

Building A Dense Surface Map Incrementally from Semi-Dense Point Cloud and RGB Images*

Qian-shan LI^{†1}, Rong XIONG^{†‡1}, Shoudong HUANG^{‡2}

¹State Key Laboratory of Industrial Control Technology, Hangzhou 310027, China

²Faculty of Engineering and Information Technology, The University of Technology, Sydney, NSW 2007, Australia

[†]E-mail: liqianshan@zju.edu.cn; rxiong@iipc.zju.edu.cn; shoudong.huang@uts.edu.au

Abstract: Building and using maps is a fundamental issue for bionic robots in field applications. A dense surface map, which offers rich visual and geometric information, is an ideal representation of the environment for indoor/outdoor localization, navigation and recognition tasks of these robots. Since most bionic robots can only use small light-weight laser scanners and cameras to acquire semi-dense point cloud and RGB images, we propose a method to generate a consistent and dense surface map from this kind of semi-dense point cloud and RGB images. The method contains two main steps: (i) generating a dense surface for every single scan of point cloud and its corresponding image(s) and (ii) incrementally fuse the dense surface of a new scan into the whole map. In step (i) edge aware resampling is realized by segmenting the scan of a point cloud in advance and resampling each sub-cloud separately. Noise within the scan is reduced and a dense surface is generated. In step (ii) the average surface is estimated probabilistically and the non-coincidence of different scans is eliminated. Experiments demonstrate that our method works well in both an indoor and outdoor semi-structured environment where there are regular shaped objects.

Key words: Bionic Robot, Robotic Mapping, Surface Fusion

doi: **Document code:** A **CLC number:** TU473

1 INTRODUCTION

Building and using maps is a fundamental issue for bionic robots in field applications like military surveillance, disaster relief, urban reconstruction, etc. where the maps are expected to carry detailed information. The map should not only show general environment shape, but also provide detailed context of the surface, which is important for navigating autonomously and implementing semantic tasks, such as object recognition and scene understanding. Most bionic robots use the configuration of a rotating 2D laser scanner and RGB camera to observe the environment. However, such a configuration can only lead to a relatively sparse map, i.e. a semi-dense point cloud. A sparse map contains very limited information which only reflects the general shape of the environment. Using this kind of map, a bionic robot can hardly localize itself or read subtle cues in the environment. Therefore it is necessary to find a way to build a detailed map from a semi-dense point cloud and its corresponding RGB images.

In the past decade, researchers in the field of robotic mapping often use laser scanners with a pan-tilt platform to collect 3D data (Cole and Newman, 2006)(Maurelli *et al.*, 2009). The obtained point cloud is said to be semi-dense because the angular step between two consecutive laser scan lines is often more than 1° and the laser scan range can be up to dozens of meters. In comparison, a point cloud collected by RGB-D cameras like Microsoft Kinect is said to be dense because the angular step between two neighboring pixel is about 0.1° only and the camera measurement range is just several meters. KinectFusion (Newcombe *et al.*, 2011) and Kintinuous (Whelan *et al.*, 2012) are good solutions for these RGB-D cameras to align and fuse dense data. But for large scenes, it is necessary to find a way to build a dense map using ordinary laser scanners and RGB cameras. The reason is two-fold: First, using Kintinuous with dense devices in large scenes is tedious because data should be collected up close in every corner of the environment. Second, it is impossible to use the combination of KinectFusion/Kintinuous with semi-dense devices because KinectFusion/Kintinuous use dense-voxels to statistically find the average surface and it doesn't support semi-dense data.

[‡] Corresponding author

* Project supported by the National Natural Science of Foundation of China (Nos. 61075078 and 61473258)



Fig. 1 Examples of semi-dense model and dense model. Top: colored point cloud gathered by a 2D laser scanner, which is semi-dense. Middle: Dense colored point cloud generated from the above semi-dense model by our method. Bottom: triangular surface mesh generated by our methods, where texture is mapped.

Although there are long range 3D lidars like Velodyne HDL-64E that can obtain a dense point cloud, they are either too expensive or of too narrow a field of view in some direction.

In this paper, we propose an incremental method to generate a dense surface model from a semi-dense point cloud. The method first generates a dense surface mesh for every single scan which consists of a 3D point cloud and its corresponding image(s) by doing edge aware resampling and triangulating the surface points. Then we incrementally fuse the dense surface mesh of a new scan into the whole map by estimating an optimal surface for the overlapping surface mesh to avoid non-coincidence of different scans. The top and the middle figure in Fig. 1 are the semi-dense point cloud obtained by our self-

made laser-camera system and the dense point cloud generated with our method, respectively. The bottom figure shows the generated dense surface map represented by a textured triangular mesh. We chose triangular mesh as the representation of the surface because it has low cost in storage and it is easy to access the neighboring vertices and triangular faces. In addition it is easy to attach the detailed texture information from an image on the surface.

The main contribution of this work is that a method is proposed to fuse a semi-dense point cloud and RGB images and generate a dense surface map, so that a bionic robot can use small light-weight 2D laser scanners and RGB cameras to build and to use a dense textured map. In detail,

- A segmentation-based resampling method is presented. To de-noise the data and resample the surface, the point cloud is first segmented into components and then every regular-shaped component is resampled individually. In segmentation, four rules on smoothness, distance, reflectivity and color respectively are tested to insure every segmented component is regular in shape.
- A probabilistic vertex relocation method is proposed for mesh fusion. After aligning two mesh frames into the same coordinate, vertices on the substantial surface are generated to replace the old ones by probabilistically averaging two observations on the same local surface around each vertex. The proposed method integrated with the re-triangulation operation ensures the fusion result is consistent both in geometry and in data structure.

In addition to the two above, we also give a triangulation method on organized point cloud data, which uses a mask on the data and then deletes unqualified triangles, and a re-triangulation method by detecting overlapping border vertices.

The remainder of the paper is organized as follows: Section 2 is about related work. Section 3 introduces how to get the dense surface from a semi-dense point cloud of a single scan. Section 4 introduces how to incrementally fuse the newly obtained dense surface into the whole map. Experimental results are presented in Section 5 and finally a conclusion is given in Section 6.

2 RELATED WORK

There are some practical instances where rotating 1D/2D laser scanners are used to generate a 3D point cloud. Schadler et al. used a continuously rotating 2D laser scanner to realize mapping and navigation in rough terrain (Schadler *et al.*, 2014). Wulf and Wagner introduced a 3D scanning system with laser time-of-flight measurement devices as well as some methods for fast scanning (Wulf and Wagner, 2003). Blésaca-Preciado et al. used their 3D laser scanning technical vision system, which is also based on a rotating laser scanner, to navigate autonomous mobile robots (Blésaca-Preciado *et al.*, 2014). And Lopez et al. introduced the design of an optical scanner based geodetic device for structural health monitoring (Lopez *et al.*, 2010). These works mainly focus on acquiring a 3D point cloud map which is mostly semi-dense, while the method proposed in this paper mainly focuses on building a dense surface model from this kind of semi-dense point cloud generated by 2D laser scanners.

