An Adaptive Framework for 3D Data Fusion with Sensors' Error Prior

Wei Dong, Xin Wang, Renju Li, Hongbin Zha
Key Laboratory of Machine Perception, Peking University

Abstract

We propose a generative probabilistic framework to fuse sequential 3D data from various sensors of static scenes. With the explicit consideration of sensors' error prior, we assign a correspondent error model for each data point. This abstraction guarantees the adaptivity of the framework to the types of sensors, since data with diverse properties are represented in a consistent form. Moreover, the data-level uncertainty modeling simplifies the process of fusion; redesigning a sensor-level probability graphical model (PGM) is not required when involving a new sensor. Fitting in a generative model, the error prior per point leads to less ad hoc probability functions in an incremental Bayesian updating strategy. From the probability map where data are fused, we extract surfaces via directional derivative computations; the method we utilize can be regarded as a generalization of several existing methods. Experiments demonstrate that our framework is capable of fusing indoor data from a moving Kinect, and dealing with data from LiDARs and stereo cameras with different characteristics in a large scale, resulting in the improvement of surface quality.

1. Introduction

Nowadays, 3D data are becoming indispensable demands from laboratories to homes. They are required by Augmented Reality (AR) for virtual object rendering, sweeping robots for route arrangement, self-driving cars for safe driving, and cultural heritages for digital copies that is immune to the erosion of time. Unfortunately, limited by the industry level of human beings, there is no such almighty 3D sensor that is flexible for all scenes. Instead, various sensors have been developed for specific tasks. Including data density, working distance, and accuracy, the characteristics of such sensors vary.

In most situations, 3D data collected at *one* viewpoint by *one* sensor are partial in space and restricted by the sensor's property, hence hardly useful. Inevitably, we need to integrate 3D data from separate viewpoints for a more comprehensive view; there are also scenarios where data from multiple sensors should be merged, *e.g.*, the scenes that are too complicated for merely one sensor, and the scenes that change through time requiring additional 3D scanning via different sensors. Many studies have been focusing on the subject of fusing 3D data measured at various viewpoints by different sensors, and have achieved remarkable results. Yet there is still a lack of an adaptive framework to deal with data error caused by sensors. Some solutions are general for 3D data, but they treat data points equally without discriminating the sources of error, or involve ad hoc computations, causing weak flexibility to the sensors' properties; others do specify the error from sensors and build PGMs upon error functions, but receive only certain types of input, reducing their adaptivity to sensors.

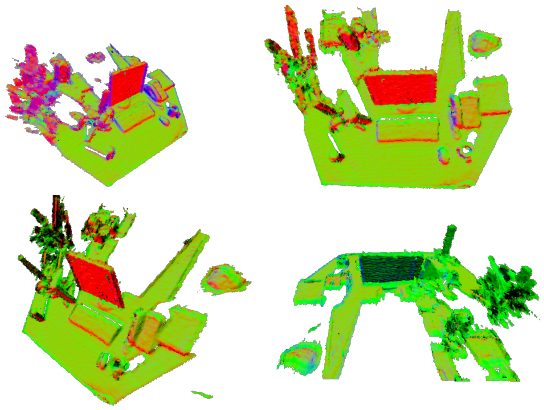

Figure 1. Normal map with pseudo-color at different viewpoints of the reconstructed scene in TUM fr2-xyz.

In this paper we build a framework to fuse 3D data acquired in static scenes by considering the error prior of every 3D point. On the one hand, we discriminate data points by specifying their error models; on the other hand, we merge them together with the same strategy providing a consistent, error-involved representation. To fulfill this, the framework uses a generative probabilistic model that simulates the process of scanning where different error distributions of data points are handled. As we model error and operate probability calculation at the data-level, we do not have to design sensor-level PGMs; only minor adjust-

ments will enable our framework to receive data from a new sensor. By choosing appropriate searching ranges and adopting a dynamic programming algorithm, our incremental Bayesian updating strategy is simple yet efficient. A surface extraction strategy is also developed to reconstruct the scene from the probability map where data are fused. Interpreted as a generalization of existing surface extraction methods from volumes, the algorithm generates surfaces of considerable quality.

