = 1, \ldots, n\}$ as a valid nearest neighbor distance. In video data the distance of valid virtual cameras to their nearest neighbor is limited due to the gradual movement of object and capturing device. Thus, we define a validity threshold t_{val} which describes the accepted exceeding of p_{75}. t_{val} should be several orders of magnitude greater than p_{75}. We compute the validity of a given camera by testing if $d_i < t_{val}$ holds. The removal of isolated cameras is important for the algorithms presented in section II-C and II-D.

C. Objectness and Outlier Removal

Misclassified pixels in the semantic segmentation sometimes lead to noise in the resulting three-dimensional reconstruction. We determine outliers in the object point cloud by computing the objectness for each 3D point X_j. We use the camera calibrations K_i, rotations R_i and centers C_i estimated during the Structure from Motion process to project the points X_j

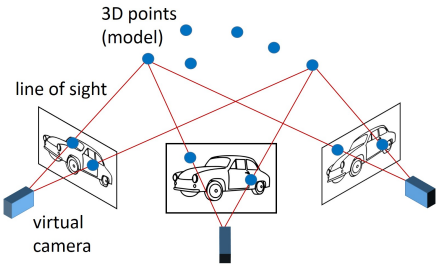


Fig. 2. Projection of 3D points onto the image plane for each virtual camera in order to compute the objectness for each object point.

TABLE I
PROJECTION POSSIBILITIES

	3D Object (O.) Point	3D Background (B.) Point
Visible	Projected on O.	Projected on B.
Occluded by O.	Projected on O.	Projected on O.
Occluded by B.	Projected on B.	Projected on B.

onto the focal plane of each virtual camera cam_i. Here, i and j denote camera and point indices, respectively. Let x_{ij} denote the projection of X_j into the virtual camera cam_i. The mapping of X_j to x_{ij} is performed according to equation 1, where X_j is represented in homogeneous coordinates.

$$x_{ij} = K_i R_i [I| - C_i] X_j. \tag{1}$$

The mapping of equation 1 is visualized in Fig. 2. By analyzing the projections x_{ij} of point X_j we determine a measure for the objectness of this point. Projections of a point lying behind a virtual camera are in the corresponding image not visible and therefore not regarded. For each visible projection x_{ij} of a point X_j we use the corresponding segmentation information in order to determine if the point belongs to the object or background category. This way, we can count the number of projections projected onto object category pixels.
Defining a threshold ratio r_{object} allows us to filter the three-dimensional points, i.e. we keep only points for which equation 2 holds.

$$nop_j > r_{object} \times nvp_j \tag{2}$$

Here nop_j and nvp_j are the number of object projections and visible projections of point j, respectively. By weighting each camera equally and without any prior knowledge the optimal decision is achieved using $r_{object} = 0.5$.
However, if the video data contains low object occlusion a better choice is possible. Let us assume for simplicity that our segmentation is perfect for a certain picture. Table I shows which projection cases occur for a 3D point in such an image. The evaluation of 3D point projections can be understood as a 3D object point classification task. A False Positive (FP) describes the case where a background point is being considered as part of the object and a False Negative (FN) represents the complementary situation. The cases FP and FN may lead to a wrong filtering of 3D points. In order to handle FPs resulting from 3D points close to the object surface

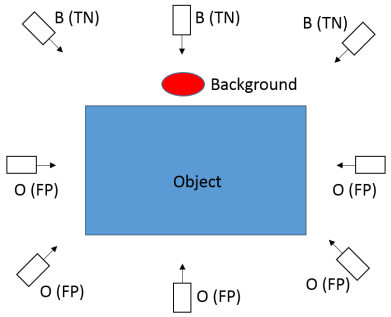


Fig. 3. An example showing the projection results of eight different cameras with respect to the background cluster. All object (O) projections are False Positives (FPs). Only a high emphasis of background (B) projections, i.e. True Negatives (TNs), will remove the background cluster correctly. Concretely, r_{object} must be greater than $\frac{5}{8}$.

we give background projections more emphasis. This can be achieved by selecting a high r_{object} value. Fig. 3 shows an example.
However, increasing r_{object} will also increase the influence of FNs. Fortunately, video data with low object occlusion and stable segmentation results allows us to increase r_{object} to an appropriate level. For example, we used $r_{object} = 0.98$ in our experiments.

