2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Incremental Probabilistic Geometry Estimation for Robot Scene Understanding

Louis-Kenzo Cahier, Tetsuya Ogata and Hiroshi G. Okuno

***Abstract*— Our goal is to give mobile robots a rich representation of their environment as fast as possible. Current mapping methods such as SLAM are often sparse, and scene reconstruction methods using tilting laser scanners are relatively slow. In this paper, we outline a new method for iterative construction of a geometric mesh using streaming time-of-flight range data. Our results show that our algorithm can produce a stable representation after 6 frames, with higher accuracy than raw time-of-flight data.**

INTRODUCTION

Service robots need a multi-sensory perception of their environment. This is because human-made environments are both unpredictable and assume understanding of a variety of modalities, either for access to distinctive features for object recognition, or communication. Geometric shape and visual appearance are particularly important for object recognition; audition and vision for communication. Indeed human communication is based mostly on oral language and auditive symbols (e.g., repetitive sounds to attract attention), as well as written language and graphical symbols (e.g., arrows).

With the latest generation of image ranging sensors and the maturity reached by Simultaneous Localization And Mapping (SLAM) algorithms [1], we now have the necessary tools to achieve unified audio-visuo-geometrical maps. Time-of-Flight (ToF) optical cameras recently bridged the gap to obtaining quick and reliable depth data even in dark conditions or without visual texture. SLAM enables mobility by providing ego-motion estimates in unknown environments. The challenge now lies in building rich maps that reflect reality, by fusing multiple sensors over time integrated in the same global reference frame using sparse SLAM features.

We propose attacking this problem under an important assumption: relevant visual and auditive information lies on the surface geometry of the environment. The justification is in the fact that the solid state of matter is by far the most frequent, and sound sources are also vibrating solid objects. Under this condition, the environment geometry acts as a topologically 2 dimensional support space for sensory information, in which auditive and visual information can be embedded. This leads to a reduction of the sensory space size, and to common addressability.

In this paper we attempt to incrementally build and maintain a polygonal mesh with embedded probabilistic spatial uncertainty that can serve as the geometric support for perception under our assumption. Our ultimate goal is to use this multi-modal scene representation as a new primitive data type, replacing raw video images, directional sound source localization and range scans as a basis for scene understanding (see Fig. 1).

All authors are with the Graduate School of Informatics, University of Kyoto, 606-8501 Kyoto, Japan {kenzo,ogata,okuno}@kuis.kyoto-u.ac.jp

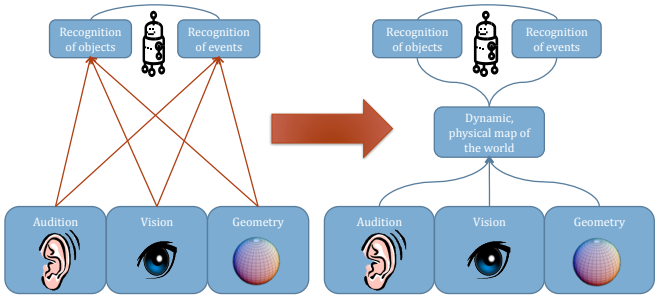

Fig. 1. Unified map scheme for scene understanding.

To this end, we make use of a recently available sensor, time-of-flight cameras. These optical cameras measure the depth of each pixel, which combined with a traditional color camera provides Red-Green-Blue-Depth (RGBD) images at high frequencies. With the mass-produced Microsoft Kinect introduced in late 2010, 30 Hz $640 * 480$ px RGB images are available for less than $200. This source of information, while not exempt of problems, is what the stereo vision community has been striving to obtain for years.

Time-of-flight technology has qualities that makes it desirable for scene understanding, provided sensor noise is reduced. First is speed; at 30 frames per second, it is possible to collect 3D data on a scene much faster than a pan-tilt laser scanner can, which is important for mobile robots that value interactivity highly. Second is low processing cost. The main drawback of ToF cameras is the significant noise, especially off reflective surfaces. It is this unreliability that we attack in this paper, to improve the ToF sensor's utility for robotics, vision, and other fields.

I. RELATED WORKS

The field of SLAM has a rich literature related to mobile mapping, but uses types of maps that do not aim at reflecting reality accurately. SLAM maps have different priorities than those for scene understanding: they are a necessary side effect of ego-motion estimation. Testimony to this is that SLAM algorithms are evaluated on the accuracy of the ego-motion trajectory, not of the map. Our approach attempts to develop dense maps to describe physical properties of the world more closely. The problem of adapting dense maps on the skeleton of sparse SLAM feature points that change over

978-1-4673-1404-6/12/$31.00 ©2012 IEEE

time is tackled by the DenseSLAM [2] framework, which is flexible enough to accommodate various types of dense maps, including the one we propose in this paper.