In the field of robotics, building a dense map incrementally in a frame by frame mode is more the concern than operating in a post processing mode. KinectFusion (Newcombe *et al.*, 2011) is a famous system that incrementally builds a dense map in small-scale scenes from a dense point cloud. Its extension, Kintinuous (Whelan *et al.*, 2012), have proven effective in larger scenes. For mesh fusion, Lin et al. introduced a mesh composition method (Lin *et al.*, 2008) and Lou et al. proposed a triangular mesh merging method (Lou *et al.*, 2010). These works mainly focus on grafting one specific shape onto another. The most related forerunner in this field is the work of Marton et al. (Marton *et al.*, 2009), which proposed a triangulation method that incrementally adds input points into the triangular mesh. It is time efficient and a good option for robotic applications.

In the field of graphics, researchers focus more on how to construct the surface model given a completed point cloud. Various triangulation methods have been introduced, either for organized point cloud data (Crossno and Angel, 1999)(Holz and Behnke, 2013) or for unorganized data (Bajaj *et al.*, 1997)(Wang *et al.*, 2007). For unorganized data, Amenta et al. proposed the Crust family of algorithms (Amenta and Bern, 1999)(Amenta *et al.*, 2001) and Dey proposed the Cocone family (Dey *et al.*, 2001)(Dey *et al.*, 2011). In these methods, point data gathered at different positions are integrated in a set, on which the triangulation is implemented by

batch. These methods are widely used and have good performance.

3 SEGMENTATION-BASED RESAMPLING

To generate a dense surface mesh for every single scan of point cloud and its corresponding image(s), edge aware resampling is executed first to denoise the data and then the resampled point cloud is triangulated into a triangular surface mesh. We select triangular mesh as the representation of the surface because it has low cost in storage and it is easy to access the neighboring vertices and triangular faces. In addition it is easy to attach the detailed texture information from an image on the surface. However, one can also choose to generate a dense point cloud as RGB-D camera output.

3.1 Edge Aware Resampling

Noise in the measured point cloud is inevitable. Fusing surfaces of multiple scans could reduce the error to some extent but it may fail when the noise in the measurement is too large. So resampling the data is essential for every scan of a point cloud.

MLS(Moving Least Squares) is a popular surface fitting method (Rusu *et al.*, 2008)(Huang *et al.*, 2013). Benefitting from the idea of estimating the surface shape in a piecewise way, MLS works well to denoise the data. But this piecewise idea also makes it easy to lose some true corners and edges, which will heavily influence the result of triangulation and fusion. To prevent that, we propose to segment the point cloud data into components before using MLS. Some of the components may each represents a smooth surface patch, and they are resampled separately by MLS. With the method we propose, sharp corners and edges are preserved well and the result model can be more accurate. Comparing to methods which detect and protect features like corners and edges before resampling outside the protected area (Dey *et al.*, 2012)(Dey and Wang, 2013), our method first finds the patches which are regular enough to be resampled, and then resamples them separately. In some cluttered areas like tree areas, the data arrangement is out of order. It's impossible to resample or interpolate the data, and there is no need to do so. Our method picks out patches which can be resampled and resamples them separately, leaving the features as well as the cluttered areas unchanged.

For segmenting the point cloud into components,

region growing is a common algorithm. It recursively searches the neighborhood for new points which belong to the same region. In each region, points as a whole represent a smooth surface patch. The main effort is to define the criteria to evaluate whether a new neighbor point belongs to the same surface patch. We have designed the following 4 rules. Once a new neighboring point, P_i , is found around a member point, P_j , it will be rejected if any of the 4 rules fails.

- **Smoothness.** Neighboring points in a component should be smoothly connected. Otherwise, they might belong to different components. As mentioned above, resampling on a surface patch that contains points actually belonging to other surface patches will weaken the boundary between patches. So this criterion is the most important. Denote \mathbf{n}_i and \mathbf{n}_j to be the unit normal at P_i and P_j , respectively, \mathbf{v}_{ij} to be a vector from P_i to P_j , d_{ij} to be the distance between P_i and P_j . The local surface connecting P_i and P_j is considered smooth if and only if the following three equations are all satisfied.

$$\begin{cases} \frac{2}{d_{ij}} \sin \frac{\arccos |\mathbf{n}_i^T \mathbf{n}_j|}{2} < C_{th} & (1) \\ \arccos \frac{|\mathbf{n}_i^T \mathbf{v}_{ij}|}{d_{ij}} < \theta_{th} & (2) \\ \arccos \frac{|\mathbf{n}_j^T \mathbf{v}_{ij}|}{d_{ij}} < \theta_{th} & (3) \end{cases}$$

Condition (1) limits the local curvature lower than a threshold C_{th} , while (2) and (3) ensure that the two sample points locate in the same local plane, by constraining the angles between each normal and line segment ij . θ_{th} is the threshold of angle.

- **Distance.** It's possible that a true gap exists between two neighboring points, especially when the data is sparse. The bigger the distance between the two points, the higher the possibility. So the distance between two sample points, d_{ij} , should be small, to ensure that the two sample points are indeed in the same surface patch. Considering the fact that the distance between two neighboring points is proportional to the measuring distance of the laser scanner, points farther away from the sensor should be permitted to have a bigger distance. So this criterion is described as

$$\frac{d_{ij}}{d_i + d_j} < \alpha, \quad (4)$$

where d_i and d_j are the distances from the sensor to P_i and P_j respectively, and α is the threshold for point distance.

- **Reflectivity.** Significant difference in reflectivity indicates different materials. Although they might well be connected in geometry, separating them up reserves more context for further semantic use, and it also supports using different resampling parameters for surface patches of different materials. Denoting the laser reflectivity value at P_i , P_j as r_i , r_j respectively, the criterion is

$$|r_i - r_j| < \beta \quad (5)$$

where β is the threshold for reflectivity.

- **Color.** The criterion on color is used if and only if more semantic context is needed. In this case, the color of the two points c_i and c_j in Lab color space is compared, with the threshold γ ,

$$|c_i - c_j| < \gamma \quad (6)$$

After the above verifications, points are segmented into many components. The member points of each component share similar surface properties, and thus the re-prediction of each point's location through its neighbors is more reliable. Second order MLS is executed for each component, avoiding the influence of other components near the boundary.

