Mesh-based 3D Textured Urban Mapping

Andrea Romanoni[1] Daniele Fiorenti[1] Matteo Matteucci[1]

Abstract— In the era of autonomous driving, urban mapping represents a core step to let vehicles interact with the urban context. Successful mapping algorithms have been proposed in the last decade building the map leveraging on data from a single sensor. The focus of the system presented in this paper is twofold: the joint estimation of a 3D map from lidar data and images, based on a 3D mesh, and its texturing. Indeed, even if most surveying vehicles for mapping are endowed by cameras and lidar, existing mapping algorithms usually rely on either images or lidar data; moreover both image-based and lidar-based systems often represent the map as a point cloud, while a continuous textured mesh representation would be useful for visualization and navigation purposes. In the proposed framework, we join the accuracy of the 3D lidar data, and the dense information and appearance carried by the images, in estimating a visibility consistent map upon the lidar measurements, and refining it photometrically through the acquired images. We evaluate the proposed framework against the KITTI dataset and we show the performance improvement with respect to two state of the art urban mapping algorithms, and two widely used surface reconstruction algorithms in Computer Graphics.

Fig. 1. Textured mesh reconstructed from the KITTI sequence. Notice how moving objects do not appear in the final results

I. INTRODUCTION

The growing interests around autonomous driving has focused the Robotics and Computer Vision communities on specific research areas such as sensing, mapping, and driving policy development [1]. Here we focus on urban dense textured mapping, which plays an important role to enable autonomous navigation through cities.

Surveying vehicles, aiming at city mapping, are usually endowed with 360° laser range finders and monocular or stereo cameras. While laser range finders directly provide accurate 3D measurements of the environment, cameras collect its dense appearance. Successful mapping algorithms have been proposed using cameras or lasers separately, but, to the best of our knowledge, none of them exploits both data sources to build dense textured maps. Only [2] considers both information sources, but the final outcome is a dense stereo matching disparity map, and not a full photoconsistent 3D map of the environment. Moreover, most systems reconstruct a point cloud or a voxelized map, while a continuous mesh of the environment which represents in a dense way the observed scene can lead to more robust navigation or localization and it allows texturing too.

Laser-based mapping algorithms [3] and [4] have shown to produce accurate maps of the environment, but fine-grained details are often discarded, due to sparsity of the

3D point clouds, and they can include moving points if not properly filtered from the laser scans. Feature-based image-based mapping algorithms [5], [6] and [7] build the map reconstruction on 3D points estimated through robust 2D to 2D correspondences; they are able to discard moving points from the final outcome, nevertheless the resulting map is still a sparse point cloud. Finally, photometric image-based approaches from multi-view stereo exploit the whole information carried in the image and result in more detailed dense or semi-dense reconstructions; however they usually assume the scene to be static not being designed to cope with moving objects.

In this paper we propose a novel visibility consistent 3D photometric mapping algorithm to reconstruct a urban textured mesh relying on both images and a sparse, although accurate, 3D point clouds, coming from a lidar sensor. The algorithm we propose is robust to the presence of moving object and it is able to reconstruct a consistent textured mesh without them (see in Fig. 1).

In Section II we discuss works in the literature close to the proposed system. In Section III we describe the novel framework proposed in the paper, which relies on both lidar data and images; in Section IV we show the experimental results and in Section V we conclude the paper proposing out some future direction of research.

II. RELATED WORK

In this paper we present a complete framework to estimate a texturized triangular mesh map of the environment in which moving object are explicitly removed from the geometry of the scene from the photometric refinement process and from the texture. We now give an overview of the various topic involved in the design of this framework

[1]Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Milano, Italy

`andrea.romanoni@polimi.it` (corresponding author)
`daniele.fiorenti@mail.polimi.it`
`matteo.matteucci@polimi.it`

Mapping from laser sensors is a well studied research area in Robotics; in the early studies, the map has been estimated in 2 dimensions [8], while, in recent years, the prevalent approach is to estimate it in 3D thanks to advances in algorithms, processing and sensors. Mapping can be pursued together with robot self-localization leading to Simultaneous Localization and Mapping systems; these algorithms do not focus on the mapping part, indeed they reconstruct a sparse point-based map of the environment, while in our case we aim at reconstructing a dense representation of it.