Works in photo-realistic reconstruction share the objective of geometric reconstruction, although with dependence on visual data and batch methods. For example, the impressive results of Newcombe et al. [3]. Two distinctions can be made: we do not use visual texture to estimate geometry, making reconstructions in dark or uncontrasted environments possible; we attempt to incrementally update the model. The approach of Marton et al. [4] has the same distinction, relying on data accumulation to reduce noise, which induces a lag in the scene understanding pipeline. The main novelty of our work is to continuously update the geometry model to mirror reality and reduce noise, with results immediately available.

Finally, the surface reconstruction community has long focused on building very accurate models with controlled precision from large batch datasets with relatively long processing times. High levels of accuracy are obtained, for example with the Poisson reconstruction algorithm introduced by Kazhan et al. [5], albeit with significant computation times in the order of several seconds, which is undesirable for mobile robot applications.

In the following sections, we describe a new method to transform ToF data into a probabilistically smoothed polygonal mesh. We first explain how the robot could internally represent its surroundings as a mesh representation. Then we describe a method for maintaining this mesh in an online manner, through triangulation of point estimates and converging to stable values through maximum likelihood estimation.

II. Probabilistic Polygonal Mesh

Let us suppose we receive 2.5D scans at a sequence of discrete times $T = (t_1, \dots, t_n)$. For each time $t \in T$, the scan of height and width $(h, w) \in \mathbb{N}^2$ is a pair $S_t = (O_t, P_t) \in \mathbb{R}^3 \times \mathbb{M}_{h,w}(\mathbb{R}^3)$; the first element O_t is the position of the optical center of the range imaging device at time t; the second element P_t is the matrix of point estimates at time t, in the form of a matrix of 3D points. We assume a thin-beam model of measurement; where this may be abusive, approaches to mitigate the effects of pixels integrating distance over multiple objects [6] may be employed. We will note $\mathcal{D}_{h,w} = \{1, \dots, h\} \times \{1, \dots, w\}$ the pixel domain of width w and height h. P_t can be obtained from a matrix of distances by using the optical model of the ranging device to obtain, for each pixel, the point result of the translation of O_t, along the optical line-of-sight to the pixel, by the measured distance.

Our goal is to build, for each time $t \in T$, a surface reconstruction of the environment E_t, using only S_t and E_{t-1}. We choose to represent surfaces by triangular meshes, for their relative simplicity of implementation, the large body of existing algorithms, and hardware support, including efficient rendering with texturing. We define a triangular mesh as a pair $M = (V, F) \in \mathbb{R}^3 \times \mathbb{N}^3$, where the first element V is the set of vertices, and the second F is a set of triplets of indices into V_t; that is, each triplet $f \in F$ represents the triangle $\triangle V_{\pi_1(f)} V_{\pi_2(f)} V_{\pi_3(f)}$ (where π_i is an informal notation for the surjective canonical projection of a 3-tuple into its i-th component). Accordingly we have $E_t = (V_t, F_t)$, and will note $\mathcal{S}(E_t)$ the surface represented by E_t, that is: $\{P \in \mathbb{R}^3 : \exists f \in F_t, P \in \triangle V_{\pi_1(f)} V_{\pi_2(f)} V_{\pi_3(f)}\}$. However, to compute the environment surface E_t incrementally from E_{t-1}, we wish to introduce a measure of probabilistic uncertainty in the surface representation, the determination and use of which will be explained in section IV-B.

A. Data structure

To this effect, we introduce an augmented mesh structure we call a Probabilistic Polygonal Mesh (PPM), consisting of a mesh each vertex of which is associated with a measure of probabilistic certainty. Thus, for each time $t \in T$, E_t is a triplet $(V_t, F_t, \Sigma_t) \in \mathbb{R}^3 \times \mathbb{N}^3 \times \mathbb{R}$, where the first two elements define the mesh, and the third element Σ_t is a tuple of real numbers with the same arity as V_t.

As will become clear in section II-B, for an index $i \in \{1, \dots, |V_t|\}$, $\pi_i(F_t)$ is an approximation of the local Gaussian variance of the belief (i.e. uncertainty) in the current surface reconstruction E_t around $\pi_i(V_t)$.

B. Probabilistic belief

Indeed, PPMs are meant to represent the belief of a robot that each point in space is occupied. At time $t \in T$, this belief is a probability distribution $\mathcal{B}_t(P) : \mathbb{R}^3 \to [0, 1]$ with $\iiint_{\mathbb{R}^3} \mathcal{B}_t(P) \, \mathrm{d}P = 1$.

