MOVING OBJECTS AWARE SENSOR MESH FUSION FOR INDOOR RECONSTRUCTION FROM A COUPLE OF 2D LIDAR SCANS

Teng Wu[1,*], Bruno Vallet[1], Cédric Demonceaux[2], Jingbin Liu[3,4]

[1] LASTIG, Univ Gustave Eiffel, ENSG, IGN, F-94160 Saint-Mandé, France - firstname.lastname@ign.fr
[2] VIBOT ERL CNRS 6000, ImViA Université Bourgogne Franche-Comté, France - cedric.demonceaux@u-bourgogne.fr
[3] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China - jingbin.liu@whu.edu.cn
[4] Department of Remote Sensing and Photogrammetry and the Center of Excellence in Laser Scanning Research, Finnish Geospatial Research Institute, 02430 Masala, Finland

Commission II, WG II/4, II/5

KEY WORDS: Lidar processing, indoor reconstruction, visibility analysis, change detection, moving objects detection, mesh fusion, mesh mosaic

ABSTRACT:

Indoor mapping attracts more attention with the development of 2D and 3D camera and Lidar sensor. Lidar systems can provide a very high resolution and accurate point cloud. When aiming to reconstruct the static part of the scene, moving objects should be detected and removed which can prove challenging. This paper proposes a generic method to merge meshes produced from Lidar data that allows to tackle the issues of moving objects removal and static scene reconstruction at once. The method is adapted to a platform collecting point cloud from two Lidar sensors with different scan direction, which will result in different quality. Firstly, a mesh is efficiently produced from each sensor by exploiting its natural topology. Secondly, a visibility analysis is performed to handle occlusions (due to varying viewpoints) and remove moving objects. Then, a boolean optimization allows to select which triangles should be removed from each mesh. Finally, a stitching method is used to connect the selected mesh pieces. Our method is demonstrated on a Navvis M3 (2D laser ranger system) dataset and compared with Poisson and Delaunay based reconstruction methods.

1. INTRODUCTION

3D reconstruction from images (Schops et al., 2017) and Lidar (Berger et al., 2017) is a widely researched topic in the photogrammetry and computer vision communities. With the development of sensor devices, 3D reconstruction is widely applied to various scenes, both outdoors (Musialski et al., 2013) and indoors (Huitl et al., 2012). During the data collection, there may exist moving objects in the scene (people inside, pedestrian and vehicles outside). For robot applications, these moving objects should be detected to adapt the behaviour of the robot to its surrounding for scene mapping. For 3D reconstruction purposes, these objects should be removed in order for the 3D model to represent only the static part of the scene (Jiang et al., 2017a). In both cases, motion analysis is a mandatory preprocessing step.

1.1 Previous Works

For indoor static scene reconstruction, the related work can be categorized into two research areas: moving objects analysis and 3D reconstruction. Moving objects analysis can be based on images(Tron, Vidal, 2007) or Lidar (Schauer, Nüchter, 2018) points clouds. In this paper, we emphasize on Lidar based indoor reconstruction, so we focus our state of the art on Lidar based methods.

1.1.1 Moving Objects Analysis The methods can be divided into two groups:

(1) **Motion flow** relies on ICP (Iterated Closet Point) using point correspondence to analyze the velocity of moving objects (Pomerleau et al., 2014). Simultaneous localization and mapping with moving object tracking method is mainly used in robot scanning (Wang, Thorpe, 2002). A 3D flow field is computed to analyze the motions during ICP (Jiang et al., 2017b), and dynamic objects are detected by flow clustering.

(2) **Visibility analysis** (Underwood et al., 2013) uses the sensor information to remove the objects that are volumetrically inconsistent between scans (objects from a scan traversed by rays from another scan). The input are two point clouds acquired at different time. The Dempster-Shafer theory can be used to improve the combination of volumetric information from the scans (Xiao et al., 2015). Several point based data structures such as Voxel (Andreasson et al., 2007; Schauer, Nüchter, 2018) and OctoMap (Gehrung et al., 2019) have been proposed to improve the performance of the ray tracing method.

Our method falls in the second category as it leverages visibility information to detect 3D volumetric changes. However, while the methods mentioned above are based on point clouds, which contains samples of the continuous surface of the scene, our method handles meshes, providing a continuous representation.

1.1.2 3D Reconstruction There are many 3D reconstruction methods for static scenes (Berger et al., 2017). They can be categorised as:

(1) **Implicit methods**: the reconstructed surface is represented as the zero set of a function defined in space, which