It's worth mentioning that if the input point cloud is organized, the region-growing based segmentation can be implemented in a more efficient way. The organized point cloud forms an undirected graph which takes each pixel in the 'image' as a vertex and constructs an edge between every two adjacent pixels. A graph-based region growing scheme proposed by Felzenszwalb and Huttenlocher (Felzenszwalb and Huttenlocher, 2004) can be used in tandem with the four rules defined above. An edge connecting two adjacent points P_i and P_j in the 'image' would be deleted if any of the rules fails.

3.2 Triangulation

Since in many cases, each scan of a point cloud is gathered at a fixed location, the resampled points can be organized easily. So we provide a method to triangulate these organized data. Owing to the ordered feature of the data, neighbors of a point can be determined directly as

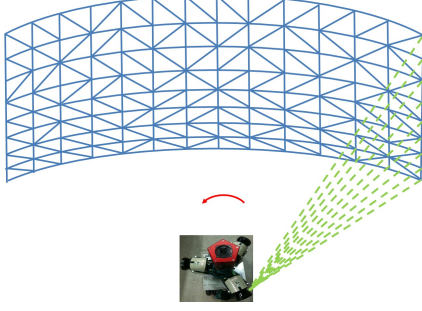


Fig. 2 Triangulation Mask

well as the triangulation. Unlike the methods in (Gopi and Krishnan, 2002) and (Marton *et al.*, 2009) which operate in a local neighborhood and gradually extend the triangulation boundary, our method directly use a triangulation mask on the organized data and then delete unqualified triangles (also called faces in this paper).

Since the point cloud data gathered by the laser scanners is in spherical coordinates, a triangular face can only be generated from three adjacent points. This means, for a vertex of a triangular face of the surface, denoted as P (\mathbf{P} is its vector form), the other two vertices of this triangular face are sure to be located in the neighborhood of P . Thus a mask shown in Fig. 2 can offer candidates of triangular faces. Each rectangle in the grid are split into two triangles by either of its diagonals, nw-se (northwest-southeast) or ne-sw (northeast-southwest), and the diagonal which produces the best triangle will be adopted. Knowing the four vertices of a rectangle, P_A, P_B, P_C, P_D , and their normals, $\mathbf{n}_A, \mathbf{n}_B, \mathbf{n}_C, \mathbf{n}_D$, the weight of a diagonal (here take AC for example) is specified by

$$w_{AC} = \max \left(\frac{(\mathbf{n}_B + \mathbf{n}_A + \mathbf{n}_C)^T ((\mathbf{P}_A - \mathbf{P}_B) \times (\mathbf{P}_C - \mathbf{P}_B))}{|(\mathbf{P}_A - \mathbf{P}_B) \times (\mathbf{P}_C - \mathbf{P}_B)|}, \frac{(\mathbf{n}_D + \mathbf{n}_A + \mathbf{n}_C)^T ((\mathbf{P}_A - \mathbf{P}_D) \times (\mathbf{P}_C - \mathbf{P}_D))}{|(\mathbf{P}_A - \mathbf{P}_D) \times (\mathbf{P}_C - \mathbf{P}_D)|} \right). \quad (7)$$

Here the maximization operator is used, which ensures that the selected diagonal can generate the best triangle. Although a poor triangle maybe generated simultaneously by the selected diagonal, there is a high possibility that the poor triangle does not exist. This is why we adopt the maximization operator rather than the average.

Because we use a full mask to generate the faces, some faces may be false, e.g. the face that contains the edge connecting foreground and background, and some faces may be very uncertain, e.g. the face with a normal direction that is nearly perpendicular to the sight line (laser beam). In order to filter out those false or

uncertain faces, the following criterion is defined: the angle between the face normal and the sight line (laser beam) should be less than a threshold θ_{th} .

$$\max \left(\arccos \left(\frac{\mathbf{n}_{\Delta_{ijk}} \cdot \mathbf{P}}{|\mathbf{P}|} \right) \right) < \theta_{th}, \mathbf{P} \in \{\mathbf{P}_i, \mathbf{P}_j, \mathbf{P}_k\}. \quad (8)$$

Here Δ_{ijk} denotes the triangular face formed by vertex P_i, P_j, P_k , and $\mathbf{n}_{\Delta_{ijk}}$ denotes its unit normal. As above, $\mathbf{P}_i, \mathbf{P}_j, \mathbf{P}_k$ are the vector form of P_i, P_j and P_k respectively. And here, the sensor locates at the origin.

Actually, $\mathbf{P}_i, \mathbf{P}_j, \mathbf{P}_k$ can be substituted by their range values r_i, r_j, r_k respectively, and a simpler criterion can be used:

$$\frac{\max(r) - \min(r)}{\min(r)} < d_{th}, r \in \{r_i, r_j, r_k\}. \quad (9)$$

d_{th} is the threshold. Faces that are rejected by this criterion are all deleted. Then camera image fragments are pasted onto the remaining faces.

4 PROBABILISTIC VERTEX RELOCATION

As the core of this paper, this section describes how the dense surfaces of multiple scans are fused together. The goal of fusion is to construct a surface model that is consistent not only in geometry but also in data structure. And the word ‘fusion’ in this section does not include the alignment of triangular meshes, but refers to fusing the denoised and aligned triangular mesh only. We use Scan-Matching and SLAM to align the surface meshes in advance. However, no matter how good the pose estimation performs, errors always exist, which means there are non-coincidences, i.e. gaps and intersections, among multiple overlapped surfaces. If the vertices of the overlapped region are re-triangulated directly after pose alignment, the resultant surface mesh would be very rough. To solve this problem, our solution is to relocate the vertices of the overlapped region onto a new ‘average’ surface, and then to re-triangulate these relocated vertices.

Since point cloud data are often obtained one scan at a time, our fusion works in an incremental way. Once a new scan is available, it is fused into the previous constructed map. So the problem is equivalent to fusing two meshes. One is the surface mesh that has been constructed from the beginning up to the current time. The other is the newest one. First, we relocate the vertices

of both frames according to each vertex's corresponding faces in the two frames and their uncertainty. Second, the relocated vertices are relinked to form a whole consistent mesh. The first step glues different layers of surface together by generating vertices on the substantial surface, while the second integrates two meshes into one. After that, image fragments with higher resolution are reserved and tailored for new faces.