To define this probability distribution from E_t, we use an auxiliary function $\phi(P; E_t)$ that associates any point P on the surface of E_t to the barycentric interpolation of the variances of the vertices of the face containing P.

For any three points $P_1, P_2, P_3 \in V_t$, the barycentric coefficients $\lambda_1, \lambda_2, \lambda_3 \in \mathbb{R}$ of a point $P \in \triangle P_1 P_2 P_3$ are the solution to the equation $P = \sum_{i=1}^{3} \lambda_i P_i$. Note that the solution is unique and always exists provided P_1, P_2 and P_3 are distinct and not collinear, as this makes it a system of three independent equations in three unknowns. The barycentric interpolation of the variances $\sigma_1, \sigma_2, \sigma_3$ at points P_1, P_2, P_3 respectively of PPM E_t, is given by the function $\phi(.; E_t)$ defined by $\phi(P; E_t) = \sum_{i=1}^{3} \lambda_i \sigma_i$.

The un-normalized belief $\widetilde{\mathcal{B}_t}$ is then given by:

$$\widetilde{\mathcal{B}_t} = \mathcal{N}\left(P ; \overbrace{\operatorname*{argmin}_{x \in \mathcal{S}(E_t)} (\|x, P\|_2)}^{mean}, \overbrace{\phi\left(\operatorname*{argmin}_{x \in \mathcal{S}(E_t)} (\|x, P\|_2) ; E_t \right)}^{variance} \right) \quad (1)$$

The normalized belief $\mathcal{B}_t$ is obtained by normalizing $\widetilde{\mathcal{B}_t}$: $\mathcal{B}_t(P) = \frac{\widetilde{\mathcal{B}_t}(P)}{\iiint_{\mathbb{R}^3} \widetilde{\mathcal{B}_t}(Q) \, \mathrm{d}Q}$. $\mathcal{B}_t$ is a parametric probability distribution with parameters given as the vertices, triangles and variances (V_t, F_t, Σ_t) of a PPM, for which the value

of each point in space is defined as the value of a single Gaussian that depends on the closest point on the PPM's surface. Connex, C^∞ class subsets of space are defined by the same Gaussian, partitioning space along either lines orthogonal to $\mathcal{S}(E_t)$ (all the points in the subset have the same closest point of $\mathcal{S}(E_t)$ on the inside of a triangle), or wedges rooted on a vertex in V_t (all points in the subset have the same closest point, a vertex in V_t).

$\mathcal{B}_t$ can be informally thought of as a distribution over space that has a constant maximum along the 2D manifold that is $\mathcal{S}(E_t)$, and diffuses into space following Gaussian distributions whose variances are interpolated from the PPM vertex variances Σ_t at the site of the closest points on $\mathcal{S}(E_t)$.

III. Incremental PPM building

We now turn to the incremental construction of the $(E_t)_{t\in T}$ by presenting a recursive procedure to build and update E_t from E_{t-1}. At the first time step t_1, we start with an empty mesh $E_0 = (\emptyset, \emptyset, \emptyset)$. The rest of this section describes the recursion step at an arbitrary time step $t \in T$.

A. Ray casting

We start by casting a ray from the optical center O_t through each estimate in the points matrix P_t and checking for the first intersection, if any, with $\mathcal{S}(E_{t-1})$ (see Fig. 3 to visualize). Formally, for each pixel $(x, y) \in \mathcal{D}_{h,w}$, if we note $\mathring{I}_t(x, y) = \underbrace{\left\{P \in \mathbb{R}^3 : \exists\lambda \in \mathbb{R}^*_+, P = O_t + \lambda\overrightarrow{O_tP_t(x, y)}\right\}}_{ray} \bigcap \mathcal{S}(E_{t-1})$ the set of all intersections of the ray with the PPM, we compute the closest intersection $I_t(x, y) = \underset{P\in\mathring{I}_t(x,y)}{\operatorname{argmin}} (\|P - O_t\|_2)$, which may be of cardinality 0 (no intersection) or 1 (the closest of at least one intersection).

This is used to partition P_t in 2 sets $\widehat{P}_t$ and $\widetilde{P}_t$ which contain, respectively, all points that had an intersection in the ray-casting step, and had none. Thus $\widehat{P}_t = \{P_t(x, y) \in P_t : |I_t(x, y)| = 1\}$ and $\widetilde{P}_t = \{P_t(x, y) \in P_t : |I_t(x, y)| = 0\}$. Intuitively, $\widetilde{P}_t$ is the set of points of the environment that have not yet been observed, and conversely for $\widehat{P}_t$.