D. 3D Boundary Generation

Reconstructions of moving objects with reflecting surfaces, e.g. cars, often produce outliers and points with incorrect normals. SfM computations typically find only a small number of points on homogeneous surfaces. Both facts often lead to meshes with poor shapes. We compute clean object meshes by replacing the original object point cloud with virtually generated points. We exploit virtual camera poses, computed during the SfM reconstruction process, as well as semantic segmentations to create 3D points consistent to the object shape. The generated points are uniformly distributed and show consistent normal vectors.
First, we compute a three-dimensional bounding box corresponding to the original sparse point cloud. Next, we divide the space of this bounding box equally in $\mathcal{O}(k^3)$ subspaces and represent each cell with one point at the center. Here k is the number of subdivisions in each dimension. By applying the algorithm described in section II-C we can assign an object or a non-object flag to each point in the grid. This divides the space of the bounding box in an object and a background volume. We compute the corresponding object boundary by removing all background points as well as all non-boundary points in the object volume. A point is considered as a boundary point if and only if one of the corresponding neighbors has a complementary flag.
Unfortunately, the computation of an accurate surface reconstruction using this approach is not reasonable as it requires $\mathcal{O}(k^3)$ (non-)object flag computations. Therefore, we extend the approach as follows. First, we use a coarse subdivision of the bounding box space to compute an initial set of boundary

Fig. 4. Outline of the Boundary Generation Algorithm

$BPs_1 = BPs_{Coarse}$
for $i = 1...k$ **do**
  $PBPs_i \leftarrow \emptyset$
  for $p \in BPs_i$ **do**
    $NNs \leftarrow$ getNearestNeighbors$(6, p, BPs_i)$
    $RNs \leftarrow$ getRealNeighbors(p, NNs)
    $PBPs_i \leftarrow PBPs_i \cup$ createPossibleBPs(p, RNs)
  end for
  $PBPs_i$ = assignFlagsToPoints$(PBPs_i)$
  $PBPs_i$ = $PBPs_i \cup BPs_i$
  $BPs_{i+1} \leftarrow \emptyset$
  for $p \in PBPs_i$ **do**
    $NNs \leftarrow$ getNearestNeighbors$(26, p, BPs_{i+1})$
    $RNs \leftarrow$ getRealNeighbors(p, NNs)
    if isBoundaryPoint(p, RNs) **then**
      $BPs_{i+1} \leftarrow BPs_{i+1} \cup \{p\}$
    end if
  end for
end for

points, i.e. points lying on the boundary of the object or the background volume. Next, we create points on a more fine-grained level inside the space spanned by the current boundary points which are possibly closer to the true object boundary. Thus, we assign (non-)object flags to the newly generated points. This allows us to update the set of real boundary points and thus to adjust the shape of the object boundary represented by these points. We repeat these computations, iteratively.

The basic steps of our algorithm are depicted in Fig. 4. We initialize the set of boundary points (BPs_i) in iteration 1 with boundary points computed at a coarse division of the bounding box space (BPs_{Coarse}). In each iteration i we build a kd-tree containing all points of BPs_i to efficiently determine the nearest neighbors of each boundary point. It is important to note that the neighbors of a boundary point differ in terms of their distance. The different neighbor types are illustrated in Fig. 5a. The varying distances must be considered while using the kd-tree to determine the nearest neighbors of a boundary point. We iterate over the current boundary points and determine for each BP the set of neighbors with distance d to create a set of possible boundary points ($PBPs$). We use the points $(x + d, y, z)$, $(x, y + d, z)$ and $(x, y, z + d)$ to compute new points on a on a more fine-grained level according to Fig. 5b. Here, (x, y, z) represents the three-dimensional coordinate of the current boundary point. We generate only a subset of points or no new points at all, if there are less or no points meeting the criteria above. This is equivalent to a division of the cell into eight cuboids as well as increasing the point density by a factor of two, if applied to all boundary points. Next, we compute (non-)object flags for all newly generated $PBPs$. By removing all non-boundary points in $PBPs_i \cup BPs_i$ we adjust the boundary. We analyze all 26 neighbors of each boundary point. After several iterations we cover the space on a fine-grained level.