* Corresponding author

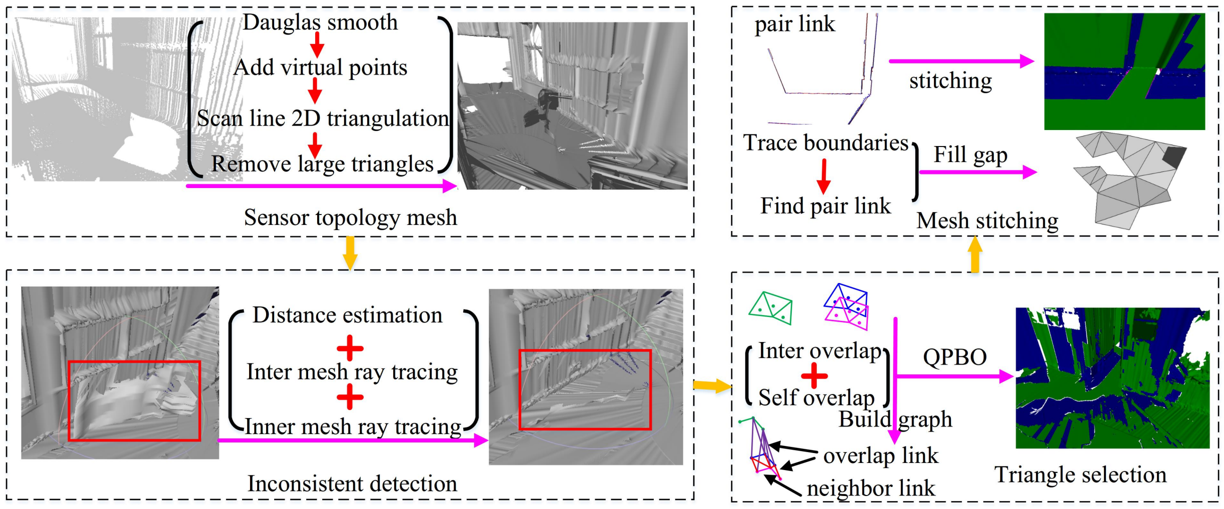


Figure 1. There are 4 steps in our reconstruction pipeline. **Step 1** (top left): generate a sensor mesh from each sensor, introducing in **Section 3.1** and **Section 3.2**; **Step 2** (bottom left): detect and remove the inconsistent objects based on a combination of distance and visibility, the detail is in **Section 3.3**; **Step 3** (bottom right): select triangles based on a boolean optimization framework, refer to **Section 3.4.1**; **Step 4**(top right): stitch the selected mesh pieces(cf **Section 3.4.2**).

values are positive outside and negative inside the solid scene. This guarantees the watertightness of the resulting surface. A triangle mesh discretizing this zero set can then be generated, usually using the Marching Cubes method (Lorensen, Cline, 1987). The most widely used is Poisson surface reconstruction (Kazhdan et al., 2006) which aligns the gradient of the function with normals computed from the point cloud. Truncated signed distance functions(TSDF) is another successful implicit method that processes RGB-D datasets (Newcombe et al., 2011). An elastic registration has been proposed to improve the performance of TSDF (Zhou et al., 2013). An extensive experiment of TSDF on multi-line Lidar point clouds has been proposed in (Roldão et al., 2019).

(2) **Explicit methods** usually use local information to estimate the surface and produce a watertight or non watertight surface (it might have boundaries). In (Ryde et al., 2013), the surface is approximated locally using a voxel-based plane detection. (Labatut et al., 2009) uses a Graphcut optimization framework to find the surface in a 3D Delaunay triangulation which is robust to noise. (Marton et al., 2009) proposed a greedy surface triangulation algorithm relying on normal information that also focuses on robustness to noise. Finally, *sensor meshing* (Boussaha et al., 2018) is a simple and fast method that makes use of the sensor topology to build triangles connecting consecutive points in each scanline, and corresponding points in consecutive scanlines.

1.2 Overview

In this paper, we propose a static mesh reconstruction method considering moving objects based on merging two scans collected at same time period from two 2D laser rangers. Sensor meshes (Boussaha et al., 2018) are generated from the two scans, with the difference that we store the optical center positions for each mesh vertex which will be mandatory for the subsequent visibility analysis. This visibility analysis detects moving objects as volumetric inconsistencies by ray tracing both within the same scan and between the two scans. A boolean optimization produces a mosaic of the two meshes by maximizing surface coverage and minimizing seam line length while forbidding overlaps. Finally, the resulting mesh pieces are stitched.

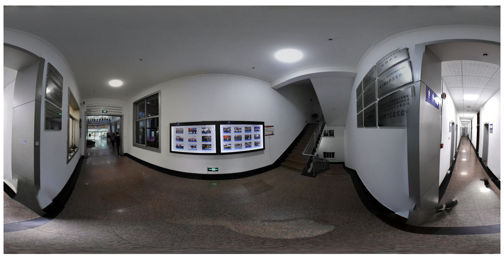


Figure 2. Panoramic image of the experimentation scene.

The paper is structured as follows: Section 2 presents the dataset and scanned area. Details of the method are presented in Section 3. Results and analysis are provided in Section 4. Finally, conclusions are drawn and perspectives proposed in Section 5.

2. DATASET

The dataset used in this paper is acquired with a Navvis M3 trolley (Marcus, Georg, 2017), which is mainly for indoor mobile mapping. This platform integrates an IMU, three HOKUYO UST-10LX 2D Lidar rangers and six cameras allowing to create panoramic images of the surrounding scene as illustrated on Figure 2. One of the laser rangers is mounted horizontally to allow for 2D Lidar based simultaneous localization and mapping(SLAM). The other two are mounted vertically, one on the left and the other on the right of the trolley so we will identify them as “Left” and “Right” Lidars respectively. For the scene shown in Figure 2, the point cloud is visualized in Figure 3.

The intrinsic parameters of the Lidar sensors are given in (Hokuyo, 2019). The intrinsic and relative extrinsic parameters are obtained from calibration, and pose parameters can be obtained from the Navvis system. As shown in Figure 4, the trajectory of the system is not a straight line, and it includes rotations in horizon plane during scanning, consequently some areas are scanned twice by the same Lidar. In the experiment, considering the computation speed and device memory, the dataset is divided into two blocks shown in Figure 4.