4.1 Vertex Relocation

The vertex of a triangular face is actually a sample point on the real surface. Therefore all the neighboring faces of a vertex in one frame can be viewed as an observation of this vertex's local surface. Meanwhile, faces from another frame, which locate near this vertex after alignment, act as another observation of the local surface. Due to the inevitable errors, the two observations do not coincide. To relocate a vertex, our idea is to fuse the two observations by taking into account all the neighboring faces in the frame that the vertex belongs to and the nearest face in the other frame. A constraint on the nearest face in the other frame is that the difference between the normal of the discussed vertex and that of the nearest face in the other frame should be smaller than a threshold. In the fusion process, the uncertainty of each face is used as its weight. A face with less uncertainty plays a more important role in determining the new location, while a face with more uncertainty plays a lesser role. Obviously, taking the uncertainty into account can ensure the certainty of the fusion result. Overall, the uncertainty of a vertex can be reduced by relocation.

In the following part of this subsection, the calculation of the vertices' new locations is introduced. First, given the neighboring vertices/faces of a vertex as well as the uncertainties of these faces, how a vertex's new location is calculated is introduced. Then the calculation of the uncertainties of these faces is discussed in detail.

Let TM_1 be one triangular mesh frame and TM_2 the other frame. Assume the two meshes have been transformed into the same coordinate via scan matching techniques. Denote the vertex needed to be relocated in TM_1 by S , and its n neighbors by P_1, P_2, \dots, P_n . S and each two of its adjacent neighbors form a triangular face. The n triangular faces are denoted by $\Delta_{P_1SP_2}, \Delta_{P_2SP_3}, \dots, \Delta_{P_nSP_1}$. Notice that S must locate in the overlapping area of TM_1 and TM_2 . Denote S 's nearest triangular face in TM_2 by $\Delta_{Q_1Q_2Q_3}$, where Q_1, Q_2 and Q_3 are its vertices. $\Delta_{Q_1Q_2Q_3}$ can be determined by finding S 's nearest vertex

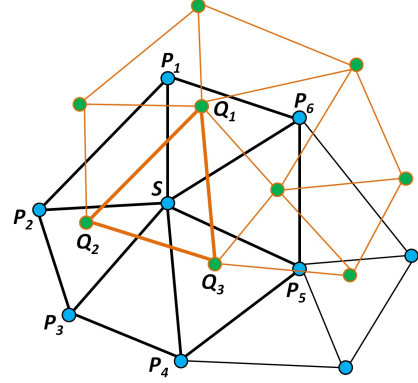


Fig. 3 For vertex S , $P_1, P_2, P_3, P_4, P_5, P_6$ are its neighbors in the frame of mesh TM_1 , while $\Delta_{Q_1Q_2Q_3}$ is its nearest triangular face in the other frame of mesh TM_2 and Q_1, Q_2, Q_3 are the three vertices.

from the kd-tree, then pick out the nearest face the vertex belongs. Fig. 3 shows an example.

If there are n ($n \geq 3$) non-coplanar faces in the neighborhood of S , e.g. $\Delta_{P_1SP_2}, \Delta_{P_2SP_3}, \Delta_{P_3SP_4}, \dots$, and a nearest face can be found in TM_2 , the new location of S , denoted as S_{new} , can be determined by

$$S_{new} = \underset{\mathbf{X}}{\operatorname{argmin}} \left(\frac{1}{n} \left(\frac{|(\mathbf{X} - \mathbf{S})^T \mathbf{n}_{P_1SP_2}|^2}{\sigma_{P_1SP_2}^2} + \frac{|(\mathbf{X} - \mathbf{S})^T \mathbf{n}_{P_2SP_3}|^2}{\sigma_{P_2SP_3}^2} + \frac{|(\mathbf{X} - \mathbf{S})^T \mathbf{n}_{P_3SP_4}|^2}{\sigma_{P_3SP_4}^2} + \dots \right) + \frac{|(\mathbf{X} - \mathbf{Q}_1)^T \mathbf{n}_{Q_1Q_2Q_3}|^2}{\sigma_{Q_1Q_2Q_3}^2} \right) \quad (10)$$

where $\mathbf{n}_{P_1SP_2}, \mathbf{n}_{P_2SP_3}, \mathbf{n}_{P_3SP_4}, \dots, \mathbf{n}_{Q_1Q_2Q_3}$ are the unit normals of corresponding triangular faces, and $\sigma_{P_1SP_2}^2, \sigma_{P_2SP_3}^2, \sigma_{P_3SP_4}^2, \dots, \sigma_{Q_1Q_2Q_3}^2$ are the variances, the details of which are described later in this section. $\frac{1}{n}$ means the influence of the faces from TM_1 is averaged, because it should be the two observations but not all the faces that play the same important role in calculating the new location.

Problem (10) can be solved in weighted least square form:

$$\mathbf{S}_{new} = (\mathbf{A}^T \mathbf{W}^2 \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W}^2 \mathbf{b}, \quad (11)$$

where \mathbf{W} is the weight matrix,

$$\mathbf{W} = \operatorname{diag} \left(\frac{\sigma_{P_1SP_2}}{\sqrt{n}}, \frac{\sigma_{P_2SP_3}}{\sqrt{n}}, \frac{\sigma_{P_3SP_4}}{\sqrt{n}}, \dots, \sigma_{Q_1Q_2Q_3} \right), \quad (12)$$

and

$$\mathbf{A} = (\mathbf{n}_{P_1SP_2} \mathbf{n}_{P_2SP_3} \mathbf{n}_{P_3SP_4} \dots \mathbf{n}_{Q_1Q_2Q_3})^T, \quad (13)$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{P}_S^T \mathbf{n}_{P_1SP_2} \\ \mathbf{P}_S^T \mathbf{n}_{P_2SP_3} \\ \mathbf{P}_S^T \mathbf{n}_{P_3SP_4} \\ \vdots \\ \mathbf{Q}_1^T \mathbf{n}_{Q_1Q_2Q_3} \end{bmatrix}. \quad (14)$$

and the covariance of the solution \mathbf{S}_{new} can be computed by

$$\Sigma_{\mathbf{S}_{new}} = (\mathbf{A}^T \mathbf{W}^2 \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W}^T \mathbf{W} \Sigma_{\mathbf{b}} \mathbf{W}^T \mathbf{W} \mathbf{A} (\mathbf{A}^T \mathbf{W}^2 \mathbf{A})^{-1}. \quad (15)$$

If a nearest face of S can be found in TM_2 , but there are fewer than three non-coplanar faces in the neighborhood of S , new constraints should be added. In case of two non-coplanar faces which intersect at a line l , the new location is constrained to be the projection of S on l . And in the case of only one face in the neighborhood, the new location is constrained to be the projection of S on the target plane.

If a nearest face of S cannot be found in TM_2 (the distance exceeds a certain value), nothing needs to be done because all the faces come from the same observation and the data has already been resampled (smoothed).