B. Triangulation

Because we have not yet seen the part of the environment that was sampled by $\widetilde{P}_t$, all the information available on this region the environment is, at this point, contained in $\widetilde{P}_t$. We will thus simply directly triangulate those points.

To this end, we consider a set of candidate triangles C_t, obtained by applying a 2-triangle-per-pixel pattern to P_t (see Fig. 2); the set of candidates is given by $C_t = \{\triangle P_t(x, y)P_t(x, y + 1)P_t(x + 1, y) : (x, y) \in \mathcal{D}_{h-1,w-1}\} \bigcup \{\triangle P_t(x + 1, y)P_t(x, y + 1)P_t(x + 1, y + 1) : (x, y) \in \mathcal{D}_{h-1,w-1}\}$. We chose this pattern for no other reason than simplicity.

As with P_t, we partition the candidate triangles C_t in 2 sets $\widehat{C_t}$ and $\widetilde{C_t}$, which contain, respectively, the candidate triangles with at least one vertex in $\widehat{P}_t$, and those with none

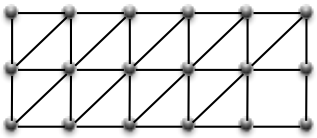

Fig. 2. Triangulation pattern.

(i.e. all 3 vertices in $\widetilde{P}_t$). Intuitively, $\widetilde{C_t}$ is the portion of the environment that has not yet been observed and we can triangulate.

For each triangle candidate $\triangle C_1C_2C_3 \in \widetilde{C_t}$ we compute its normal $\vec{n}$, by taking the cross product of two of its edges; to guarantee the normal is oriented towards the sensor, we compute $\vec{n} = \overrightarrow{P_t(x, y)P_t(x, y + 1)} \times \overrightarrow{P_t(x, y)P_t(x + 1, y)}$ for triangles of the form $\triangle P_t(x, y)P_t(x, y + 1)P_t(x + 1, y)$, and $\vec{n} = \overrightarrow{P_t(x, y + 1)P_t(x + 1, y + 1)} \times \overrightarrow{P_t(x, y + 1)P_t(x + 1, y)}$ for triangles of the form $\triangle P_t(x, y+1)P_t(x+1, y+1)P_t(x+1, y)$.

The angle of incidence ι is then taken as $\iota = \arccos\left(\frac{\vec{n}}{\|\vec{n}\|_2} \cdot \frac{\sum_{i=1}^3 \overrightarrow{C_iO_t}}{\|\sum_{i=1}^3 \overrightarrow{C_iO_t}\|_2}\right)$. As discussed above, we now add the triangle $\triangle C_1C_2C_3$ to F_t if and only if its angle of incidence is less than a maximum angle of incidence ι_{max}, that is if we have $\iota \leq \iota_{max}$. This filters out legitimate triangles sampled from regions of the environment that are angled away from the camera, but also rejects neighboring samples that were indeed not connected. The ι_{max} threshold should be adjusted to reduce the two types of error. The advantage of updating the surface continuously is that when a better viewpoint is available, the error will be corrected.

This procedure incrementally builds a mesh from an empty start, by creating new connected components when parts of the environment are newly observed. However, estimates whose ray intersected the existing mesh are not used during the triangulation step.

IV. PPM updating

Intersecting estimates – points in $\widehat{P}_t$ – are instead used to update the existing surface reconstruction E_{t-1}. This is done through an iterative procedure that considers each point in $\widehat{P}_t$ sequentially. In the rest of this section, we will describe this procedure for an arbitrary estimate point $P \in \widehat{P}_t$ with pixel coordinates $(x, y) \in \mathcal{D}_{h,w}$.

The distance measurement $O_tI_t(x, y)$ is what would be expected if the current model E_{t-1} were perfectly accurate, but the value returned by the sensor is O_tP. The difference between those quantities is due to both the error of the current reconstructed surface E_{t-1} in modeling the real surface, and the noise from the sampling of the surface's distance by the ranging sensor.