(a) Neighbor types of a boundary point.

(b) Creation of possible boundary points.

Fig. 5. (a) For reasons of clarity, this figures shows only one of eight grid cells which are considered to determine the neighbors of a boundary point (black). A boundary point has maximal 26 neighbor points. Six neighbors with distance d (red), twelve neighbors with distance $\sqrt{2}d$ (blue), and eight neighbors with $\sqrt{3}d$ (green), where d is the length of the edges of the cells in the grid. (b) Creation of seven new possible boundary points (blue) using three neighbors with distance d (red).

The generated boundary point set is used to generate a mesh describing the object contour.

The number of required iterations depends on the granularity of the points before the first iteration d_0 as well as the desired point density d_i. In each iteration the distance between points is halved. To reach a point density of d_d the algorithm requires $\lceil \log \frac{d_0}{d_i} \rceil$ iterations.

III. Experiments and Evaluation

A. Dataset

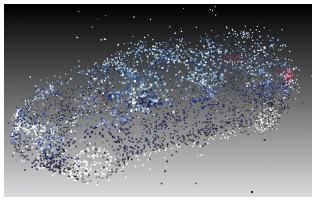
The dataset[2] consists of 25 videos capturing a car moving on eight different trajectories. The video sequences cover a high variety of viewing angles. We created a laser scan, see Fig. 7a, of the vehicle present in the dataset serving as ground truth. The laser scans were acquired using a Zoller+Fröhlich scanner. The distance to the reflecting object is estimated based on the phase shift between received and emitted signal. The scanning head was operated on a rigid tripod which results in ranging accuracies of a few millimeters. The laser scans are automatically registered. Noise in the measurement is manually removed.

B. Semantic Segmentation

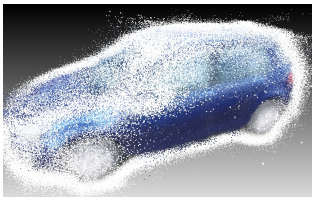
All evaluations presented in the following use the semantic segmentation ConvNet proposed by Zheng et al. [4]. We choose this one over [3] since the latter creates a less accurate silhouette, produces sometimes false positives as well as disconnected components. All segmentations have been refined using the method described in section II-A. In order to compute the dilation and erosion we select an ellipse as structuring element and set its radius to ten pixels, since our investigation of different segmentation samples showed that the boundary inaccuracies of the ConvNet are usually smaller than five pixels. Fig. shows 6e the refinement of a ConvNet segmentation [4].

[2]Project page: http://s.fhg.de/boundarygeneration

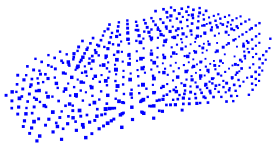
(a) Input Image Sequence



(b) Sparse Reconstruction with Outliers

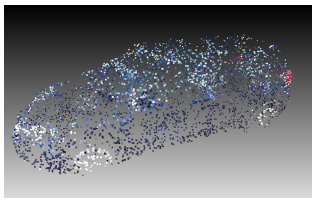


(c) Dense Reconstruction with Outliers

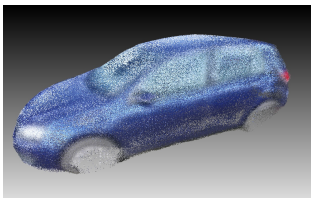


(d) Boundary Generation Iteration 0

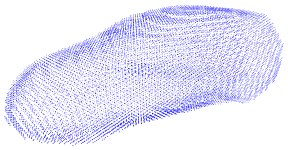
(e) Refined Segmentation



(f) Sparse Reconstruction without Outliers



(g) Dense Reconstruction without Outliers



(h) Boundary Generation Iteration 2

(i) Left: ConvNet Segmentation, Right: Refined Segmentation

(j) Colorized Mesh Using Sparse Points without Outliers

(k) Colorized Mesh Using Dense Points without Outliers

(l) Colorized Mesh Using Boundary Points (Iteration 2)

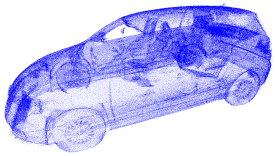
Fig. 6. Results using a single sequence of 510 images.