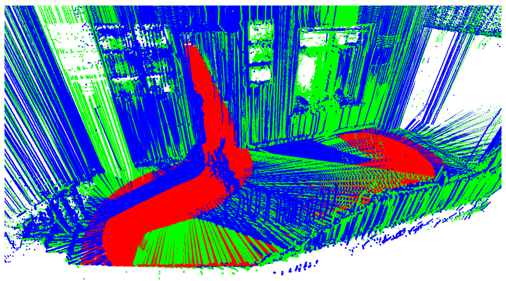


Figure 3. The point cloud of the experimentation scene. Points from **left** sensor are in 🟦, from **right** are in 🟩, and added **virtual** points are in 🟥.

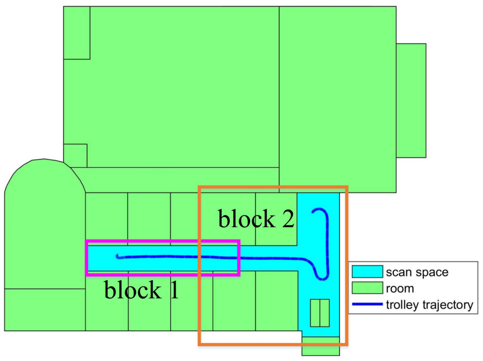


Figure 4. Trajectory of the trolley and blocks division.

3. METHOD

There are four steps in our pipeline shown in Figure 1: sensor mesh generation including pre-processing(cf Section 3.1) and mesh generation(cf Section 3.2), moving objects detection and removal introduced in Section 3.3, triangle selection using a boolean optimization, detail is in Section 3.4.1, and mesh pieces stitching is in Section 3.4.2 respectively.

3.1 Pre-Processing

The HOKUYO UST-10LX Lidar scans an angular sector of 270° such that it has a 90° blind angle, usually oriented towards the ground as illustrated in Figure 5. In order to fill this blind angle, some virtual points are added by intersecting lines sampling the blind angle and the ground plane. Considering the platform is mainly for indoor mobile mapping using 2D based SLAM(Marcus, Georg, 2017), the ground plane is the $z = 0$ plane for the platform. The distance of the lidar sensors to the plane and angle relative to the plane are fixed by the design of the trolley, the ground plane is at a constant position in the Lidar scans. The blind angle is divided into two parts symmetrically, only the last scan length r is near to the hypothetical length, points are added in the half of the blind sector, as the red points are added shown in Figure 3.

Then, in order to reduce the noise level, smooth filter is considered. Whole mesh based smooth filter maybe better than scan line based method, but scan line based smooth is fast. We rely on the prior that indoor environment often presents piecewise flat surfaces by using a Douglas-Peucker algorithm (Wu, Marquez, 2003) to smooth the scan line. Douglas-Peucker creates a polygonal approximation of the points along the scanline on which the input points are projected along their ray, as shown in Figure 6.

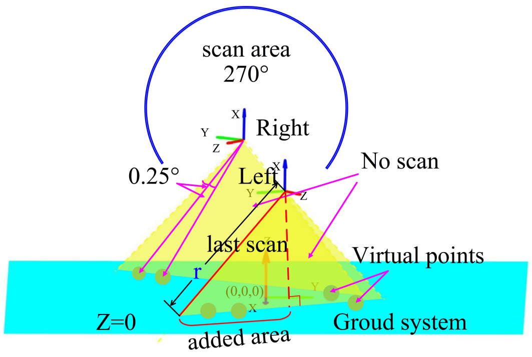


Figure 5. Blind angle of the Lidar (yellow) and virtual completion.

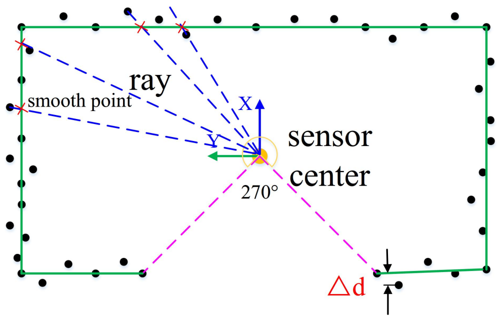


Figure 6. Douglas-Peucker smoothing: smoothed points (red) result from the projection of the input points (black) on a polygonal approximation of the scanline (green) for the laser scan.

3.2 Sensor Mesh

In planar Lidars such as the HOKUYO UST-10LX, the laser beam is directed at a rotating mirror such that the reflected beam rotates in a plane. Points are acquired in order at a constant sampling rate, such that each point have two natural neighbors, the points acquired just before and just after. Moreover, the angle of the beam can be accessed for each point, such that we can define four additional neighbors for each point: the points with the beam angle just over and under his own beam angle in the preceding and following lines. These 6 neighbors allow to create 6 triangles defining the sensor mesh, as explained in more details in (Boussaha et al., 2018). Elongated triangles appearing on objects borders (depth discontinuities) can be filtered out based on simple geometric rules such as maximum edge length or circumradius, such that the sensor mesh can have holes. Sensor mesh of block 2(cf Figure 4) is shown in Figure 7.

3.3 Moving Objects Analysis

Moving objects are present in the Lidar scans thus in the sensor mesh. We propose to detect them through a visibility analysis as in (Xiao et al., 2015) for instance. Because a ray indicate free space, if it intersects a sensor mesh triangle, either from the other mesh or from the same mesh but acquired at another time, then the intersected triangle belongs to a part of the scene that has changed, this is considered moving in this situation.

