Fast Incremental Probabilistic Surface Reconstruction for Robot Scene Understanding

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

Introduction

Robots need a rich, multi-modal representation of the world from which to learn in a generalized manner. Indeed, we now have at our disposal methods for sensing geometric, visual and auditory information, all spatialized in the same global reference frame and bound together through techniques such as SLAM [1]. For humans, those modalities are the dominant forms of information. Robots should also take advantage of these tools for moving around in and interacting with the physical world (geometry), recognizing common objects (vision), and communicating with humans (audition).

All of these sources of information need to be linked together in space and in time. In this paper, we describe a geometric mesh which can be used as the base layer onto which we can attach other information, such as color, texture, and sound. Our ultimate goal is to use this multi-modal scene representation as a new primitive data type, replacing raw video images or range scans as a basis for recognition.

Our main source of information is the relatively new technology of Time-of-Flight (ToF) cameras. These optical cameras measure the depth of each pixel, and combined with a traditional color camera, this type of sensor provides Red-Green-Blue-Depth (RGBD) images at high frequencies. With the mass-produced Microsoft Kinect introduced in late 2010, 30Hz $640 * 480$ 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.

As a sensor itself, time-of-flight technology has several advantages and disadvantages. At around 10 frames per second, it is possible to collect 3D data of a scene much faster than a pan-tilt laser scan (such as used in [2]). This increase in speed is significant especially for scene reconstruction for mobile robots, which cannot wait the 5-10s required to obtain one 3D scan from a laser, or the long processing time of more elaborate batch reconstruction methods such as the Poisson reconstruction algorithm introduced by Kazhan and Hoppe [3]. The major drawback of TOF technology is its high frame-to-frame 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.

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.

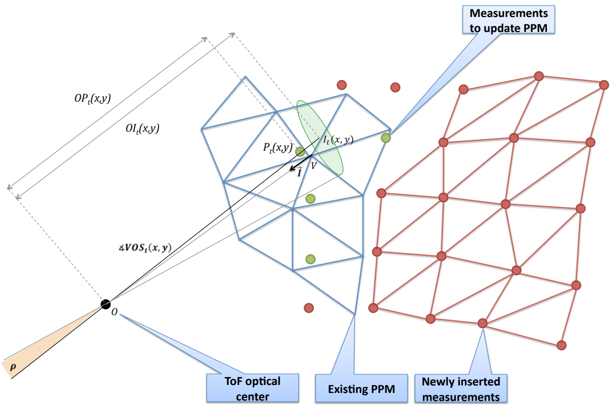

Fig.1: Overview of update quantities.

1. Probabilistic Polygonal Mesh

Let us suppose we receive 2.5D scans at a sequence of discrete times $T = (t_1, \ldots, t_n)$. 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, as shown in Fig. 1. The second element P_t is the matrix of point estimates at time t, in the form of a matrix of 3D points. We will note $\mathcal{D}_{h,w} = \{1, \ldots, h\} \times \{1, \ldots, w\}$ 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 (pixel value).

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.

1·1 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 same arity as V_t.

For an index $i \in \{1, \ldots, |V_t|\}$, $\pi_i(F_t)$ is an approximation of the local Gaussian variance of the belief (ie. uncertainty) in the current surface reconstruction E_t, in the neighborhood of $\pi_i(V_t)$.

1·2 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 $\iiint\limits_{\mathbb{R}^3} \mathcal{B}_t(P)\, \mathrm{d}P = 1$.

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 colinear, as this makes it a system of 3 independent equations in 3 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:

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

The normalized belief $\mathcal{B}_t$ is obtained by normalizing $\widetilde{\mathcal{B}_t}$: $\mathcal{B}_t(P) = \frac{\widetilde{\mathcal{B}_t}(P)}{\iiint\limits_{\mathbb{R}^3} \widetilde{\mathcal{B}_t}(Q)\, \mathrm{d}Q}$. $\mathcal{B}_t$ 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 $\mathcal{S}(E_t)$.

2. 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 $t \in T$.

2·1 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. 1 to visualize). Formally, for each pixel $(x, y) \in \mathcal{D}_{h,w}$, if we note $\mathring{I}_t(x,y) = \underbrace{\left\{ P \in \mathbb{R}^3 : \exists \lambda \in \mathbb{R}_+^*, P = O_t + \lambda \overrightarrow{O_t P_t(x,y)} \right\}}_{ray} \bigcap \mathcal{S}(E_{t-1})$ 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 $\widehat{P_t}$.

2·2 Triangulation

The triangulation procedure incrementally builds a mesh from an empty start, by creating new connected components when parts of the environment are newly observed, as shown in Fig. 1, "newly inserted measurements".

We consider a set of candidate triangles C_t, obtained by applying a 2-triangle-per-pixel pattern to P_t (see Fig. 1); 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 (ie. 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 thus compute its normal $\vec{n}$, by taking the cross product of 2 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 $\iota \leq \iota_{max}$.

3. PPM updating

Estimates whose ray intersected the existing mesh are not triangulated, but used for update, noted by "existing PPM" in Fig. 1. This is done through an iterative procedure that sequentially considers each point in the set of intersecting points $\widehat{P_t}$. The idea here is that P brings new information about a small region of the environment's surface that was already mapped in E_{t-1}, and that we should locally integrate this information to create the updated surface reconstruction E_t.In the rest of this section, we will describe this procedure for an arbitrary estimate point $P \in \widehat{P_t}$ with pixel coordinates $(x, y) \in \mathcal{D}_{h,w}$.