2. Related Work

Distance field has been a mature solution to fuse 3D data without explicitly considering data uncertainties. Curless proposed the signed distance function (SDF) to represent a field in the 3D space [3]. He discretized the space into voxels and approximated the distance from each voxel to its closest plane. Utilizing modified SDF, Newcombe *et al*. proposed KinectFison and achieved remarkable results on static desktop-scale scenes with a moving Kinect [15]. This research was followed by a solution to reconstruct dynamic scenes with an almost static Kinect [14]. Whelan *et al*. extended the working range of KinectFusion to a large indoor scene's scale by adopting a strategy to move volumes around [22]. Hu and Mordohai attempted to introduce probability to SDF by embedding the distance into a probability energy function [9]. These online systems have achieved good results on surface reconstruction. They were mostly tested in indoor scenes, and did not explicitly involve the uncertainties from the data.

Meanwhile, probabilistic approaches that consider data error in fusion can be classified into *inverse models* and *generative models* by how the probability functions are defined. Konolige used an inverse model to infer the uncertainty of a 2D map's occupancy state from the sensor readings [12]. He increased the probability values in the regions where observed points hit, while decreased the values in the rest. Kim *et al*. fused 3D data from ToF and stereo cameras with an inverse probability function that integrates multiple energy functions [10]. Hornung *et al*. proposed OctoMap for general 3D data with a simple inverse model [8]. These methods are usually simple and efficient. However, sometimes they are to some extent ad hoc and require parameter adjustments according to the scenarios.

Thrun introduces a generative model for 2D data fusion [20]. He considered the process of data generation, updated the occupancy state of the map through time-consuming EM iterations and computed the probability for regions accordingly. Pathek *et al*. extended this approach to 3D data and simplified it with a beam model; some ad hoc calculations were involved [18]. Woodford *et al*. proposed an improved 3D version of Thrun's solution to deal with outliers explicitly [23]. They employed an outlier ratio term to reflect the confidence of data's accuracy. Yet sensors' error in reading was not counted in such generative methods.

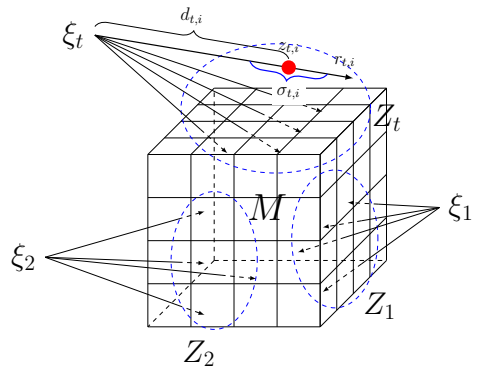

Figure 2. Illustration of major concepts and terms. M stands for the map, ξ_t for the sensors' position, Z_t for the collected data set, $z_{t,i}$ for the individual scanned points. r, d, σ are futhur discussed in 3.1.

Kolev *et al*. fused 3D data with high uncertainties from a moving monocular camera [11], including factors such as angle, distortion and distance. Ulusoy *et al*. considered ray potential during the formation of images and fused data in a space-carving fashion [21]. They carefully designed an MRF that incorporated several prior error functions. The consideration of error sources in both methods helped to improve the data quality for RGB image fusion; a generalization for other sensors was not discussed.

Cabezas *et al*. built a PGM to fuse multi-modal data from LiDAR and camera plus GPS, along with selected error models [1]. Marin *et al*. integrated data from a ToF and a stereo camera at the level of depth-image generation [13]. These approaches directly take error prior into account, but lacked the adaptivity to sensors; it might require much effort for them to adapt to a new kind of sensor.

3. 3D Data Fusion with Sensors' Error Prior

3.1. Major Concepts

In general, most contactless sensors measure the distances from a detected object to a sensor along scanning rays, *e.g*., ToF sensors. A stereo camera (built with a monocular camera pair) can also be interpreted as such a scanner: a traditional camera approximates a pinhole model with pixels corresponding to rays, hence a generated disparity image (or depth image) shares the same property.