So far, the idea of vertex relocation has been presented, and the only question left is how to determine the variance $\sigma_{P_1SP_2}^2$, $\sigma_{P_2SP_3}^2$, $\sigma_{P_3SP_4}^2$ and $\sigma_{Q_1Q_2Q_3}^2$ in (10), by which the observation uncertainties take effect. Since these variances indicate the uncertainty of point-plane distance, they can be quantized by investigating the uncertainty propagation process.

When a point P is sampled by a general 3D laser scanner, 3 values about its location are obtained, including range r , vertical angle of observation ϕ and horizontal angle of observation θ . The variances of the three values are $\sigma_r^2(r)$, σ_ϕ^2 and σ_θ^2 respectively. $\sigma_r^2(r)$ is a function of range r . σ_ϕ^2 and σ_θ^2 indicate the angular accuracy. Then the point P can be expressed in local Cartesian coordinates by

$${}_L\mathbf{P} = \begin{bmatrix} r \cos(\phi) \cos(\theta) \\ r \cos(\phi) \sin(\theta) \\ r \sin(\phi) \end{bmatrix}, \quad (16)$$

with the covariance matrix

$$\Sigma_L \mathbf{P} = \mathbf{J}_{P2C} \begin{bmatrix} \sigma_r^2(r) & & \\ & \sigma_\phi^2 & \\ & & \sigma_\theta^2 \end{bmatrix} \mathbf{J}_{P2C}^T, \quad (17)$$

where \mathbf{J}_{P2C} is the Jacobian of the Polar-Cartesian Transformation.

On the other hand, the uncertainty of device pose is added to that of sample points. Given the 6D pose of the device

$$\mathbf{Pos}_d = (x_d \ y_d \ z_d \ \alpha_d \ \beta_d \ \gamma_d)^T, \quad (18)$$

with its covariance matrix $\Sigma_{\mathbf{Pos}_d}$, the global position of point P can be computed by

$$\mathbf{P} = f(\mathbf{Pos}_d, {}_L\mathbf{P}) = \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} {}_L\mathbf{P} \quad (19)$$

where

$$\begin{cases} R_{11} = \cos(\gamma_d) \cos(\beta_d) \\ R_{12} = \cos(\gamma_d) \sin(\beta_d) \sin(\alpha_d) - \sin(\gamma_d) \cos(\alpha_d) \\ R_{13} = \cos(\gamma_d) \sin(\beta_d) \sin(\alpha_d) + \sin(\gamma_d) \sin(\alpha_d) \\ R_{21} = \sin(\gamma_d) \cos(\beta_d) \\ R_{22} = \sin(\gamma_d) \sin(\beta_d) \sin(\alpha_d) + \cos(\gamma_d) \cos(\alpha_d) \\ R_{23} = \sin(\gamma_d) \sin(\beta_d) \cos(\alpha_d) - \cos(\gamma_d) \sin(\alpha_d) \\ R_{31} = -\sin(\beta) \\ R_{32} = \cos(\beta_d) \sin(\alpha_d) \\ R_{33} = \cos(\beta_d) \cos(\alpha_d) \end{cases}. \quad (20)$$

For any of the faces in Fig. 3, e.g. $\Delta_{P_1SP_2}$, the distance between S_{new} and the plane is derived by

$$\begin{aligned} d_{S_{new}-P_1SP_2} &= (\mathbf{S}_{new} - \mathbf{S})^T \mathbf{n}_{P_1SP_2} \\ &= (\mathbf{S}_{new} - \mathbf{S})^T \frac{\mathbf{N}_{P_1SP_2}}{|\mathbf{N}_{P_1SP_2}|} \\ &= (\mathbf{S}_{new} - \mathbf{S})^T \frac{\mathbf{V}_{SP_1} \times \mathbf{V}_{SP_2}}{|\mathbf{V}_{SP_1} \times \mathbf{V}_{SP_2}|} \\ &= (\mathbf{S}_{new} - \mathbf{S})^T \frac{(\mathbf{P}_1 - \mathbf{S}) \times (\mathbf{P}_2 - \mathbf{S})}{|(\mathbf{P}_1 - \mathbf{S}) \times (\mathbf{P}_2 - \mathbf{S})|}. \end{aligned} \quad (21)$$

Let

$$\mathbf{P}_1 = (x_{P_1} \ y_{P_1} \ z_{P_1})^T, \quad (22)$$

$$\mathbf{P}_2 = (x_{P_2} \ y_{P_2} \ z_{P_2})^T, \quad (23)$$

$$\mathbf{S} = (x_S \ y_S \ z_S)^T, \quad (24)$$

$$\mathbf{S}_{new} = (x \ y \ z)^T. \quad (25)$$

(21) can be rewritten as

$$d_{S_{new}-P_1SP_2} = \frac{\begin{bmatrix} x-x_S \\ y-y_S \\ z-z_S \end{bmatrix}^T \begin{bmatrix} (y_{p_1}-y_S)(z_{p_2}-z_S) - (z_{p_1}-z_S)(y_{p_2}-y_S) \\ (z_{p_1}-z_S)(x_{p_2}-x_S) - (x_{p_1}-x_S)(z_{p_2}-z_S) \\ (x_{p_1}-x_S)(y_{p_2}-y_S) - (y_{p_1}-y_S)(x_{p_2}-x_S) \end{bmatrix}}{\begin{vmatrix} (y_{p_1}-y_S)(z_{p_2}-z_S) - (z_{p_1}-z_S)(y_{p_2}-y_S) \\ (z_{p_1}-z_S)(x_{p_2}-x_S) - (x_{p_1}-x_S)(z_{p_2}-z_S) \\ (x_{p_1}-x_S)(y_{p_2}-y_S) - (y_{p_1}-y_S)(x_{p_2}-x_S) \end{vmatrix}} \quad (26)$$

By denoting \mathbf{J}_d as the Jacobian matrix of $d_{S_{new}-P_1SP_2}$ with respect to $x_{P_1}, y_{P_1}, z_{P_1}, x_S, y_S, z_S, x_{P_2}, y_{P_2}, z_{P_2}, x_S, y_S$ and z_S , the variance of $d_{S_{new}-P_1SP_2}$ can be derived.