As shown in Figure 8, to make the inconsistent situations more clear, a virtually visibility analysis in 2D is performed. Figure 8(a) shows the two sensor meshes with visibility information. Note that $mesh_1$ has two different sight vectors meaning the scanner has scanned the same part of the scene twice

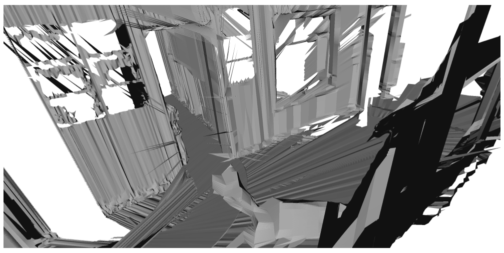
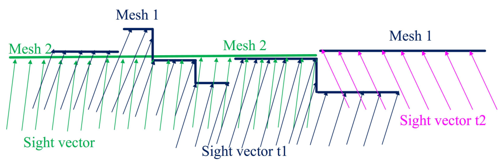
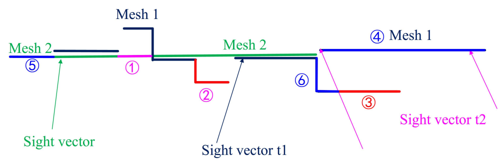


Figure 7. A sensor mesh from the left laser sensor. There are holes in specular reflection areas(glass and metal).



(a) Input of the visibility analysis: two sensor meshes (in dark blue and green) with visibility information (sight vectors = rays)



(b) Output of the visibility analysis: mesh parts tagged as inconsistent (in red for $mesh_1$ and magenta for $mesh_2$)

Figure 8. A 2D view (from above) of visibility analysis.

from two separate viewpoints, which special case called *self-intersection* is handled by our method.

Visibility analysis consists in tagging as inconsistent the triangles of each mesh that are traversed by a ray either from the other mesh (cross-intersections) or from the same mesh (self-intersections). Figure 8(b) illustrates the two types of inconsistent areas:

1. ① and ② are cross-intersections (triangles from $mesh_1$ intersected by rays from $mesh_2$)
2. ③ is a self-intersection of $mesh_1$

④, ⑤ and ⑥ are not inconsistent but they are considered to belong to the static part of the scene and will be the inputs of the following mesh fusion. Note that ⑥ may be also belong to a moving object as it is in the continuity of an inconsistent part, but we do not have enough information to validate this (no rays traverse it).

In our experiment, visibility analysis is performed by ray/triangle intersection in 3D mesh. As the number of intersections to compute is the product of the number of rays and of triangles which can get huge, it is accelerated by building a AABB tree structure in CGAL (The CGAL Project, 2018) for the triangles. In practice, when the same part of the scene is scanned multiple times, many intersections can happen due to noise and uncertainty on the trajectory which will generate many false alarms. Thus we combine the visibility analysis with the distance analysis: if two triangles are close enough (based on a distance threshold, can be from the accuracy of the platform) and that they overlap, they are considered consistent. In this case, the corresponding rays are not intersected, and the consistency information is stored for the next step.

3.4 3D Mesh Fusion

The objective of mesh fusion is to create a single mesh from all the remaining triangles. After removing inconsistent triangles and defining consistent ones, the remaining triangles can be in one of two cases:

1. *Single*: the triangle has no corresponding triangles, which means this part of the scene was seen only once, and by a single sensor.
2. *Redundant*: the triangle has corresponding triangles either in the other mesh, either in the same mesh (this part of the scene has been scanned more than once by the sensors)

While single triangles should trivially be kept in the merged mesh, some redundant triangles should be eliminated in order to keep a single mesh layer in all the parts of the scene. This is the aim of triangle selection.

3.4.1 Triangle Selection generalizes to 3D mesh the notion of 2D mosaicking for images. Once the images to mosaic are resampled in the same geometry, the remaining problem is to decide which pixels to keep in the overlaps. This is usually done by a labelling (one label per pixel in input images) optimization in a way that minimizes relief displacement and radiometric differences across seam lines (Lin et al., 2015). In 3D, the overlaps are defined by redundant triangles, and we will also aim at minimizing seam length, but it is not a grid labelling problem anymore as the sampling is different in the various meshes. Thus we pose the problem as a boolean optimization with the constraint that two overlapping triangles should not be kept together.

Considering two meshes : M_1 and M_2, we define the following:

- $x_i^j \in \{0,1\}$ is a boolean on each triangle T_j^i of mesh M_i, indicating if the triangle is selected (1) or removed (0) in the result. $\mathbf{x}$ is a vector concatenating all the x_i^j.
- $Q(T_j^i)$ is the quality of triangle T_j^i which choice will be discussed later.
- $\mathcal{C}$ is the set of all consistent pairs of triangles (from the same mesh or not), which means they are below the distance threshold and that they overlap, as computed in the previous step.
- $\mathcal{B}_i$ is the set of triangles of mesh M_i with at least one boundary edge. For a triangle $T_i^j \in \mathcal{B}_i$, we call $L(T_i^j)$ the length of its boundary edge(s).
- For two triangles T_{j1}^i, T_{j2}^i of the same mesh M_i sharing an edge, $L(T_{j1}^i, T_{j2}^i)$ is the length of their common edge.
- $XOR(x_1, x_2) = x_1 + x_2 - 2x_1.x_2$ the exclusive OR logical operator.

Then the triangle selection problem is defined as finding the minimum of:

$$\begin{aligned} E(\mathbf{x}) = & -\sum_{M_i}\sum_{T_j^i \in M_i} Q(T_j^i)x_i^j + P \sum_{(T_j^i, T_{j'}^{i'}) \in \mathcal{C}} x_i^j . x_{i'}^{j'} \\ & + \sum_{M_i}\sum_{T_j^i \in \mathcal{B}_i} L(T_j^i)x_i^j \\ & + \sum_{M_i}\sum_{(T_{j1}^i, T_{j2}^i) \in \mathcal{A}_i} L(T_{j1}^i, T_{j2}^i) XOR(x_i^{j1}, x_i^{j2}) \end{aligned} \tag{1}$$

1. The first term ensures that the result has as many input triangles as possible and encourages to select higher quality triangles when triangles overlap. Quality definition can combine resolution and noise/uncertainty metrics. In our case with similar noise levels, we focus on resolution with the very simple choice $Q(T_j^i)$ is square root of area size of the triangle, as higher resolution means more triangles for the same area of the scene, i.e. smaller triangle.