The readings of distance from 'ray-based' sensors suffer from error, more or less, due to the hardware or software deficiencies. Comparatively, rays are accurate in direction; rectification techniques can further compensate error on ray directions for a lot of sensors. In this paper, we focus on these 'ray-based' sensors, assuming that data uncertainties

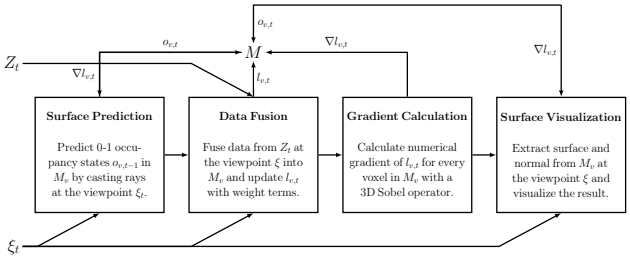

Figure 3. The pipeline of data fusion.

come from the reading error along the rays, while the ray directions are accurate. In addition, readings on different rays are concerned to be independent with each other.

Having specified the assumptions of the sensors and error models, we define two major concepts in our framework: *observation* and *map*. Figure 2 depicts these relevant terms.

One observation Z_t represents a bunch of inter-independently observed 3D points $z_{t,i}$ acquired by a 3D sensor at the timestamp t. We use $\xi_t \in se3$ to represent the rigid transformation from the sensor's coordinate system to the world's, and $T(\xi) \in SE3$ to denote the 4×4 matrix form. ξ is assumed to be known in this paper, since our focus is on data fusion; it can be determined by exterior motion detectors or other mature solutions.

As we have discussed above, every $z_{t,i}$ is acquired along a ray. Each observed 3D point $z_{t,i}$ holds a ray $r_{t,i}$ with its length $d_{t,i}$ in the sensor's coordinate system; most importantly, it stores an uncertainty variance $\sigma_{t,i}$ for a normal distribution to describe the uncertainty along the ray; other than a normal distribution, $\sigma_{t,i}$ can represent a relevant parameter. This error model should be estimated in advance; there are studies that model a sensor's reading error at the certain direction, *i.e.*, $r_{t,i}$, and distance, *i.e.*, $d_{z,i}$ [2, 8, 16]; putting stereo camera aside which needs the analysis of algorithms (software), conducting an experiment to measure the error model of a sensor (hardware) is not complicated.

Every Z_t is fused into an occupancy map M represented in a volumetric form where each voxel is denoted by M_v. 3D sensors we discussed cannot detect the inside of objects, hence we regard 'is occupied' and 'is on surface' for a voxel as equivalent concepts in the context. This definition might clarify misunderstandings.

M_v holds a log odds value when T observations have been fused

$$l_{v,T} = \log \frac{P(M_v \mid M_{-v}; Z_1, \xi_1 \cdots, Z_T, \xi_T)}{P(\bar{M}_v \mid M_{-v}; Z_1, \xi_1 \cdots, Z_T, \xi_T)}, \quad (1)$$

where M_{-v} denotes the whole map excluding M_v. For simplicity, we use $Z_{1:T}$ instead of $Z_1, \cdots, Z_T$; and ξ the same. The reason for choosing log odds and such a distribution is discussed in Section 3.4.

M_v also holds a binary value $o_{v,T}$ indicating the predicted occupancy state of M_v after T observations have been fused. The algorithm determining $o_{v,T}$ is detailed in Section 3.5.

To avoid confusion about whether a notation is an index or a position, we use $x(z_{t,i})$ and $x(M_v)$ to denote the position of an observed 3D point and the center of a voxel in the world coordinate system; $z_{t,i}$ and M_v can be interpreted as names. $x(z_{t,i}) = d_{t,i}T(\xi_t)r_{t,i}$; $x(M_v)$ is determined by the scale of M.

3.2. Pipeline

The pipeline of the fusing process in Figure 3 describes how Z_t is fused into M incrementally.

At the 1st surface extraction stage, we determine the voxels that are most likely to be on the surface at the viewpoint ξ_T before Z_T is fused. For these voxels, $o_{v,T-1}$ are set to 1; others are set to 0. $o_{v,T-1}$ is regarded as the prediction of the real surface from the accumulated knowledge.

At the Bayesian updating stage, we enumerate every M_v influenced by Z_T: the 0-1 state of each M_v is discussed separately according to the generative model by performing a probability computation. This computation updates the stored $l_{v,T}$ per affected voxel, hence influences the entire map M.