C. Example Model Computations

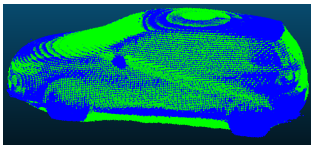
Fig. 6 shows some example results using a video sequence of 510 images viewing the car from all sides. Fig. 6b shows the sparse point cloud of the car using [11]. The dense model representation shown in Fig. 6c is computed applying the Multi-View Stereo algorithm [20]. The stereo matching technique uses a previously computed SfM result to build a depth map for each virtual camera. The dense model is created by projecting the depth values of each virtual camera into the world coordinate system. For both, the sparse and the dense point clouds, outliers are removed using the method described in section II-C. The corresponding results are shown in Fig. 6f and 6g. Also the boundary generation uses the virtual camera poses estimated during the SfM computation. Fig. 6d and 6h show the results after the first and third iteration, respectively. All meshes, see Fig. 6j, 6k and 6l, are computed using the Poisson surface reconstruction algorithm [16]. The colorization is performed by applying [17].

D. Boundary Generation Accuracy Evaluation

We applied our pipeline to all 25 sequences contained in our dataset. In nine cases the SfM fails, i.e. produces multiple models or matches wrong images due to object symmetries (e.g. wheels). In seven of the eight trajectory-types the car is captured from at least three sides. We select for each of these seven trajectories one sequence and compute the distance between the boundary and the laser scan point cloud. We automatically scale and register the boundary point cloud to the ground truth using the Iterative Closest Point implementation of CloudCompare [21]. It is important to note that in contrast to the generated object boundaries the laser scan data contains no points at windows and at the bottom side of the vehicle. Since there is no correspondence information between laser scan and boundary generation points, we use the following steps to find reasonable correspondences. First, we compute for each boundary point the nearest neighbor in the laser scan. We determine the distance between each boundary-laser-scan-point-pair. If multiple boundary points share the same nearest neighbor we keep only the boundary point with the smallest distance. We use the average distance and the standard deviation of the remaining boundary-laser-scan-point-pairs as quality measure. Since our video has no occlusion we use a high object ratio of $r_{object} = 0.98$. Fig. 7 shows the laser scan and an overlay of a generated boundary and the ground truth. Table II shows the evaluation of the seven trajectories



(a) Laser Scan



(b) Overlay of Boundary and Laser Scan Points

Fig. 7. Laser scan (a) and overlay of a generated boundary with the ground truth (b).

TABLE II
EVALUATION BOUNDARY ACCURACY.

Trajectory	1	2	3	4	5	6	7
Average Distance (cm)	4.4	3.0	2.6	3.1	3.2	3.3	3.4
Standard Deviation (cm)	5.8	3.0	2.6	2.9	3.1	3.3	3.4

using the output of the 4th iteration and roughly 1000 cells as initial subdivision.

E. Comparison to Results Presented by Lebeda et al.

We compare our boundary generation method visually to the approach presented in [2] on publicly available video data. The video sequence consists of 76 input images showing a rally car. Fig. 8 contains two example input pictures, the result computed by Lebeda et al. as well as a mesh based on the output of our boundary generation algorithm. The shape of our result is clearly more accurate, i.e. closer to the real shape of the car. A quantitative comparison is due to missing ground truth data not possible.

IV. CONCLUSION

We presented a pipeline to reconstruct the three-dimensional structure of objects in video data using semantic segmentations computed by ConvNets. In contrast to previous approaches, our extraction pipeline does not rely on object movement present in subsequent frames. We are able to extract the object's 2D shape from a single image. This approach has several advantages over previous works, i.e. it is robust to occlusion, handles stationary objects and extends naturally to multiple video sequences. We observed that SfM based point clouds of moving objects with reflecting surfaces result in crumbled meshes due to outliers, irregular point densities and incorrect normal vectors. We successfully tackled this problem by introducing an algorithm combining the information contained in virtual camera poses and semantic segmentations. This approach even constrains surfaces not directly seen by the camera. We applied our algorithm on publicly available video data and on 25 sequences from our dataset. The algorithm achieves an average point distance of 3.3 cm evaluating seven trajectories contained in the dataset using a laser scan as ground truth.