$$\sigma_{d_{S_{new}-P_1SP_2}}^2 = \mathbf{J}_d \Sigma_{P_1, P_2, S} \mathbf{J}_d^T \quad (27)$$

where Σ_{P_1, S, P_2} is the covariance matrix of the three points P_1, S and P_2 . According to (19), we can denote the vector of the combination of P_1, S and P_2 as

$$\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{S} \\ \mathbf{P}_2 \end{bmatrix} = f(\mathbf{Pos}_{d, L} \mathbf{P}_1, L \mathbf{S}, L \mathbf{P}_2), \quad (28)$$

where $L\mathbf{P}_1, L\mathbf{S}$ and $L\mathbf{P}_2$ respectively represent the locations of P_1, S and P_2 in the local coordinate of the mesh frame they belong to. And the covariance matrix $\Sigma_{P_1, P_2, S}$ in (27) is calculated by

$$\Sigma_{P_1, S, P_2} = \mathbf{J}_f \begin{bmatrix} \Sigma_{\mathbf{Pos}_d} & & & \\ & \Sigma_{L\mathbf{P}_1} & & \\ & & \Sigma_{L\mathbf{S}} & \\ & & & \Sigma_{L\mathbf{P}_2} \end{bmatrix} \mathbf{J}_f^T, \quad (29)$$

where \mathbf{J}_f is the Jacobian of $f(\mathbf{Pos}_{d, L} \mathbf{P}_1, L \mathbf{S}, L \mathbf{P}_2)$ in (28), and $\Sigma_{\mathbf{Pos}_d}, \Sigma_{L\mathbf{P}_1}, \Sigma_{L\mathbf{S}}$ and $\Sigma_{L\mathbf{P}_2}$ are covariance matrices which can be calculated according to (17).

4.2 Re-triangulation

After all the vertices in the overlapping part are re-located onto a new consistent surface, the only operation left is to relink them. The method of Marton *et al.* (Marton *et al.*, 2009) provides a convenient option because it's simple and fast. In the cases when frames of data are obtained at sparse locations and the overlapping part between two frames is small, a simpler re-triangulation method by detecting overlapping border is introduced here.

Instead of destroying all the triangular faces in the overlapping region, we use the following strategy: set

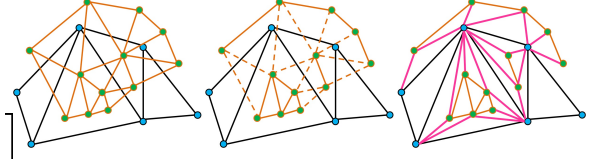


Fig. 4 Relink of vertices.

one frame as the reference denoted as RF , the link pattern of which is preserved, and then incrementally relink each vertex in the overlapping part of another frame (denoted as AF) into RF . The relink process is demonstrated in Fig. 4.

- Label the vertices in AF . In the previous relocation step, each vertex in the overlapping region of RF has found a nearest face in AF . So we label the vertices of this face in AF by the corresponding vertex in RF .
- Delete the edges in AF that connect vertices with different labels, as shown in the middle figure of Fig. 4. Along with the edge deleting, the faces that these edges belong to are also deleted. We call the vertices that these deleted edges connect as ‘border vertices’ and the other vertices as ‘non-border vertices’.
- Insert the ‘non-border vertices’ into their corresponding nearest triangular face in RF , where the original edges among them are reserved.
- Link the ‘border vertices’ in AF with RF . It can be regarded as linking between two polylines. First each vertex on one polyline (denoted by $PL1$) is linked to the nearest vertex on the other polyline (denoted by $PL2$), and then each unlinked vertex on $PL2$ is linked to the nearest vertex on $PL1$.

5 EXPERIMENTAL RESULTS

5.1 Experimental Device

The device used for experiments is shown in Fig. 5, which integrates three SICK LMS-151 laser scanners with a PointGrey Ladybug III panoramic camera. It is an extension of the design introduced by Sheehan *et al.* (Sheehan *et al.*, 2012). We designed this prototype to verify the feasibility of our method, and further a light-weight edition will be developed and deployed on a bionic robot. The method in (Sheehan *et al.*, 2012) is used for the intrinsic calibration of the three-laser-scanner, and that in (Pandey *et al.*, 2010) is for the cross-



Fig. 5 The RGB-D Unit: three SICK LMS-151 laser scanners and a PointGrey Ladybug 3 panoramic camera are integrate together.

calibration between the lasers and the camera. The RGB-D data obtained has a nearly full spherical field of view but has two black areas in the laser scanner data. One is between $-90^\circ \sim -45^\circ$, close to the bottom area of the device, where the data contains the bracing components of the device, e.g. bracket and car roof. Another is the zone where the panoramic camera is mounted. The laser scanners can hardly block the view of the camera. With the scanning rate of 50 lines/s of each SICK LMS-151 and the rotational speed of $60^\circ/s$ of the disc, the device achieves an horizontal angular resolution of 1.2° . Using this device, we can not only build the surface but also attach high resolution textures to the surface.

5.2 General results

To demonstrate the effectiveness of our method, several experiments in both indoor and outdoor scenes of our university are conducted. Fig. 6 demonstrates two results: one is from 4 frames of point cloud collected in the yard of our institute, and the other is the model built in a corner of our lab with 10 frames of data.

Fig. 7 demonstrates the result in a semi-structured environment. When the robot traverses in the wild, potential targets like buildings and vehicles can be recorded.

Comparing to the results of (Cole and Newman, 2006) and (Maurelli *et al.*, 2009) which only generate semi-dense point clouds of the environment, our result is dense and textured on the regular shaped parts of the environment. This is important for a bionic robot in inspecting the environment. Comparing to the results of KinectFusion (Newcombe *et al.*, 2011) and Kintinuous (Whelan *et al.*, 2012) which use a dense point cloud as input and use a dense voxel grid to estimate

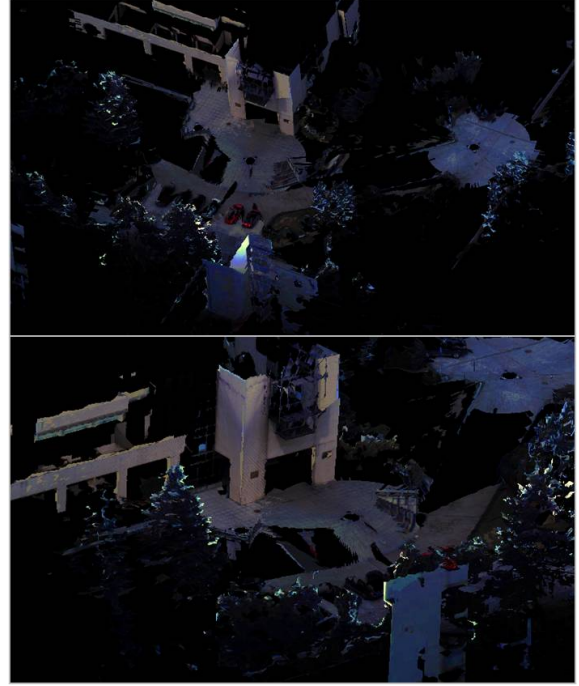


Fig. 7 Results in semi-structured environment where there are regular shaped objects like buildings as well as irregular shaped objects like trees.

the ‘average’ surface, our result is not that exquisite. However, while they take dozens of scans to reconstruct a small scene, we reconstruct a large scene using just a few scans. Comparing to the result of (Marton *et al.*, 2009) which uses RMLS(Robust Moving Least Square) to resample the noisy data and gradually triangulate new points into existing surface mesh, our result is better when fusing surface meshes of significantly different uncertainties because the probabilistic fusion method gives more weight to observations with more certainties.