The gradient calculation stage is simple: voxel-wise 3D numerical gradient is computed with a Sobel 3D operator; Section 3.5 provides the mathematics of its usage.

Finally, we process surface extraction for a second time, reconstructing the surface after Z_T is fused. Outputs include the newest surface image and the normal image.

3.3. Generative Model

For each observed 3D point $z_{t,i}$, we model how it is generated by the entire map M. For 'ray-based' scanners, every voxel in M along $r_{t,i}$ potentially generates $z_{t,i}$; we push these voxels into an array $\boldsymbol{c}_{t,i} = \{\boldsymbol{c}_{t,i,1}, \cdots, \boldsymbol{c}_{t,i,j}, \cdots\}$ to denote the correspondence to $z_{t,i}$. By introducing these voxels as latent variables, we compute the marginal distribution in Equation (2):

$$P(z_{t,i} \mid M, \xi_t) = \sum_j P(z_{t,i}, \boldsymbol{c}_{t,i,j} \mid M, \xi_t) \quad (2)$$

$$= \sum_j P(z_{t,i} \mid \boldsymbol{c}_{t,i,j}, M, \xi_t) P(\boldsymbol{c}_{t,i,j} \mid M, \xi_t). \quad (3)$$

Figure 4 illustrates the related terms along the ray.

The probability for each latent variable $\boldsymbol{c}_{t,i,j}$ in Equation (3) can be split into two parts. One is the sensor error term

$$P(z_{t,i} \mid \boldsymbol{c}_{t,i,j}, M, \xi_t) \triangleq \int_{\Delta_{t,i}} \mathcal{N}(x(z_{t,i}); x(\boldsymbol{c}_{t,i,j}), \sigma_{t,i}). \quad (4)$$

which takes a voxel as the cause of a sensor's reading, directly involving the error prior of an observed point along a ray. By default, we choose a 1-D normal distribution as the error probability distribution; it can be replaced accordingly if the sensor owns a unique error function. We compute the integral on a small interval $\Delta_{t,i}$ around $z_{t,i}$ to convert the continuous distribution to a probability value. As an approximation, we multiply the value of the error distribution with a coefficient proportional to the interval length. This probability is a simple yet general description of the data's uncertainty.

The other term in Equation (3) is the hit function:

$$\begin{aligned} &P(\boldsymbol{c}_{t,i,j} \mid M, \xi_t) \\ \triangleq &P_{hit}(\boldsymbol{c}_{t,i,j} \mid M, \xi_t) \prod_{k=1}^{j-1} (1 - P_{hit}(\boldsymbol{c}_{t,i,k} \mid M, \xi_t)), \end{aligned} \tag{5}$$

$$P_{hit}(M_v \mid M, \xi) \triangleq \cos\langle x(M_v) - x(\xi_t), n(M_v)\rangle \cdot o_{v,t-1} \tag{6}$$

The hit model describes the probability that the scanning ray hits a certain voxel, *i.e.*, the probability that a voxel really generates the sensor's reading. For the jth voxel along the searching ray, this probability consists of two parts: the scanner has missed the previous 1 to $j-1$ voxels along the ray, before it hits the jth.

In Equation (6), to measure the hit rate per voxel, we consider the predicted binary occupancy state along with the measurement of the angle between the ray and the surface normal. A voxel predicted to be occupied owns the possibility to generate an observed 3D point while a free voxel does not. In practice, if only one voxel along the ray satisfies $o_{v,T-1} = 1$, numerical problems will emerge; we set its neighbors along the ray to be 1 temporarily under such circumstances. As for the hit angle, cos is utilized as a measurement [11]. In radiometry, fixing other variables, the amount of energy casted on a surfel by light is multiplied by a factor of the cos value of the angle between the surfel's normal and the ray direction; similarly, the rate of a scanning ray to hit the voxel is influenced by the angle. The normal $n(M_v)$ can be roughly estimated by the normal computed from $z_{t,i}$ and its neighbors.

We have assumed that $z_{t,i}$s are independent in one observation Z_i. Thus, for the entire observation, we have

$$P(Z_t \mid M, \xi_t) = \prod_i P(z_{t,i} \mid M, \xi_t) \tag{7}$$

3.4. Bayesian Updating Strategy

In Equation (7), the whole map M is on the condition. Directly enumerating the 0-1 state of entire M is too expensive to compute; we prefer separately discussing the state of each voxel. For every M_v, we split M into $\{M_v, M_{-v}\}$, where M_{-v} represents the map excluding M_v.