A. Limitations

Currently, our implementation does not handle multiple instances of the same object category. For example, the reconstruction process of two moving cars present in the same video sequence may interfere with each other. However, we are able to tackle this issue by integrating tracking functionality. Furthermore, we observe that our current segmentation pipeline fails to segment objects correctly seen from a bird's eye perspective. This is due to the fact that these views are missing in the training data. Finally, it is important to note that the quality of the derived information is relative to the camera as well as the object motion. Moreover, our method is subject to general SfM constraints.

B. Future Work

In future work, we will enable our pipeline to handle multiple instances of the same object category, independently. Moreover, we intend to analyze the distribution of virtual cameras in the reconstruction to create a more efficient boundary generation algorithm. Another research direction is the combination of boundary information and dense reconstructions such that the resulting point cloud contains accurate structure information as well as regular point densities and consistent normal vectors.

REFERENCES

[1] A. Kundu, K. M. Krishna, and C. V. Jawahar, "Realtime multibody visual slam with a smoothly moving monocular camera," in *Computer Vision, 2011. ICCV 2011. IEEE International Conference on*, 2011. 1, 2

[2] K. Lebeda, S. Hadfield, and R. Bowden, "Dense rigid reconstruction from unstructured discontinuous video," in *Proceedings of the International Conference on Computer Vision workshop on 3D Representation and Recognition*, IEEE. Santiago, Chile: IEEE, 17Dec. 2015. 1, 2, 6, 7

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CVPR (to appear)*, Nov. 2015. 1, 4

[4] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional random fields as recurrent neural networks," in *International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 4

[5] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015720 1, 2

[6] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, August 2013. 1

[7] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *CoRR*, vol. abs/1412.7062, 2014. [Online]. Available: http://arxiv.org/abs/1412.7062 2

[8] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel, "Efficient piecewise training of deep structured models for semantic segmentation," *CoRR*, vol. abs/1504.01013, 2015. [Online]. Available: http://arxiv.org/abs/1504.01013 2

[9] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," *CoRR*, vol. abs/1512.04412, 2015. [Online]. Available: http://arxiv.org/abs/1512.04412 2

[10] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 835–846, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141964 2

[11] C. Wu, "Visualsfm: A visual structure from motion system," 2011. [Online]. Available: http://ccwu.me/vsfm/ 2, 5

[12] P. Moulon, P. Monasse, R. Marlet, and Others, "Openmvg. an open multiple view geometry library." https://github.com/openMVG/openMVG. 2

[13] C. Sweeney, *Theia Multiview Geometry Library: Tutorial & Reference*, University of California Santa Barbara. 2

[14] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010. 2

[15] S. Fuhrmann, F. Langguth, N. Moehrle, M. Waechter, and M. Goesele, "MVE – An Image-Based Reconstruction Environment," *Computer and Graphics*, 2015. 2

[16] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, ser. SGP '06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 61–70. [Online]. Available: http://dl.acm.org/citation.cfm?id=1281957.1281965 2, 5

(a) Image of the video input sequence.

(b) Image of the video input sequence.

(c) Mesh using [2].

(d) Mesh using our boundary generation method.

Fig. 8. Comparison of our boundary generation method using the video sequence presented in [22].

[17] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! Large-scale texturing of 3D reconstructions," in *ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., 2014, vol. 8693, pp. 836–850. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10602-1_54 2, 5

[18] Y. Feng, Y. Wu, and L. Fan, "On-line object reconstruction and tracking for 3D interaction." in *ICME*. IEEE Computer Society, 2012, pp. 711–716. 2

[19] C. Yuan and G. G. Medioni, "3D reconstruction of background and objects moving on ground plane viewed from a moving camera," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, 2006, pp. 2261–2268. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2006.16 2

[20] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *Proceedings of the 11th International Conference on Computer Vision (ICCV 2007)*. Rio de Janeiro, Brazil: IEEE, 2007, pp. 265–270. 5

[21] D. Girardeau-Montaut, "Cloudcompare 2.62," 2016. [Online]. Available: http://www.cloudcompare.org/ 5

[22] K. Lebeda, S. Hadfield, and R. Bowden, "2D or not 2D: Bridging the gap between tracking and structure from motion," in *Proceedings, Asian Conference on Computer Vision (ACCV)*, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-16817-3_42 7