Let us assume the range imaging sensor follows a zero-mean Gaussian stochastic observation model; that is, for a real distance to the surface d, it can be modeled by a random variable $O \sim \mathcal{N}(d, \sigma)$, where σ is the sensor variance. This is an assumption we can get closer to if necessary by using

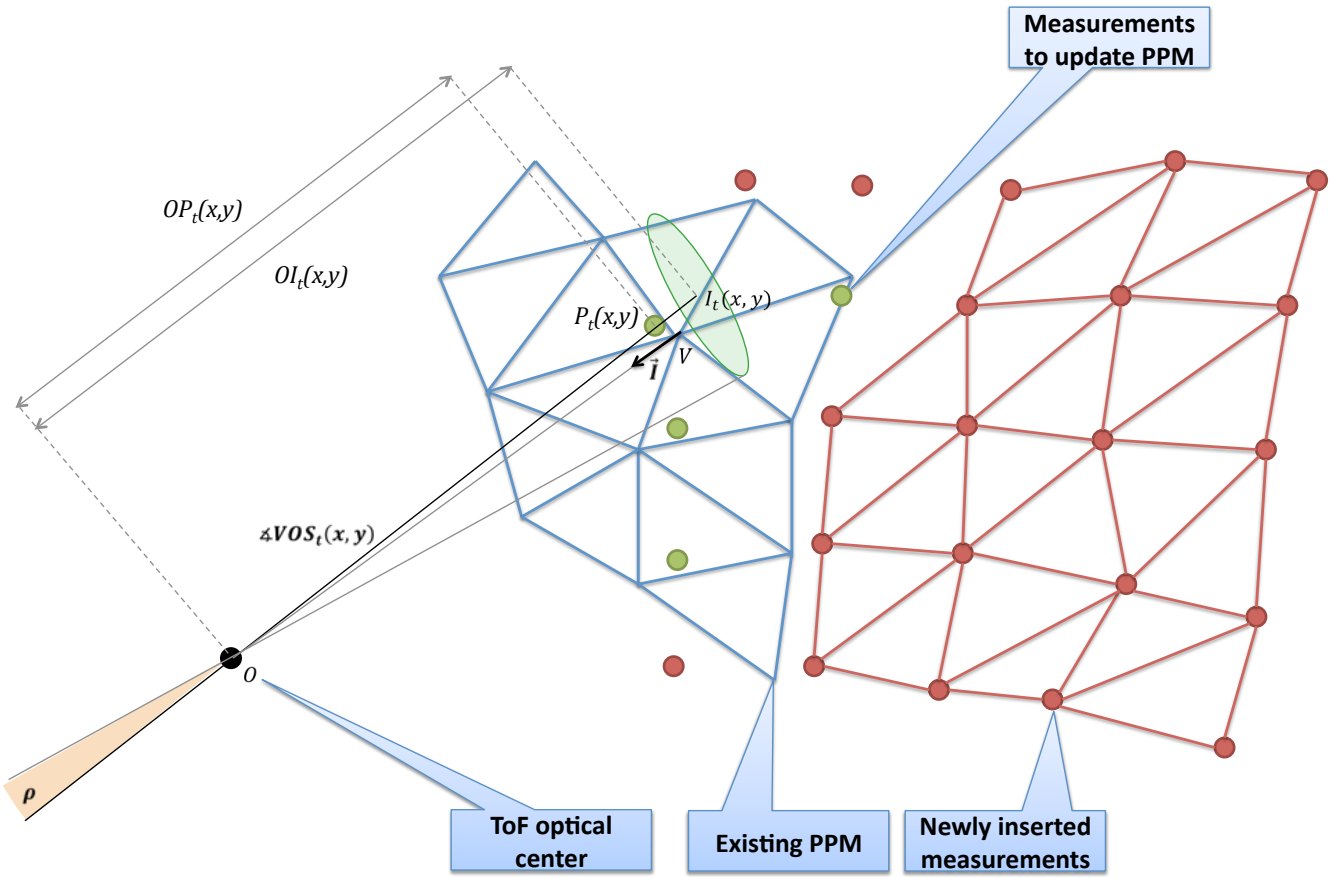

Fig. 3. Overview of update quantities.

an offline per-pixel calibration method such as estimating a Fixed Pattern Noise matrix for the sensor [7].

We can then estimate the real distance d from a sequence of samples by creating a Maximum-Likelihood (ML) estimator: assuming independent estimates, the ML estimator of d is the expected value $\mathbb{E}[O] = d$. However, even with a static environment, the position of the optical center may vary. We assume that when observing a vertex multiple times, by averaging the sequentially estimated vertex positions measured from potentially different viewpoints, the expected value of the estimates will be the true vertex' position. This is asymptotically true in that if the reconstructed surface E_{t-1} is precise enough, $I_t(x, y)$ coincides with the real point it models.

A. Geometric aspect

Because we cannot in the general case easily move the surface of E_{t-1} at $I_t(x, y)$, we update the vertex V of the face containing $I_t(x, y)$ closest to $I_t(x, y)$ (see Fig. 3 to visualize the quantities mentioned in this section). However, because the viewpoint can vary over time, P may not even contain any information about V, as it may lie in the middle of a triangle that was created at a larger resolution. We thus mitigate this by introducing a measure of the overlap of the optical frustum corresponding to the pixel (x, y) with V.