2. The second term forbids overlapping triangles to be selected (P is a very large constant). This is easier to implement and optimize than a strict constraint with the same result.

3. The last two terms measure boundary length as there are two types of boundaries in the output mesh: boundaries from the input for which the corresponding triangle is kept and edges between a triangle kept and a triangle removed.

Note that while quality should be maximized, the other terms (overlap and boundary length) should be minimized, explaining the signs. The result of this optimization consists in a small number of quite compact, non overlapping mesh *pieces* (connected components) as the boundary penalty discourages creating many pieces and pieces with complex boundaries, and overlaps are very highly penalized. These pieces are not connected so the resulting mesh have gaps that need to be filled, which we describe in the next section.

3.4.2 Seam Line Stitching is to create bands of triangles to connect the mesh *pieces* together to recover the continuous nature of the scene. It is performed in two steps:

(1) **match boundaries between pieces**: Thanks to the halfedge data structure in CGAL (The CGAL Project, 2018), the boundary edges of each mesh piece and their orientation can be accessed efficiently. The boundary matching is done in a greedy manner: we look for the closest pair of edges from separate pieces, then grow two boundaries starting at these edges while the two boundaries are close enough. This creates a first pair of matched piece boundaries. This process is then iterated on the remaining edges while the pair of closest edges is close enough. A k-d tree structure in CGAL is used to accelerate nearest edge search.

(2) **link the pairs of matched boundaries**: To connect the pairs of matched piece boundaries, we start by snapping the vertices that are close enough to the opposite boundary to the closest vertex. For the remaining unsnapped boundary, we create stitching triangles by filling the hole formed by the two matched boundaries connected at their endpoints by adapting a standard hole filling algorithm (Liepa, 2003). This dynamic programming algorithm needs a comparison operator "<" definition which is the best between two triangles. We found that the proposition of (Liepa, 2003) that minimizes the sum of triangle area and largest dihedral angle, produces elongated triangles, so we propose the following comparison operator integrating the perimeter of the triangle:

$$T_1 < T_2 \iff \begin{cases} \mu_1 < \mu_2 & \text{if } \mu_1 < \epsilon \text{ and } \mu_2 < \epsilon \\ \Omega_1 + \frac{1}{2}\left(\frac{C_1}{3}\right)^2 < \Omega_2 + \frac{1}{2}\left(\frac{C_2}{3}\right)^2 & \text{else} \end{cases} \tag{2}$$

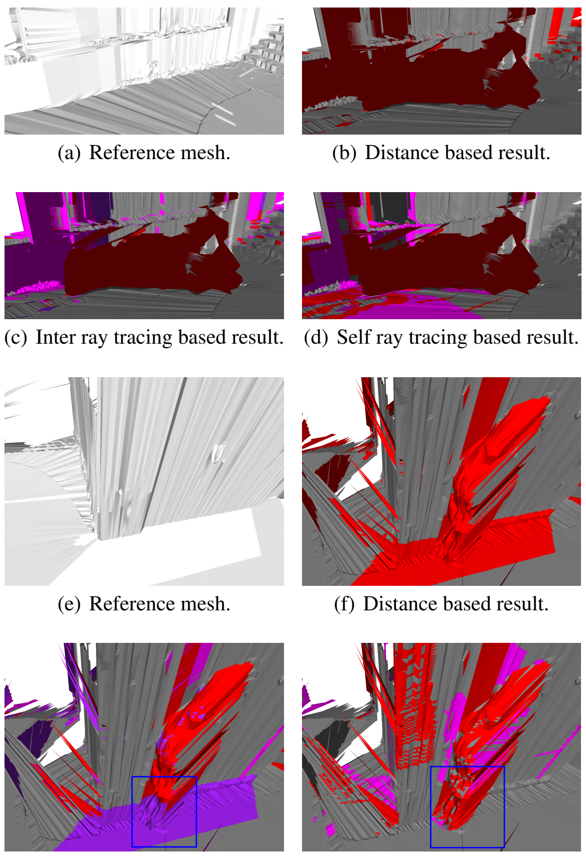

(a) Reference mesh. (b) Distance based result.
(c) Inter ray tracing based result. (d) Self ray tracing based result.
(e) Reference mesh. (f) Distance based result.
(g) Inter ray tracing based result.. (h) Self ray tracing based result.

Figure 9. Inconsistent object detection exhibit two types of dynamic object: (a),(e) is the reference mesh, (b),(c),(d) is a moving person, (f),(g),(h) is a shortly standing person. Color code: Inconsistent 🟥 , single 🟪 and triangles behind inconsistent in ■ .

where for the triangle T_i, μ_i is the maximum dihedral angle between T_i and its neighboring triangles, Ω_i is the area of T_i, and $C(T_i)$ is the perimeter of T_i (sum of its edges lengths). This choice favors smoothness of the reconstructed hole surface for small dihedral angles, but acknowledges that noise maiy imply some large dihedral angles in which case optimizing the filling triangle shapes is prefered (the second criteria). The parameter ϵ is a threshold on the dihedral angle to switch between these two behaviors, and $90°$ is used in our experiment.