Some approaches estimate a 2.5D map of the environment by populating a grid on the ground plane with the corresponding cell heights [9]. These maps are useful for robot navigation, but neglect most of the environment details. A more coherent representation of the scene is volumetric, i.e, the space is partitioned into small parts classified as *occupied*, *free* and, in some cases, *unknown*, and the boundary between occupied and free space represents the 3D map. In laser-based mapping the most common volumetric representation is voxel-based due to its good trade-off between expressiveness and easiness of implementation [10]; the drawback of this representation is the large memory consumption, and, therefore its non-scalability. Many efforts have been directed to improve the scalability and accuracy of voxel based mapping. Ryde and Hu [11] store only occupied voxels, while Dryanovski *et al.* [12] store both occupied and free voxels, in order to represent also the uncertainty of unknown space. The state-of-the-art system OctoMap [3], and its extension [4], are able to efficiently store large maps by including an octree indexing to add flexibility to the framework.

Voxel-based approaches usually produce unappealing reconstructions, due to the voxelization of the space, and they need a very high resolution to capture fine details of the scene, trading off their efficiency. In Computer Vision community, different volumetric representations have been explored, in particular many algorithms adopt the 3D Delaunay triangulation [13], [5], [6], [14]. Delaunay triangulation is self-adaptive according to the density of the data, i.e., the points, without any indexing policy; moreover its structure is made up of tetraedra from which it is easy to extract a triangular mesh, widely used in the Computer Graphics community to accurately model objects. These algorithms are consistent with the visibility, i.e., they mark the tetrahedra as free space or occupied according to the camera-to-point rays, assuming that a tetrahedron is empty if one, or at least one, ray intersects them.

Among image-based dense photoconsistent algorithms, the mesh-based algorithm [14], [15] have been proven to estimate very accurate models and to be scalable in large-scale environments. They bootstrap form an initial mesh with a volumetric method such as [5] or [16] and they refine it by minimizing a photometric energy function defined over the images. The most relevant drawback happens when moving objects appear in the images: their pixels affect the refinement process leading to inaccurate results.

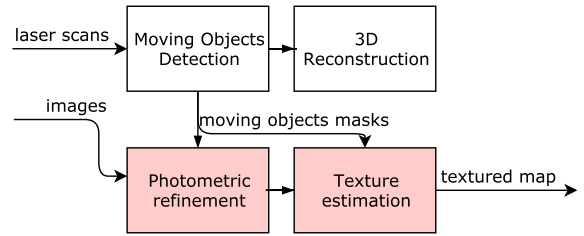In our paper, in order to filter out moving objects from

Fig. 2. Textured mesh pipeline. In this paper we focused on the two red boxes.

the lidar data and the images, we need to explicitly detect them. A laser-based moving objects detection algorithm has been proposed by Petrovskaya and Thrun [17] to detect a moving vehicles using model-based vehicle fitting algorithm; the method performs well, but it needs models for the objects. Xiao *et al.* [18] and the Vallet *et al.* [19] model the physical scanning mechanism of lidar using Dempster-Shafer Theory (DST), evaluating the occupancy of a scan and comparing the consistency among scans. A further improvement of these algorithms has been proposed by Postica *et al.* [20] where the authors include an image-based validation step which sorts out many false positive. Pure image-based moving objects detection has been investigated in static camera videos (see [21]), also for the jittering case [22], however it is still a very open problem when dealing with moving cameras.

Once moving points have been removed and the photo-consistent map is estimated, a texture can be computed from the images. Computer Graphics literature investigated many texuring algorithms [23], [24], [25], [26] which however suppose the model to be very accurate. In our application we estimate a realistic model useful for navigation purposes, but it still does not capture the fine details of the scene, which are required from these algorithms to work properly. Especially the resolution is not comparable to the resolution the Computer Graphics algorithm are used to deal with.

III. Textured Mesh reconstruction

In Fig. 2 we depict the whole pipeline of our system, from the laser and camera data, to the final textured reconstruction. After an initial preprocessing of laser data, we first detect the moving object, and, from these points, we create a moving object mask corresponding to the images captured by the cameras; then, we estimate a 3D mesh and we refine it leveraging on the information carried by the images and the moving objects masks. Finally, we texturize the mesh with a novel efficient algorithm robust with respect to low resolution meshes. The camera setting we considered is the monocular one. The steps of the pipeline are described in the following.