5.3 Resampling

Fig. 8 illustrates the importance of resampling: if the input point data are not resampled before triangulation, the triangular surface would be influenced by noise. In some serious cases, too many faces are deleted, leaving large holes on the surface, as shown in the left sub-figure. Fig. 9 compares the point clouds after resampling with the traditional MLS method and our segmentation-based MLS method. It is shown that sharp edges and corners are well preserved by our method, due to non-interference among different geometric components.



Fig. 6 General results of outdoor and indoor scenes. Upper left: outdoor surface. Upper right: outdoor surface with texture. Middle: indoor surface. Bottom: indoor surface with texture.

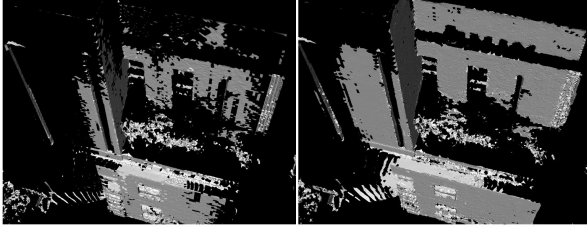


Fig. 8 Comparison of triangulation results without and with resampling. Left: direct triangulation without resampling. Right: triangulation with resampling first.

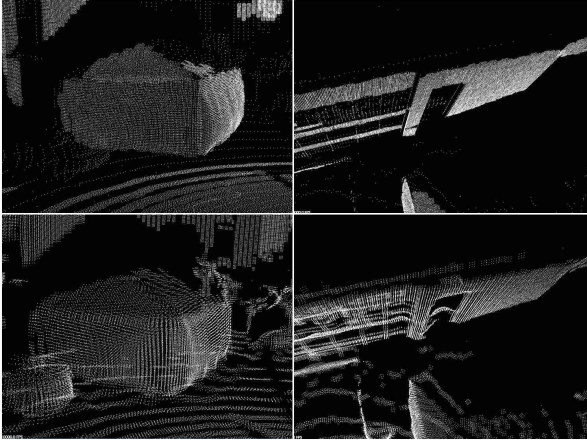


Fig. 9 Segmentation based resampling v.s. Ordinary MLS. By resampling separately on each geometric component, the resulted cloud preserves better shape and is much cleaner (upper left and upper right), while ordinary MLS gets confused at the boundary (lower left and lower right). In this example, points are upsampled to better demonstrate the resampling result.



Fig. 10 Comparison of results without and with vertex relocation. Left: having rough surface at the joint when directly re-triangulating vertices. Right: much smoother at the joint when relocating vertices then re-triangulating.

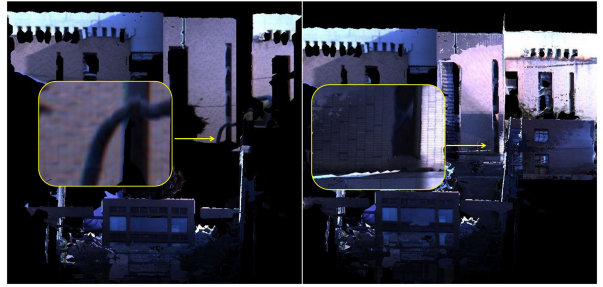


Fig. 11 Texture Update: Left: observation from 10m away. Right: updated by observations from 3m away.

5.4 Vertex relocation

This experiment demonstrates the necessity of vertex relocation. As mentioned above, this operation relocates the vertices of overlapped surface mesh onto the underlying surface, which would remove the gap between two surface mesh and make the resultant mesh smooth at the joint area. Fig. 10 shows the part of a building where two frames of surface mesh overlapped. In the left figure, re-triangulation is directly carried out without relocating any of the vertices, so the surface at the joint is rough. While in the right one, vertices are first relocated onto the underlying surface, so the fused mesh is much smoother.

5.5 Surface update

Since the proposed method has a probabilistic nature, we are able to demonstrate its advantage in surface update. As is indicated in Fig. 11, as soon as a closer observation (which is more certain) is made, the previous low resolution surface is updated to be finer.

5.6 Change in vertices' covariance

Fig. 12 plots the histograms of maximum eigenvalue of the point covariance matrix (square root) and

Fig. 13 plots that of the traces. The maximum eigenvalue indicates the primary axis length of covariance of a point, while the trace indicates the total length of the three axes, which also measures the uncertainty of a point. For almost every histogram, it is true that the black shape (data after fusion) locates more on the left than the green one (data before fusion) does. Thus fusion indeed decreases the uncertainties of points, (i.e. vertices of surface).

6 CONCLUSIONS

In this paper, we present a method to fuse a semi-dense point cloud and RGB images and generate a dense surface map. The resultant dense surface map offers rich visual and geometric information, and is an ideal representation of the environment for indoor/outdoor localization, navigation and recognition tasks of bionic robots. With this method, a bionic robot can use small light-weight 2D laser scanners and RGB cameras to build and to use a dense textured map. Experiments conducted in our university campus demonstrate the effectiveness of our method in the semi-structured environment.

First, the current method to generate a dense surface map relies on a hypothesis that the environment surface is ‘smooth’ enough to be represented by textured mesh. This hypothesis heavily limits the usage of bionic robots. In un-structured environments, the data is cluttered and no surface can be generated. Then the image, which is more dense than the sparse laser points, can not be used to interpolate and to enrich the surface, not to mention the self-localization problem. So, to extend this method to un-structured environments is of great significance.

Second, the current stop-and-go mode of mapping is not that intelligent nor elegant, and mapping while the robot is moving can be more practical. Considering that bionic robots do not have odometry to indicate the displacement of the robot, visual methods can be used and incorporated with laser observations to estimate the robot pose in real time.