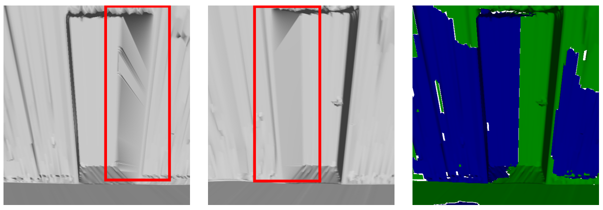
4. RESULTS AND DISCUSSION

In order to illustrate the performance of the proposed method, we experiment it on the Navvis dataset described in Section 2.

4.1 Moving Object Detection

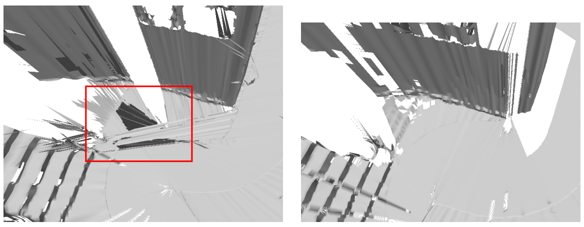
In the dataset, the present people may move fast or stay still for a short time. The moving object detection is based on inconsistency analysis through ray tracing and distance computation.

A focus on an inconsistent area is shown in Figure 9, there are two type of inconsistent objects: moving object and shortly static object. As shown in Figure 9(b), 9(c), 9(d), only the combination of distance computation, inter ray tracing and self ray tracing can recover all the moving objects without false alarms on the static part of the scene. In Figure 9(g) and 9(h), the blue

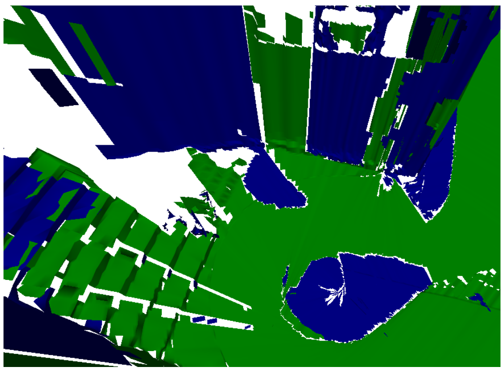


(a) Mesh from left sensor. (b) Mesh from right sensor. (c) Mesh fusion result.

Figure 10. An example of mesh fusion on triangle quality. In (a) and (b), triangles in red rectangles are elongated. In (c), the left mesh is in 🟦, and right mesh is in 🟩.



(a) Mesh from left sensor. (b) Mesh from right sensor.



(c) Result of triangle selection

Figure 11. An example of triangle selection. The left mesh is in 🟦, right mesh is in 🟩.

rectangle shows self ray tracing can remove self inconsistence objects in the single area between two meshes.

4.2 3D Mesh Fusion

There are two steps in the mesh fusion : triangle selection and stitching. In the triangle selection, we emphasize the triangle quality and self overlap. After triangle selection, stitching is utilized to obtain a continuous mesh.

(1) **triangle quality**: Because of the different scan direction of the two laser sensors, some mesh triangles can be very large. The label optimization selects the highest quality (=smallest) triangles, as shown in Figure 10, the high quality triangles are selected rather than the elongated triangles in the red rectangles due to the depth discontinuity.

(2) **self overlap triangles**: Some areas are scanned several times. In the experiment shown in Figure 11, most self overlaps occur on the ground areas: in Figure 11(a), the red area is covered several times while in Figure 11(c), only one layer is selected.

(3) **mesh stitching**: Mesh stitching is the last step to produce a continuous mesh. The gaps between mesh pieces resulting from triangle selection are filled by triangles. As shown in Figure 12, the triangles in magenta are added to fill the gaps. Boundary which are close enough are merged to avoid creating very small triangles.

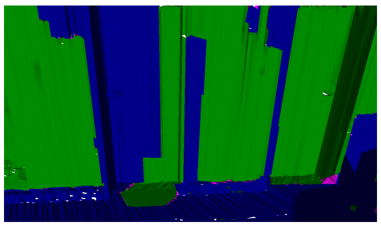


Figure 12. An example of mesh stitching. The left mesh is in 🟦, right mesh is in 🟩, and the added triangles are in 🟪.

In our experiment, if there are no self overlaps in the mesh, all the nodes can be labeled using QPBO method. If there is self overlap, not all the nodes are labeled so the QPBO-I method is used to improve the label ratio, which is slow if there are too many unlabelled triangle nodes. After mesh stitching, a few small holes may remain in the mesh, so the filling hole method of (Liepa, 2003) can be used to improve visualization and completeness.

4.3 Comparison

In order to show the effectiveness of the proposed method, results are compared with the Poisson method of (Kazhdan et al., 2006) and the greedy triangulation reconstruction method of (Marton et al., 2009) available in the PCL library (Rusu, Cousins, 2011). To ease comparison, the virtual points(cf Section 3.1) are also added to the point cloud as the input of the Poisson and triangulation methods.

After Poisson reconstruction, the mesh can be trimmed with density. In the greedy triangulation reconstruction, input point cloud is smoothed using bilateral smoothing in CGAL (The CGAL Project, 2018). The experiment shows that the Poisson method is robust to noisy points, but points on moving objects are different. As shown in Fig. 13(a), the result is different along to it density. Using density, some triangles are removed, but if there are a lot points, the triangles are remained as shown in Figure 13(b). The origin points are coarse, even Poisson give a rough reconstruction. And triangulation reconstruction is sensitive to noise points, as shown in Figure 13(c), even after smooth, the result is rough. Our method can produce a clean mesh.