A. Moving Objects Detection and Removal

In the first step of the proposed system we detect the moving objects with the approach proposed in [20]. Here we briefly describe how it works, but for an in-depth description we refer the readers to the original paper [20]. We bootstrap from an initial point cloud of the scene, i.e., the first laser scan; as a new scan arrives, we align it to the existing
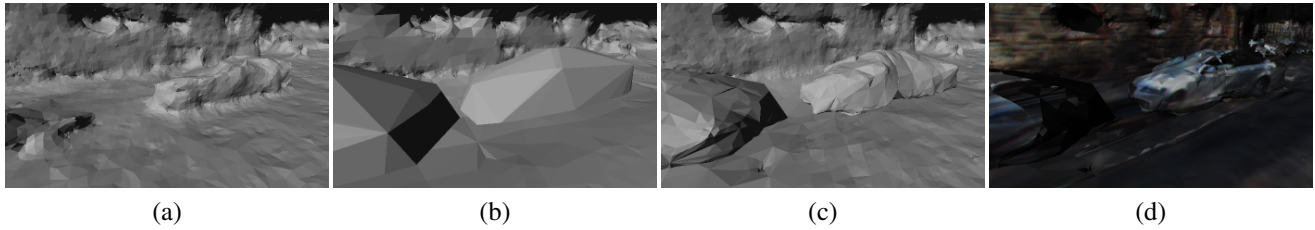
Fig. 3. Reconstruction of Lidar data: (a) with the visibility consistency [6]; (b) with the car detection; (c) with the car detection and refinement and (d) after texturing

point cloud through the Generalized Iterative Closest Point (GICP) [27] algorithm and we neglect points too faraway from the sensor, i.e., whose distance from the point cloud center is greater than a given threshold $\tau = 30m$. To avoid a redundancies we downsample the point cloud by two orders of magnitude; then we estimate and remove ground points. To this extend we model the ground as a grid and we classify points as ground by bootstrapping from the tile occupied by the laser sensor and by applying belief propagation on the grid; the detailed procedure is reported in [20].

As soon as a set of aligned and filtered scans is available, we detect moving objects as follows. Given a point in the space we define a visibility-based rule to estimate its occupancy with respect to a generic laser beam as $\{empty\}, \{occupied\}, \{unknown\}$, then, for each point P belonging to scan S_k we aggregate evidences for each beam of a scan S_i through Dempster-Shafer Theory (DST) and we classify the point as $\{empty\}$ or $\{occupied\}$.

Then, we classify a point P belonging to a scan S_k as static or moving by comparing its occupancy values with previous and future scans, and computing the probability of a conflicting or consistent state according to DST: if a conflict among states is detected, the corresponding point is classified as moving, otherwise as static.

B. Laser-based 3D Mapping

Once the points belonging to moving objects have been removed, we build a map of the environment. The 3D laser data and sensor-to-point visibility rays perfectly fits the visibility-consistent approach presented in [6] applied to image-based Structure from Motion (SfM) data; moreover, the algorithm ensures the manifold property of the reconstructed mesh which is needed for the photometric refinement. However, in real applications, laser beams pass through transparent surfaces, while the SfM points natively adopted by visibility consistent algorithms such as [28] or [6], do not reconstruct what is behind windows. In general this does not represent a big issue, but in the presence of cars, laser-based reconstruction is not able to capture adequately the geometry of the scene: rays traverse the interior of the car from different points of view and a visibility consistent reconstruction carves almost all the occupied space, leaving only the lower part of the car in the reconstruction. To avoid this undesired behavior, we propose to detect the cars in the point cloud and replace their points with a 3D model, e.g., Fig. 3.

In our system we implemented a simple hand-crafted car detector being interested here in the joint textured reconstruction from laser and image data. We discretize and project the 3D points on the 3D ground plane, then we cluster the set of points which have a rectangular shape. Finally, we check if a cluster projected along the longest dimension of the rectangle has a silhouette similar to a car (see the Appendix for a complete explanation). Learning based approaches could be exploited, for instance, Russel *et al.* [29] or Visin *et al.* [30].

Once the points belonging to cars have been detected, we remove them from the point cloud of the whole scene, and we group them into a set of clusters:

$$Cars = \{c_1, \cdots, c_i, \cdots, c_{N_{cars}}\}, \tag{1}$$

where c_i represents a single cluster, i.e., a car.

To obtain a consistent reconstruction we also recover the ground points removed for the moving object detection and using the resulting point cloud, the sensor-to-point rays, and the position of the lidar in metric coordinates after each GICP registration, we are able to apply the visibility consistent mesh reconstruction algorithm presented in [6]. This is a space carving-based method which partitions the space into tetrahedra, and classifies each tetrahedron as free space or matter according to the visibility rays; the boundary between the two classes is the resulting mesh.

In this context we do not need to map the environment incrementally as in [6]; therefore, we apply the batch version of the algorithm, i.e., we first add every laser center, 3D point and visibility ray, then we estimate the mesh.