This quantity $\gamma \in [0, 1]$ we call angular gain depends on the angle $\alpha = \angle VO_tP$ and the angular resolution of the range imaging sensor ρ, and is defined by $\gamma = 1 - min(\alpha/\rho, 1)$. For example $\rho = 0.24°$ for the SwissRanger SR4000 camera. γ is a gain in the sense it is a number between 0 and 1 used to modulate how much an estimate will be incorporated as a new estimate for a vertex. Physically, the value of each pixel does not only measure the distance along a line that passes through the center of the pixel, the thin-beam model, but rather integrates the distances over a frustum around this thin-beam. We notice that if $\alpha > \rho$, $\gamma = 0$, which means that when the estimate's conical frustum does not contain V at all, the estimate has no impact on V; conversely, if $\alpha = 0$, $\gamma = 1$ and the estimate is fully used to update the position of V.

The difference between the expected distance and the sampled measurement is used to compute what we call the innovation vector $\vec{I}$, which is given by $\vec{I} = \left(\overline{OP} - \overline{OI_t(x, y)}\right) \frac{\overrightarrow{OV}}{\|\overrightarrow{OV}\|_2}$ (where the over-bar denotes an algebraic distance). The innovation vector is collinear to $\overrightarrow{OV}$, and its magnitude equal to the difference in measured depth.

B. Maximum-likelihood estimation

To carry out the Maximum-Likelihood averaging, we wish to avoid storing the history of innovation vectors and angular gains for each vertex in the PPM, as the number of vertex

may rise into the hundreds of thousands. Additionally, we wish to compute an online incremental reconstruction with updated vertex positions after each scan S_t.

We thus turn to implementing an online-ML scheme at low cost by exploiting the recursive identity $X_n = \sum_{i=1}^{n} a_i x_i = X_{n-1} + \frac{a_n}{\sum_{i=1}^{n-1} a_i + a_n}(x_n - X_{n-1})$ [8]. Indeed, this identity enables us to update the weighted average of a sequence of values $(x_i)_{i\in\{1,\dots,n\}}$ with weights $(a_i)_{i\in\{1,\dots,n\}}$, from only the previous online average (X_{n-1}) and summed weights $(\sum_{i=1}^{n-1} a_i)$, and the latest weight (a_n) and value (x_n).

In our case, we apply this scheme on a succession of values given by the history of innovation vectors $\left(\overrightarrow{I_i}\right)_{i\in\{1,\dots,n\}}$, and weights given by the history of angular gains $(\gamma_i)_{i\in\{1,\dots,n\}}$. We do this by maintaining for each vertex, on top of the running average position V, a number we call the effective number of observations $\Gamma_n = \sum_{i=1}^{n} \gamma_n$. When a vertex is first created, it is not updated but added as part of a new triangle, and thus needs special initialization $\Gamma_1 = \gamma_1 = 1$.

The update of V_{n-1} when estimating the latest innovation vector $\overrightarrow{I_n}$ and angular gain γ_n is then given by:

$$V_n = V_{n-1} + \frac{\gamma_n}{\Gamma_{n-1} + \gamma_n}\overrightarrow{I_n} \tag{2}$$

Note that for an effective number of observations Γ, we can retrieve the variance σ_V of the probabilistic belief described in section II-B for vertex V, by using the sensor variance σ (introduced in section IV), taking $\frac{1}{\sigma_V^2} = \frac{\Gamma}{\sigma^2}$.

V. Summary of algorithm

1) Cast a ray from the camera center O_t through each ToF measurement $P_t(x, y)$; intersections are $I_t(x, y)$.
2) Create a new mesh with all non-intersecting measurements, filtering out angles of incidence ι higher than ι_{max}.
3) Find vertex V closest to each ray intersection $I_t(x, y)$ part of its facet. Compute then innovation vector $\vec{I}$, collinear to vector $\overrightarrow{O_tV}$, proportional to measurement error $\overline{O_tP_t(x, y)} - \overline{O_tI_t(x, y)}$ with gain $\gamma = \frac{\angle VO_tP_t(x,y)}{\rho}$, where ρ is the time of flight camera's angular resolution.
4) Adjust V by translating by $\frac{\gamma_n}{\Gamma_{n-1}+\gamma_n}\overrightarrow{I_n}$.