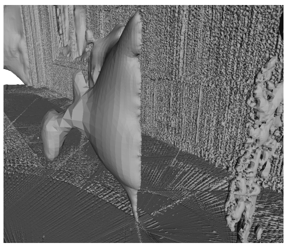
Another example is that, if a person stays static shortly, because there are a lot points on the back, the person is reconstructed well as shown in Figure 14(a). In our method, most part of the points are removed.

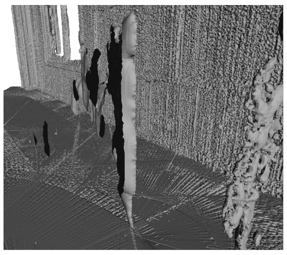
4.4 Block Stitch

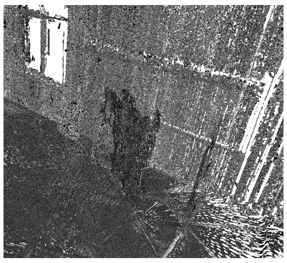
For large scale indoor reconstruction, QPBO-I method may become too slow if having too many triangles. We can divide the scene into blocks as shown in Figure 4, and use the triangle selection optimization framework block by block, then stitch all the resulting pieces as shown in Figure 15.

5. CONCLUSIONS AND FUTURE WORK

Emphasizing on moving objects removal and mesh fusion, this paper proposes a static indoor scene reconstruction method, using visibility analysis to remove moving objects, then using


(a) Poisson reconstruction **w/o** smooth and **w/o** trimmer.


(b) Poisson reconstruction **w/o** smooth **with** trimmer.


(c) triangulation reconstruction **with** smooth.

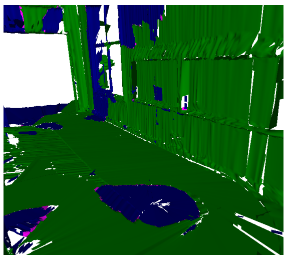
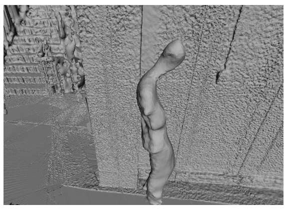

(d) Our method.

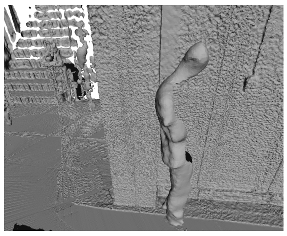
Figure 13. A comparison if there is a moving person with Poisson reconstruction(implicit method) and triangulation reconstruction(explicit method).

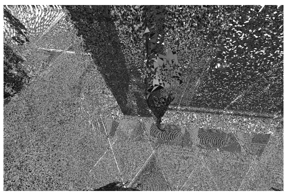
boolean optimization to select the triangles and a stitching method to fill the gaps. An important aspect of the method is that it relies on preserving sensor information, both optical center position for ray tracing and sensor topology for sensor mesh generation. Sensor information is important for 3D geometry analysis as advocated in (Xiao et al., 2015). Most 3D mesh reconstruction methods lose the sensor information despite it being physically significant (space is empty along the rays). Sensor topology mesh generation is an extremely fast way to produce a high quality mesh, but it does not cope with self overlaps that are very frequent as soon as the acquisition platform is moving freely, and it does not allow to integrate multiple data sources, which is why our proposed approach is a mandatory post-processing for such meshes. However, any reconstruction method adapted to keep the sensor information (in fact only the optical center position for each mesh vertex) such as the Graphcut method of (Labatut et al., 2009) can be used in our pipeline. It should also be well adapted fo fusing depth maps from dense matching, as the optical center is the same for all the points from the same depth map, and this information is often available.

Many objects in indoor scenes are unsustainable along the time. For example, in our experiment, only people are moving, but on a longer time scale objects and furniture could also move. Thus an interesting future work would be to perform a time series analysis of several scans acquired at different times of day and even at different dates to analyse the dynamic behavior of the scene at various time scales.

In the optimization step, all the triangles are treated as a graph node which does not scale up very well. If the mesh is very large, memory and time consumption will grow quadratically. A possible solution to this problem would be a divide and conqueer approach where the data is divided until the block sizes become reasonable enough to be processed at once, then the results can be iteratively stiched.


(a) Poisson reconstruction **w/o** smooth and **w/o** trimmer.


(b) Poisson reconstruction **w/o** smooth **with** trimmer.


(c) triangulation reconstruction **with** smooth.

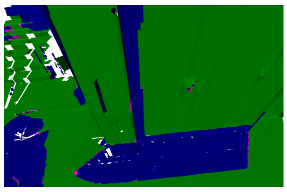
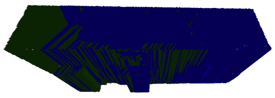
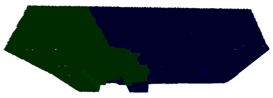

(d) Our method.

Figure 14. A comparison if there is a shortly static person with Poisson reconstruction(implicit method) and triangulation reconstruction(explicit method).


(a) Mesh from block 1 and 2.


(b) Mesh stitch result from block 1 and 2

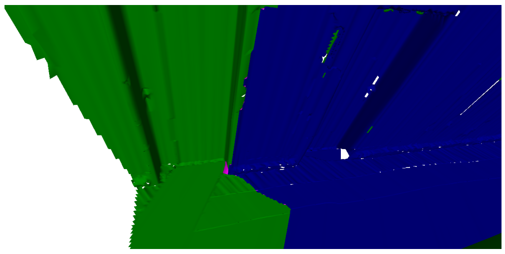

(c) A detail of block stitch result.