The 3D reconstruction does not contain the moving points and the cars we removed previously. Since we aim at mapping the static part of the scene, we might consider parked cars as a part of it; these cars are then integrated into the final reconstruction. Each cluster of points $c_i \in Cars$ represents a car; and to include it into the model of the scene, we first compute its 3D convex hull then we add it to the estimated 3D map. In Fig. 3 we show how the car detection affects the final reconstruction.

C. Photometric Refinement without moving objects

The removal of points belonging to cars and the explicit inclusion of their convex hull, allows a more consistent reconstruction and it overcomes the issue related to the transparency of the car windows. Now, mapped cars have no more holes, and they are represented more coherently by a convex shape; of course the convex hull is not able to

Fig. 4. Red regions corresponds to the moving objects

capture all the details, but it is close enough to the real scene to allow for the photometric refinement.

To apply the refinement step, we estimate the mask of the moving objects starting from the detection performed with the algorithm of Postica et al. [20]. To compute the mask corresponding to the i-th camera, we project the lidar points detected as moving into the i-th image. Since the resulting mask is very sparse, we first filter it with an all-1 11x11 kernel that grows the detected regions in the neighborhood; then we refine the results by applying a 10px radius disk dilation and a 7px radius disk erosion. In Fig. 4 we show an example mask in red overlayed with the original image.

The refinement step described extends the ideas presented in [14] to cope with moving objects. The goal is to minimize the energy:

$$E = E_{photo} + E_{smooth}, \qquad (2)$$

where E_{photo} is the data term related to the image photo-consistency measure, and E_{smooth} is a smoothness prior.

To define the energy E_{photo} let consider two images I and J, and the triangular mesh S. Let x and $\overrightarrow{n}$ be a point on this mesh and the corresponding normal, and $err_{I,J}(x)$ a function that decreases if the similarity between the patch around the projection of x in J and I increases, then:

$$E_{photo} = \sum_{i,j} \int_{\Omega^S_{i,j}} err_{I,I^S_{ij}}(x_i) dx_i, \qquad (3)$$

where I^S_{ij} is the reprojection of the image from the j-th camera in the image I through the mesh S and $\Omega_{i,j}$ represents the domain of the mesh where the projection is defined. We minimize Eq. (3) through gradient descent by moving each vertex $X_i \in \mathbb{R}^3$ of the mesh according to the gradient:

$$\frac{dE(S)}{dX_i} = \int_S \phi_i(x) \nabla E_{photo}(x) dx, =$$
$$= -\sum_{i,j} \int_{\Omega^S_{i,j}} \phi_i(x) f_{ij}(x_i)/(\overrightarrow{n}^T \mathbf{d}_i) \overrightarrow{n} dx_i, \qquad (4)$$

$$f_{ij}(x_i) = \partial_2 err_{I,I^S_{ij}}(x_i) DI_j(x_j) D\Pi_j(x) \mathbf{d}_i, \qquad (5)$$

where $\phi_i(x)$ represents the barycentric coordinates if x is in the triangle containing X_i, otherwise $\phi_i(x) = 0$; Π_j is the j-th camera projection, the vector $\mathbf{d}_i$ goes from camera i to point x, the operator D represents the derivative and $\partial_2 err_{I,I^S_{ij}}(x_i)$ is the derivative of the similarity measure $err_{ij}(x)$ with respect to the second image.

We modify the previous equation in order to discard the moving objects. Let m_i and m^S_{ij} the mask of moving objects

in the i-th camera and the mask of the moving objects in the j-th camera projected through S in the camera i, we define $\Lambda = \Omega^S_{i,j} \cap m_i \cap m^S_{ij}$.

$$\frac{dE(S)}{dX_i} = -\sum_{i,j} \int_{\Lambda} \phi_i(x) f_{ij}(x_i) \overrightarrow{n}/z^3_i \frac{z^3_i}{\overrightarrow{n}^T \mathbf{d}_i} \overrightarrow{n} dx_i \qquad (6)$$

.

We minimize the energy E_{smooth} as in [14] by means of the Laplace-Beltrami operator approximated with the umbrella operator [31], which moves each vertex in the mean position of its neighbors.

D. Mesh texturing

After the refined map is available, we run the texturing process. Differently from the existing Computer Graphics texturing algorithms, in the proposed method, we want to be able to estimate incrementally the texture, such that it is possible to color the mesh while the images are acquired by the robot. Moreover we avoid to texture the map with the moving objects by discarding the pixels corresponding to the moving object.