References

- Cole, D.M., Newman, P.M., 2006. Using laser range data for 3D S-LAM in outdoor environments. IEEE International Conference on Robotics and Automation (ICRA), p.1556-1563.
- Maurelli, F., Droschel, D., Wisspeintner, T., May, S., Surmann, H., 2009. A 3d laser scanner system for autonomous vehicle navigation. IEEE International Conference on Advanced Robotics (ICAR), p.1-6.
- Schadler, M., Střžekler, J., Behnke, S., May, S., Surmann, H., 2014. Rough Terrain 3D Mapping and Navigation Using

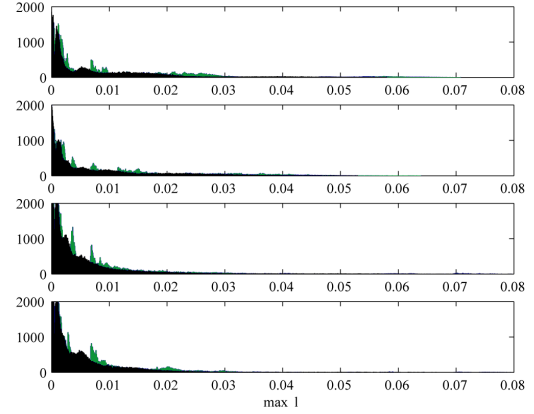


Fig. 12 Histograms of four frames of surface vertices on maximum eigenvalue of squared covariances. Blue: data before fusion. Black: data after fusion.

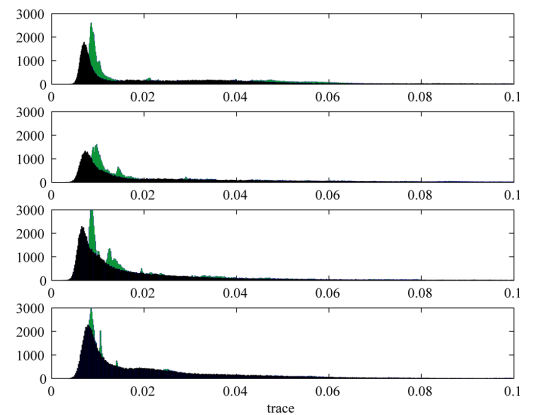


Fig. 13 Histograms of four frames of surface vertices on the trace of squared covariances. Green: data before fusion. Black: data after fusion.

- a Continuously Rotating 2D Laser Scanner. *KI-Künstliche Intelligenz*, p.1-7.
- Wulf, O., Wagner, B., 2003. Fast 3D scanning methods for laser measurement systems. *International conference on control systems and computer science*, p.2-5.
- Blácsa-Preciado, L. C., Sergiyenko, O. Y., Rodríguez-Quinonez, J. C., Garcíla, X., Garcíla, X., Rivas-Lopez, M., ..., Starostenko, O., 2014. Optical 3D laser measurement system for navigation of autonomous mobile robot. *Optics and Lasers in Engineering*, **54**:159-169.
- Lopez, M. R., Sergiyenko, O. Y., Tyrsa, V. V., Perdomo, W. H., Cruz, L. D., Balbuena, D. H., ..., Hipolito, J. N. 2010. Optoelectronic method for structural health monitoring. *Structural Health Monitoring*, **9**(2):105-120.
- Níczter, A., Lingemann, K., Hertzberg, J., Surmann, H., 2007. 6D SLAM - 3D mapping outdoor environments. *J. Field Robotics*, **24**(8-9):699-722.
- Newcombe, R.A., Davison, A.J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Fitzgibbon, A., 2011. KinectFusion: Real-time dense surface mapping and tracking. *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, p.127-136.
- Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., McDonald, J., 2012. Kintinuous: Spatially extended kinectfusion. *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*.
- Crossno, P., Angel, E., 1999. Spiraling edge: Fast surface reconstruction from partially organized sample points. *IEEE International Conference on Visualization*, p.317-324.
- Holz, D., Behnke, S., 2013. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. *Intelligent Autonomous Systems*, **12**:61-73.
- Bajaj, C. L., Bernardini, F., Xu, G., 1997. Reconstructing surfaces and functions on surfaces from unorganized three-dimensional data. *Algorithmica*, **19**(1-2):243-261.
- Wang, Y. B., Sheng, Y. H., Lv, G. N., TIAN, P., ZHANG, K., 2007. A Delaunay-based Surface Reconstruction Algorithm for Unorganized Sampling Points. *J. Image and Graphics*, **12**(9):1537-1543.
- Amenta, N., Bern, M., 1999. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, **22**(4):481-504.
- Amenta, N., Choi, S., Kolluri, R.K., 2001. The power crust. *The Sixth ACM Symposium on Solid Modeling and Applications*, p.249-266.
- Dey, T.K., Giesen, J., Hudson, J., 2001. Delaunay based shape reconstruction from large data. *IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*, p.19-146.
- Dey, T.K., Dyer, R., Wang, L., 2011. Localized Cocone surface reconstruction. *Computers and Graphics*, **35**(3):483-491.
- Lin, J., Jin, X., Wang, C.C., Hui, K.C., 2008. Mesh composition on models with arbitrary boundary topology. *IEEE Transactions on Visualization and Computer Graphics*, **14**(3):653-665.
- Lou, R., Pernot, J.P., Mikchevitch, A., Vléron, P., Lou, R., Pernot, J.P., Mikchevitch, A., Vléron, P., 2010. Merging enriched finite element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance. *Computer-Aided Design*, **42**(8):670-681.
- Marton, Z. C., Rusu, R. B., Beetz, M., 2009. On fast surface reconstruction methods for large and noisy point clouds. *IEEE International Conference on Robotics and Automation (ICRA)*, p.3218-3223.
- Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M., Beetz, M., 2008. Towards 3D point cloud based object maps for household environments. *Robotics and Autonomous Systems*, **56**(11):927-941.
- Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U., Zhang, H.R., 2013. Edge-aware point set resampling. *ACM Transactions on Graphics (TOG)*, **32**(1):9.
- Dey, T. K., Ge, X., Que, Q., Safa, I., Wang, L., Wang, Y., 2012. Feature-Preserving Reconstruction of Singular Surfaces. *Computer Graphics Forum*, **31**(5):1787-1796.
- Dey, T. K., Wang, L., 2013. Voronoi-based feature curves extraction for sampled singular surfaces. *Computers and Graphics*, **37**(6):659-668.
- Felzenszwalb, P.F., Huttenlocher, D.P., 2004. Efficient Graph-Based Image Segmentation. *Intl. J. Computer Vision*, **59**(2):167-181.
- Gopi, M., Krishnan, S., 2002. A fast and efficient projection-based approach for surface reconstruction. *IEEE XV Brazilian Symposium on Computer Graphics and Image Processing*, p.179-186.
- Sheehan, M., Harrison, A., Newman, P., 2012. Self-calibration for a 3D laser. *Intl. J. Robotics Research*, **31**(5):675-687.
- Pandey, G., McBride, J., Savarese, S., Eustice, R., 2010. Extrinsic calibration of a 3d laser scanner and an omnidirectional camera. *7th IFAC Symposium on Intelligent Autonomous Vehicles*.