VI. Implementation

We implemented our PPM algorithm on our custom-made mobile robot named Kappa, which is equipped with a Swiss Ranger ToF camera as a distance sensor (see Fig. 4). The code is written as a C++ ROS node for portability to other systems. Mesh processing was performed on an off-board computer with an Intel Xeon 2.4 GHz processor, though no parallel processing was used. This is notable, as the presented algorithm relies on parallelizable ray-casting and incremental update. We built upon the CGAL polyhedron class for mesh representation, with an AABB tree to optimize the ray tracing step. The source code is open source and available as a Mercurial repository at http://winnie.kuis.kyoto-u.ac.jp/-kenzo/kappa/source.

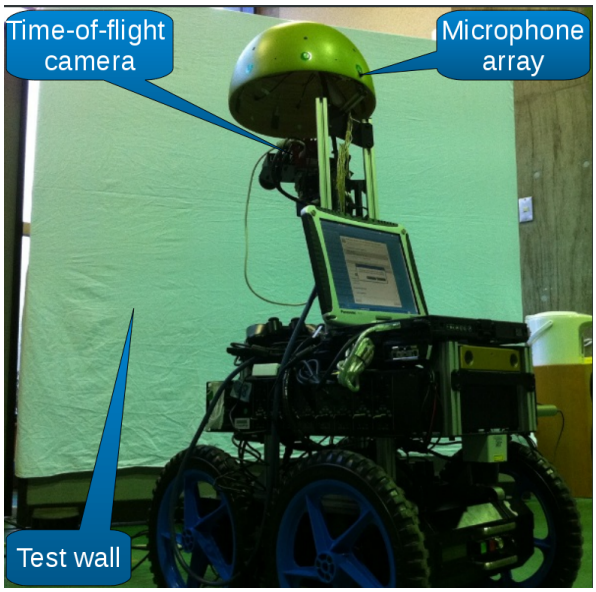

Fig. 4. Experimental setup, with robot facing white wall.

VII. Evaluation

Our main purpose is 3D surface reconstruction from a set of 3D range images. A 3D range image can be regarded to a point cloud by ignoring the optical center and considering the 3D point associated with each pixel. It follows that a set of 3D range images can be considered as one large point cloud by taking the union of the points in each range image, assuming they're registered in a global coordinate system. Following this procedure, we can evaluate our algorithm in the same way 3D reconstruction from point cloud is evaluated. Note that this ignores the additional information contained in the mesh compared to a point cloud (extracted surface, normals, variance) as well the persistent, incremental nature of the PPM, which would allow keeping scene understanding results associated with the mesh over time.

A. Experimental setup

In our experiment, we placed a white wall approximately 1.3 m away from the robot, and faced the ToF camera towards it such that the image filled the sensor's field of view entirely (Fig. 4). The robot remained still during the entire experiment. Our goal here is to measure how planar the PPM model becomes, versus how planar the raw data is.

We recorded 200 frames of ToF camera images, and tested the two methods as follows: 1) extract the mesh generated by raw point cloud data or the PPM; 2) fit a plane to the vertices of the mesh using linear least squares; 3) extract the correlation coefficient between the mesh and the plane.

B. Results

The results are plotted in Fig. 6. The higher the coefficient, the better the mesh fits the plane. Note the quick increase

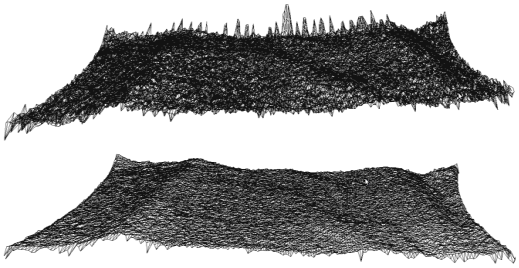

Fig. 5. Example comparing directly triangulated data (above) with a PPM built over several scans (below).

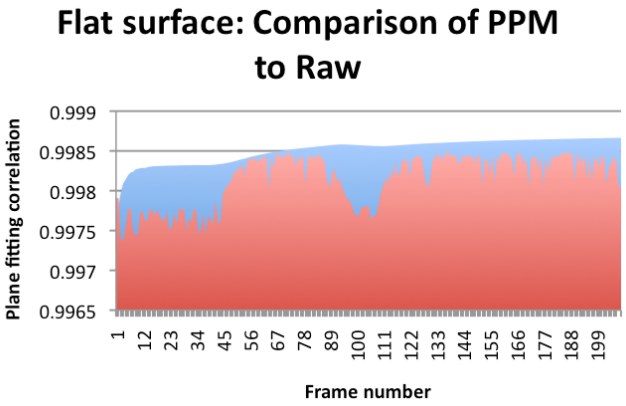

Fig. 6. Comparison of raw and PPM correlation with linear-least squares best-fitting plane.

and stability of the PPM correlation over the lower and more erratic precision of the raw point clouds. Using this simple reconstruction metric we can observe that within about 5-6 frames (approximately 2s), the mesh accuracy stabilizes; this is with our unparallelized implementation. In addition, the PPM accuracy is higher than that of the raw data at all times. An example comparing directly triangulated raw data and the corresponding PPM is shown in Fig. 5, and a real world example can be seen in Fig. 7. A more thorough evaluation would consist of more complex, real environments with metrics such as one presented in [9].