The idea behind our method is to sum the color contributions coming from different images according to a weight given by the perpendicularity of the viewing ray from the camera to the point on the surface. Given a point $\mathbf{x} \in S$ belonging to the surface of the model, $\overrightarrow{n}$ the corresponding normal, and the camera center $\mathbf{c}_j$ we define:

$$w_j(\mathbf{x}) = (\overrightarrow{\mathbf{x} - \mathbf{c}_j}) \cdot \overrightarrow{n} = cos\theta \qquad (7)$$

, where θ is the angle between the normal and the camera to point direction. Let $c_k(\mathbf{x})$ be the color of the point $\mathbf{x}$ at frame k, equal to 0 if $\mathbf{x}$ is not visible; at frame $n+1$ we estimate the color of the texture at location $\mathbf{x}$ as:

$$C_{n+1}(\mathbf{x}) = \frac{W_n(\mathbf{x}) \cdot C_n(\mathbf{x}) + w_{n+1}(\mathbf{x})^\alpha \cdot c_{n+1}(\mathbf{x})}{W_n(\mathbf{x}) + w_{n+1}(\mathbf{x})^\alpha} \qquad (8)$$

where

$$W_n(\mathbf{x}) = \sum_{i=1}^{n} w_i(\mathbf{x})^\alpha, \qquad C_1(\mathbf{x}) = c_1(\mathbf{x}) \qquad (9)$$

We raise the weight to the exponent α to increase the importance of weighting, i.e., to increase the importance of the contributions of the pixels which are more perpendicular to the surface (by experimental evaluation we fixed $\alpha = 8$).

TABLE I
RESULTS ON KITTI SEQUENCES: MAP ERROR

	seq 0095		seq 0104	
	avg	std	avg	std
Romanoni *et al.* [6]	0.089	0.131	0.194	0.311
after refinement	**0.082**	**0.098**	**0.082**	**0.103**

Fig. 5. Ball Pivoting reconstruction: dark regions are caused by non consistent facet normals

(a) Without cars detection

(c) With cars convex hulls

(d) After refinement

(d) Refinement and texturing

(e) corresponding frame

Fig. 6. Results on frame 16. Let notice that moving objects (the people by bicycle) do not affect the reconstruction

IV. Experimental Results

We tested our approach against the publicly available KITTI dataset [32]; in particular we used the sequences 0095 and 0104, captured by a Velodyne 64HD with respectively 268 and 313 1392x512 gray scale frames. The algorithm runs on a 4 Core i7-2630QM CPU at 2.2Ghz (6M Cache), with 6GB of DDR3 SDRAM and NVIDIA GeForce GT 630M.

A. Mapping

To provide a quantitative evaluation of the reconstructed mesh, we compare it against the full point cloud, i.e., without the downsampling needed for moving object detection (see Section III-A). Lidar data are dense and accurate enough to be considered as ground truth at least locally. We removed from the full point cloud both the moving point and the interior of the cars we do not want to map. The mesh to point cloud comparison was computed by the tool CloudCompare [33] which averages the distances from each point of the ground truth, to the nearest triangle in the estimated mesh.

We compare our algorithm against different approaches applied to the same downsampled point cloud to provide a fair evaluation: the method proposed in [6]; two widespread algorithms for mesh reconstruction from point clouds (i.e., Poisson Reconstruction [34] and Ball Pivoting [35]); and OctoMap [3], i.e., state-of-the-art laser-based mapping algorithm.

In Table I we show the result of our comparison with [6]. The average errors are below 0.1 m which is enough accurate for a wide variety of robotics tasks, such as localization and navigation. The proposed algorithm improves the accuracy of [6], and both car detection and photometric refinement have contributed to this enhancement; since the number of cars in the 0104 sequence is greater than those in sequence 0095, the improvement is more evident in the former case.