Figure 15. A block stitching experiment. Block 1 is in 🟦, block 2 is in 🟩, added triangle is in 🟪.

ACKNOWLEDGEMENTS

This research has been funded by ANR (Agence Nationale de la Recherche-France) International Project pLaTINUM (ANR-15-CE23-0010).

REFERENCES

Andreasson, H., Magnusson, M., Lilienthal, A., 2007. Has somethong changed here? autonomous difference detection for security patrol robots. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 3429–3435.

Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., Sharf, A., Silva, C. T., 2017. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36number 1, Wiley Online Library, 301–329.

Boussaha, M., Vallet, B., Rives, P., 2018. Large scale textured mesh reconstruction from mobile mapping images and lidar scans. *ISPRS 2018-International Society for Photogrammetry and Remote Sensing*, 49–56.

Gehrung, J., Hebel, M., Arens, M., Stilla, U., 2019. A fast voxel-based indicator for change detection using low resolution octrees. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5, 357–364.

Hokuyo, 2019. Hokuyo-USA::UST-10LX. https://www.hokuyo-usa.com/products/scanning-laser-rangefinders/ust-10lx.

Huitl, R., Schroth, G., Hilsenbeck, S., Schweiger, F., Steinbach, E., 2012. Tumindoor: An extensive image and point cloud dataset for visual indoor localization and mapping. *2012 19th IEEE International Conference on Image Processing*, IEEE, 1773–1776.

Jiang, C., Paudel, D. P., Fougerolle, Y., Fofi, D., Demonceaux, C., 2017a. Incomplete 3d motion trajectory segmentation and 2d-to-3d label transfer for dynamic scene analysis. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 606–613.

Jiang, C., Paudel, D. P., Fougerolle, Y., Fofi, D., Demonceaux, C., 2017b. Static and dynamic objects analysis as a 3d vector field. *2017 International Conference on 3D Vision (3DV)*, IEEE, 234–243.

Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction. *Proceedings of the fourth Eurographics symposium on Geometry processing*, 7.

Labatut, P., Pons, J.-P., Keriven, R., 2009. Robust and efficient surface reconstruction from range data. *Computer graphics forum*, 28number 8, Wiley Online Library, 2275–2290.

Liepa, P., 2003. Filling holes in meshes. *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, Eurographics Association, 200–205.

Lin, C.-C., Pankanti, S. U., Natesan Ramamurthy, K., Aravkin, A. Y., 2015. Adaptive as-natural-as-possible image stitching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1155–1163.

Lorensen, W. E., Cline, H. E., 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics*, 21(4), 163–169.

Marcus, B., Georg, S., 2017. NavVis - Enabling Digital Value Creation Indoors. *Surveying the world of tomorrow - From digitalisation to augmented reality. FIG Working Week*, Helsinki, Finnland.

Marton, Z. C., Rusu, R. B., Beetz, M., 2009. On fast surface reconstruction methods for large and noisy point clouds. *2009 IEEE international conference on robotics and automation*, IEEE, 3218–3223.

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Van Gool, L., Purgathofer, W., 2013. A survey of urban reconstruction. *Computer graphics forum*, 32number 6, Wiley Online Library, 146–177.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A., 2011. Kinectfusion: Real-time dense surface mapping and tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, IEEE, 127–136.

Pomerleau, F., Krüsi, P., Colas, F., Furgale, P., Siegwart, R., 2014. Long-term 3d map maintenance in dynamic environments. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 3712–3719.

Roldão, L., de Charette, R., Verroust-Blondet, A., 2019. 3d surface reconstruction from voxel-based lidar data. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2681–2686.

Rusu, R. B., Cousins, S., 2011. 3d is here: Point cloud library (pcl). *2011 IEEE international conference on robotics and automation*, IEEE, 1–4.

Ryde, J., Dhiman, V., Platt, R., 2013. Voxel planes: Rapid visualization and meshification of point cloud ensembles. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 3731–3737.

Schauer, J., Nüchter, A., 2018. Removing non-static objects from 3D laser scan data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 143, 15–38.

Schops, T., Schonberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., Geiger, A., 2017. A multi-view stereo benchmark with high-resolution images and multi-camera videos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3260–3269.

The CGAL Project, 2018. *CGAL User and Reference Manual*. 4.11.3 edn, CGAL Editorial Board.

Tron, R., Vidal, R., 2007. A benchmark for the comparison of 3-d motion segmentation algorithms. *2007 IEEE conference on computer vision and pattern recognition*, IEEE, 1–8.

Underwood, J. P., Gillsjö, D., Bailey, T., Vlaskine, V., 2013. Explicit 3d change detection using ray-tracing in spherical coordinates. *2013 IEEE international conference on robotics and automation*, IEEE, 4735–4741.

Wang, C.-C., Thorpe, C., 2002. Simultaneous localization and mapping with detection and tracking of moving objects. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, 3, IEEE, 2918–2924.

Wu, S.-T., Marquez, M. R. G., 2003. A non-self-intersection douglas-peucker algorithm. *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, IEEE, 60–66.

Xiao, W., Vallet, B., Brédif, M., Paparoditis, N., 2015. Street environment change detection from mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 107, 38–49.

Zhou, Q.-Y., Miller, S., Koltun, V., 2013. Elastic fragments for dense scene reconstruction. *Proceedings of the IEEE International Conference on Computer Vision*, 473–480.