VIII. Discussion and Future work

Our main application is to create a rich, multi-modal representation of the world as a base data type for robot intelligence. The geometric mesh described in this paper can be used as the base layer onto which we can attach other information, such as color, texture, and sound. This multi-modal scene representation may serve as a new primitive data type, replacing raw video images or range scans as a basis for recognition. One application in particular is sound source reflection estimation. Azimuthal localization of sound-sources from multiple viewpoints can be used to find their location as

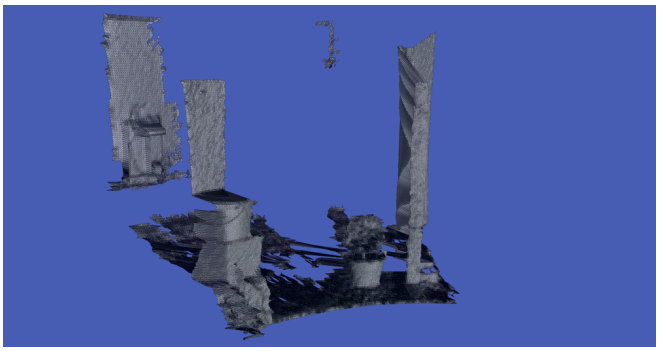

Fig. 7. Example of PPM built from a real world scene.

per [10]; however in real-world settings, auditive reflections are a problem. Using the physical geometry of the world, we can estimate whether a sound comes directly from a human, for example, or is simply a refraction of sound off surfaces.

There is still much work to be done in order for the PPM to be used for mobile robot scene reconstruction. We plan to merge data from multiple viewpoints using ICP, for example to recreate a 3D object online. In case of dynamic data such as moving humans, we also need to add a hole-punching step to remove improbable data when newer data becomes available. Once the geometric mapping is complete, the addition of color data, texture, and sound is of high priority.

References

[1] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich, "The vSLAM Algorithm for Robust Localization and Mapping," *International Conference on Robotics and Automation (ICRA)*, pp. 24–29, 2005.

[2] J. Nieto, J. Guivant, and E. Nebot, "DenseSLAM: Simultaneous Localization and Dense Mapping," *International Journal of Robotics Research*, vol. 25, pp. 711–744, Aug. 2006.

[3] R. A. Newcombe and A. J. Davison, "Live Dense Reconstruction with a Single Moving Camera," in *Computer Vision and Pattern Recognition (CVPR)*, pp. 1498 – 1505, 2010.

[4] Z. C. Marton, R. B. Rusu, and M. Beetz, "On fast surface reconstruction methods for large and noisy point clouds," *International Conference on Robotics and Automation (ICRA)*, pp. 3218–3223, May 2009.

[5] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," in *Symposium on Geometry Processing (SGP)*, pp. 61–70, 2006.

[6] K. Pathak and A. Birk, "Sub-pixel depth accuracy with a time of flight sensor using multimodal gaussian analysis," *Intelligent RObots and Systems (IROS)*, vol. 1, pp. 22–26, 2008.

[7] T. Kahlmann, F. Remondino, and H. Ingensand, "Calibration for increased accuracy of the range imaging camera swissranger," *Image Engineering and Vision Metrology*, no. 4, pp. 136–141, 2006.

[8] C. M. Bishop, "Chapter 2: Probability Distributions," in *Pattern Recognition and Machine Learning*, vol. 248, ch. 2, pp. 67–136, Springer, Oct. 2006.

[9] B. Barshan, "Objective Error Criterion for Evaluation of Mapping Accuracy Based on Sensor Time-of-Flight Measurements," *Sensors*, vol. 8, pp. 8248–8261, Dec. 2008.

[10] H. Asoh, F. Asano, T. Yoshimura, K. Yamamoto, Y. Motomura, N. Ichimura, I. Hara, and J. Ogata, "An application of a particle filter to bayesian multiple sound source tracking with audio and video information fusion," in *International Conference on Information Fusion (ICIF)*, pp. 805–812, 2004.