Poisson Reconstruction was not able to produce a proper reconstruction due to the sparsity of the downsampled laser data; for the same reason Ball Pivoting produces big holes in the reconstruction and Octomap was not able to recover a dense structure. In addition to holes, Ball Pivoting reconstructs a mesh whose normals are not consistent and contains severe self intersections (see Fig. 5). Moreover, while the appearance of the results of the proposed algorithm, of [6] and, to some extent, of the Ball Pivoting are realistic,
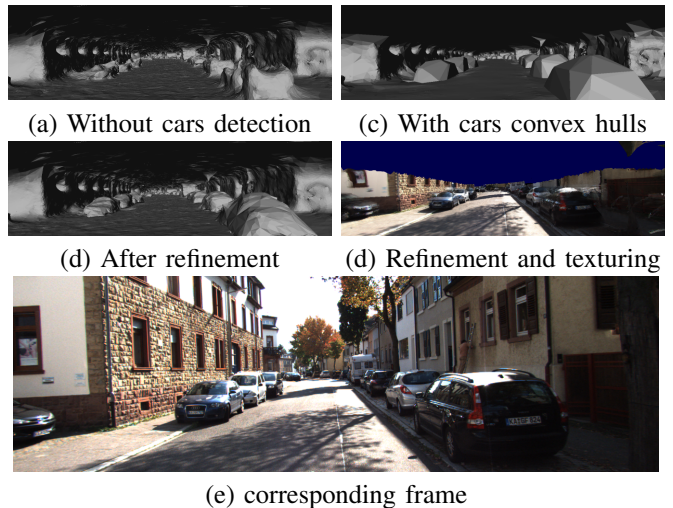
Octomap reconstructs a voxelized map in which many details are lost (see Fig. 7).

The algorithm reconstructed and refined the 0095 (268 frames) sequence in 73 minutes and the sequence 0104 (313 frames) in 80 minutes.

Since we rely the reconstruction on laser scans which have been georeferenced, the final map is in turn georeferenced and could be useful for classical robotic localization or navigation tasks.

B. Texturing

We tested our texturing method against the widely adopted texturing approaches proposed by the Computer Graphic community, i.e., Mask Photo Blending algorithm proposed in [36]. Here we evaluate the results by visual inspection since no quantitative evaluation is possible. We observed that our method, applied to a typical Computer Graphics dataset, achieves results comparable with the Mask Photo Blending algorithm, even if it is aimed to texturize lower resolute and less accurate meshes. In our scenario the proposed texturing algorithm obtains much better results with respect the Mask Photo Blending: it keeps the color continuity among neighboring facets. In Fig. 8 and Fig. 9 we show two examples of the comparison between the two approaches.

The algorithm texturized the sequence 0095 (268 frames) in 0.95187 seconds per frame and the sequence 0104 in 1.0374 seconds per frame.

V. Conclusions and Future Work

In this paper we proposed for the first time a complete framework to extract a 3D textured map of the environment by exploiting both accuracy of the lased data and the dense appearance captured by images. We were able to recover a continuous 3D map consistent with the visibility of the laser beams, and refine it by relying on the images. We explicitly removed moving objects from the laser-data and we detected and modeled independently the cars, which windows are usually traversed by laser beams. Finally, we textured the
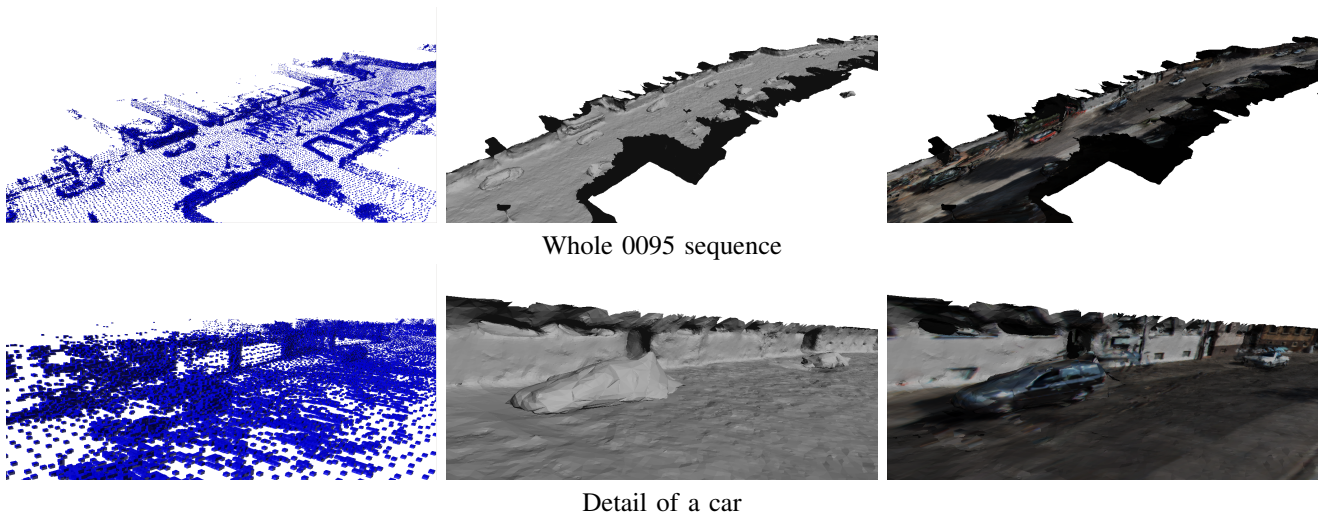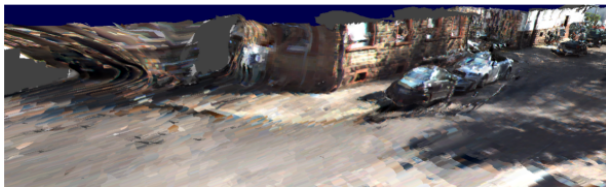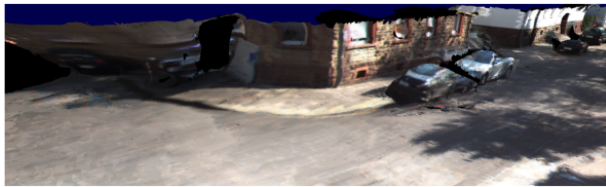
Whole 0095 sequence