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. We can 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 $\mathbb{E}[O] = d$.

3·1 Vertex weighting

An new estimate will update a vertex proportionally to two values: 1) the relative distance to the nearest vertex and 2) the resolution of the camera at that point. This "relative angular distance" is a ratio $\gamma \in [0, 1]$ we call angular gain which 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. 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 $\vec{I} = \left(\overline{OP} - \overline{OI_t(x,y)}\right)\frac{\overrightarrow{OV}}{\|OV\|_2}$ (where the over-bar denotes an algebraic distance). This innovation vector is used to move the vertex to its new position.

3·2 Maximum-likelihood estimation

To carry out the ML averaging, we wish to avoid storing the history of innovation vectors and angular gains for each vertex in the PPM, as the number of vertices may rise into the hundreds of thousands. We thus turn to implementing an online-ML scheme at low cost by exploiting the recursive identity $X_n = \sum_{i=1}^{n} a_ix_i = X_{n-1} + \frac{a_n}{\sum_{i=1}^{n-1} a_i + a_n}(x_n - X_{n-1})$ [6]. Indeed, this identity enables us to update the weighted average of a sequence of values $(x_i)_{i\in\{1,\ldots,n\}}$ with weights $(a_i)_{i\in\{1,\ldots,n\}}$, 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 $\left(\overrightarrow{I_i}\right)_{i\in\{1,\ldots,n\}}$, and weights given by the history of angular gains $(\gamma_i)_{i\in\{1,\ldots,n\}}$. 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:

$$lV_n = V_{n-1} + \frac{\gamma_n}{\Gamma_{n-1} + \gamma_n}\overrightarrow{I_n} \qquad (2)$$

Note that for an effective number of observations Γ, we can retrieve the variance σ_V of the probabilistic belief described in section 1·2 for vertex V, by using the sensor variance σ (introduced in section 3.), taking $\frac{1}{\sigma_V^2} = \frac{\Gamma}{\sigma^2}$.

4. 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. 2). The code is written as a C++ ROS node for portability to other systems. Mesh processing was performed on an off-board computer with two Intel Xeon Quadcores 2.4 GHz, though no parallel processing was used in particular for multicore optimization. This is notable, as the presented algorithm relies on parallelizable ray-casting and incremental update. We used an enriched version of

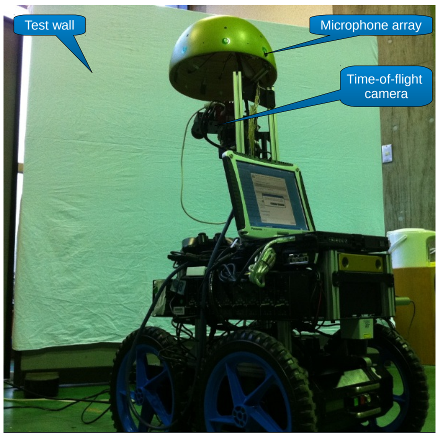

Fig.2: Experimental setup, with robot facing white wall.

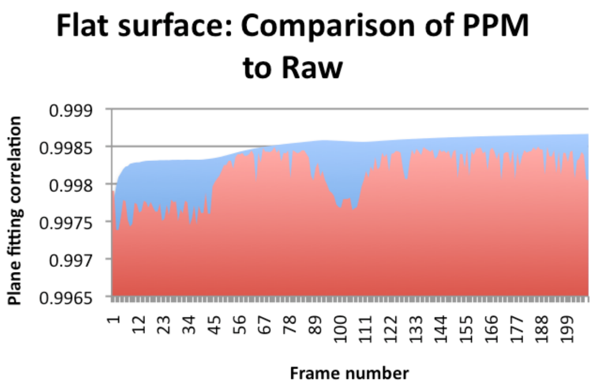

Fig.3: Comparison of raw data (in red) and PPM (in blue) over time, showing better accuracy and stability of our method for all time frames.

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.

5. 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.

5·1 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. 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.

5·2 Results

The results are plotted in Fig. 3. The higher the coefficient, the better the mesh fits the plane. Note the quick increase 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. A real world example can be seen in Fig. 4. A more thorough evaluation would consist of more complex, real environments with metrics such as one presented in [7].

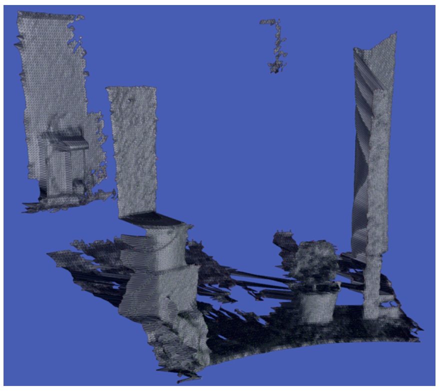

Fig.4: Example of PPM built from a real world scene.

6. Future work

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," *ICRA*, pp. 24–29, 2005.

[2] Z. C. Marton, R. B. Rusu, and M. Beetz, "On fast surface reconstruction methods for large and noisy point clouds," *ICRA*, pp. 3218–3223, May 2009.

[3] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," 2006.

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

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

[6] C. M. Bishop, *Chapter 2: Probability Distributions*, vol. 248, ch. 2, pp. 67–136. Springer, Oct. 2006.

[7] 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.