Detail of a car

Fig. 7. Reconstruction of 0095 sequence with Octomap (first column) and with the proposed approach with and withoud texture (second and third columns)

Masked Photo Blending [36]

Proposed texturing

Fig. 8. Texturing results

Masked Photo Blending [36]

Proposed texturing

Fig. 9. Texturing results

map: both refinement and texturing have not been affected by the presence of moving objects, since we infer their image positions from the laser-based data.

In the future we are interested in improving the convergence of the refinement algorithm adapting the method presented in [16] and we would exploit the semantics of the scene, e.g., as extracted by deep learning segmentation proposed in [30], to further improve the reconstruction. We could extend the method presented here to recover the cars to multiple class of objects, e.g., trees and poles. Moreover, we could increase the accuracy of the texturing, with an ad-hoc method which exploits the photometric scores to properly weight the per-triangle texturing.

REFERENCES

[1] A. Shashua. Autonomous driving, computer vision and machine learning (kenote). Computer Vision and Pattern Recognition (CVPR). [Online]. Available: https://www.youtube.com/watch?v=n8T7A3wqH3Q

[2] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3d lidar and dense stereo," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 2181–2188.

[3] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[4] S. Khan, D. Wollherr, and M. Buss, "Adaptive rectangular cuboids for 3d mapping," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2132–2139.

[5] V. Litvinov and M. Lhuillier, "Incremental solid modeling from sparse structure-from-motion data with improved visual artifacts removal," in *International Conference on Pattern Recognition (ICPR)*, 2014.

[6] A. Romanoni and M. Matteucci, "Incremental reconstruction of urban environments by edge-points delaunay triangulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 4473–4479.

[7] C. Wu, "Towards linear-time incremental structure from motion," in *3D Vision-3DV 2013, 2013 International Conference on*. IEEE, 2013, pp. 127–134.

[8] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 2432–2437.

[9] M. Herbert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. IEEE, 1989, pp. 997–1002.

[10] H. P. Moravec, "Robot spatial perceptionby stereoscopic vision and 3d evidence grids," *Perception*, 1996.

[11] J. Ryde and H. Hu, "3d mapping with multi-resolution occupied voxel lists," *Autonomous Robots*, vol. 28, no. 2, pp. 169–185, 2010.

[12] I. Dryanovski, W. Morris, and J. Xiao, "Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1553–1559.

[13] A. Romanoni and M. Matteucci, "Efficient moving point handling for incremental 3d manifold reconstruction," in *International Conference on Image Analysis and Processing (ICIAP)*. Springer, 2015, pp. 489–499.

[14] H. H. Vu, P. Labatut, J.-P. Pons, and R. Keriven, "High accuracy and visibility-consistent dense multiview stereo," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 5, pp. 889–901, 2012.

[15] Z. Li, K. Wang, W. Zuo, D. Meng, and L. Zhang, "Detail-preserving and content-aware variational multi-view stereo reconstruction," *arXiv preprint arXiv:1505.00389*, 2015.

[16] A. Romanoni, A. Delaunoy, M. Pollefeys, and M. Matteucci, "Automatic 3d reconstruction of manifold meshes via delaunay triangulation and mesh sweeping," in *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016.

[17] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.

[18] W. Xiao, B. Vallet, and N. Paparoditis, "Change detection in 3d point clouds acquired by a mobile mapping system," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, no. 2, pp. 331–336, 2013.

[19] B. Vallet, W. Xiao, and M. Brédif, "Extracting mobile objects in images using a velodyne lidar point cloud," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, pp. 247–253, 2015.

[20] G. Postica, A. Romanoni, and M. Matteucci, "Robust moving objects detection in lidar data exploiting visual cues," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016.

[21] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, 2014.

[22] A. Romanoni, M. Matteucci, and D. G. Sorrenti, "Background subtraction by combining temporal and spatio-temporal histograms in the presence of camera movement," *Machine Vision and Applications*, vol. 25, no. 6, pp. 1573–1584, 2014.

[23] M. Callieri, P. Cignoni, and R. Scopigno, "Reconstructing textured meshes from multiple range rgb maps." in *VMV*, 2002, pp. 419–426.

[24] Y. Alj, G. Boisson, P. Bordes, M. Pressigout, and L. Morin, "Multi-texturing 3d models: How to choose the best texture?" in *3D Imaging (IC3D), 2012 International Conference on*. IEEE, 2012, pp. 1–8.

[25] M. Waechter, N. Moehrle, and M. Goesele, "Let there be color! large-scale texturing of 3d reconstructions," in *European Conference on Computer Vision*. Springer, 2014, pp. 836–850.

[26] I. Garcia-Dorado, I. Demir, and D. G. Aliaga, "Automatic urban modeling using volumetric reconstruction with surface graph cuts," *Computers & Graphics*, vol. 37, no. 7, pp. 896–910, 2013.

[27] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Robotics: Science and Systems*, vol. 2, no. 4, 2009.

[28] V. Litvinov and M. Lhuillier, "Incremental solid modeling from sparse and omnidirectional structure-from-motion data," in *BMVC*, 2013.

[29] C. Russell, P. Kohli, P. H. Torr, *et al.*, "Associative hierarchical crfs for object class image segmentation," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 739–746.

[30] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville, "Reseg: A recurrent neural network-based model for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 41–48.

[31] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun, "Discrete laplace operators: no free lunch," in *Symposium on Geometry processing*, 2007, pp. 33–37.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[33] D. Girardeau-Montaut, "Cloud compare, (last access mar, 22 2015)." [Online]. Available: http://www.cloudcompare.org/

[34] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[35] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE transactions on visualization and computer graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[36] M. Callieri, P. Cignoni, M. Corsini, and R. Scopigno, "Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3d models," *Computers & Graphics*, vol. 32, no. 4, pp. 464–473, 2008.

APPENDIX

In this appendix we give the details of the car detection algorithm adopted after the reconstruction step. First, we project the point cloud on the ground plane and we aggregate the points according to a 2D grid on the *XY* plane similarly to what we do in the ground removal step; the cell dimension is 0.1x0.1m. Then we empty the cells containing points higher than a threshold $\tau = 2.2m$ to neglect most of the walls and trees, very common in urban environments. We treat the resulting grid as an image and we apply the closure morphological operator, to close small holes and gaps and to filter out isolated points. We finally extract connected cells with at least one point and we keep only the regions whose bounding box has a shape compatible with a car, as detected by a template. Let l_{BB} and w_{BB} be respectively the length and width of the bounding box around a connected set of cells, and ρ_{BB} the radius circumscribing the box; we filter out all the regions which do not satisfy: $\hat{\rho}_{min} < \rho_{BB} < \hat{\rho}_{max}$ and $\hat{r}_{min} < \frac{w_{BB}}{l_{BB}} < \hat{r}_{max}$, where, in our case, we take into account the average dimensions of the cars and we choose $\hat{\rho}_{min} = 1.5m$, $\hat{\rho}_{max} = 5.5m$, $\hat{r}_{min} = \frac{1.2}{5.0}$ $\hat{r}_{max} = \frac{3.5}{5.0}$.

As a further filtering, we take into account the *Z* dimension that the previous projection has neglected. For each 2D bounding box we project the points along the direction parallel to the longest dimension between l_{BB} and w_{BB} in a discrete grid, which is again composed by 0.1mx0.1m cells. We compute the convex hull of the projected points and we treat the result as an image. For each column we collect the number of white pixels in a vector γ; we compute the discrete derivative of γ; then, we compute a three bin histogram of it. The group of points is classified as car if the first and third bin represent respectively an increasing and a decreasing ramps of at least $\frac{\pi}{6}$ and the second bin is almost flat (at most $\frac{\pi}{3}$). The method runs in 0.19s for a sequence of around 300m, almost all the cars are detected, moreover, even if some false negative exists, i.e., some cars are not detected, they are going to be reconstructed anyway, at least partially.