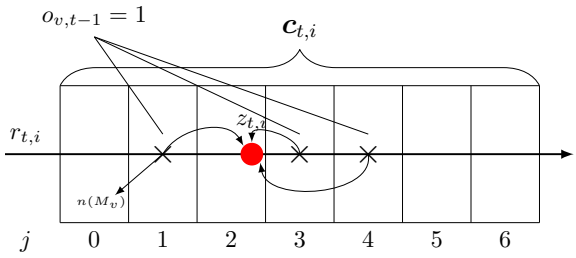

Figure 4. Illustration of terms along a scanning ray in the generative model. $\boldsymbol{c}_{t,i}$ are the voxels that potentially generate $z_{t,i}$ on $r_{t,i}$.

When switching the occupancy state of M_v, we keep M_{-v} unchanged. The form of Equation (3) becomes

$$\begin{aligned} P(z_{t,i} \mid M_v, M_{-v}, \xi_t) = \sum_j &P(z_{t,i} \mid \boldsymbol{c}_{t,i,j}, M_v, M_{-v}, \xi_t) \\ &P(\boldsymbol{c}_{t,i,j} \mid M_v, M_{-v}, \xi_t). \end{aligned} \tag{8}$$

The separation of M_v and M_{-v} divides the whole M into partitions that are easier to compute; by applying the Bayesian rule, a transformation of the equation can describe the occupancy probability. Suppose we have already fused $T-1$ observations, for the Tth observation, according to the Bayesian rule, we have:

$$\begin{aligned} &P(M_v \mid M_{-v}; Z_{1:T}, \xi_{1:T}) \\ =&P(Z_T \mid M_v, M_{-v}, \xi_T; Z_{1:T-1}, \xi_{1:T-1}) \\ \times&\frac{P(M_v \mid M_{-v}; Z_{1:T-1}, \xi_{1:T-1})}{P(Z_T \mid M_{-v}; Z_{1:T-1}, \xi_{1:T-1})}. \end{aligned} \tag{9}$$

This equation can be regarded as an approximation to the occupancy probability of M_v; although not as precise as $P(M_v \mid Z_{1:T}, \xi_{1:T})$, it is more computable in the generative framework, and has been proved to be effective in [20].

The denominator is intractable in Equation (9). We rewrite its dual form:

$$\begin{aligned} &P(\bar{M}_v \mid M_{-v}; Z_{1:T}, \xi_{1:T}) \\ =&P(Z_T \mid \bar{M}_v, M_{-v}, \xi_T; Z_{1:T-1}, \xi_{1:T-1}) \\ \times&\frac{P(\bar{M}_v \mid M_{-v}; Z_{1:T-1}, \xi_{1:T-1})}{P(Z_T \mid M_{-v}; Z_{1:T-1}, \xi_{1:T-1})}, \end{aligned} \tag{10}$$

and then process dividing operation to eliminate the denom-

inator:

$$\frac{P(M_v \mid M_{-v}; Z_{1:T}, \xi_{1:T})}{P(\bar{M}_v \mid M_{-v}; Z_{1:T}, \xi_{1:T})} = \frac{P(Z_T \mid M_v, M_{-v}, \xi_T; Z_{1:T-1}, \xi_{1:T-1})}{P(Z_T \mid \bar{M}_v, M_{-v}, \xi_T; Z_{1:T-1}, \xi_{1:T-1})} \times \frac{P(M_v \mid M_{-v}; Z_{1:T-1}, \xi_{1:T-1})}{P(\bar{M}_v \mid M_{-v}; Z_{1:T-1}, \xi_{1:T-1})} \tag{11}$$

$$= \frac{P(Z_T \mid M_v, M_{-v}, \xi_T) P(M_v \mid M_{-v}; Z_{1:T-1}, \xi_{1:T-1})}{P(Z_T \mid \bar{M}_v, M_{-v}, \xi_T) P(\bar{M}_v \mid M_{-v}; Z_{1:T-1}, \xi_{1:T-1})}. \tag{12}$$

$Z_{1:T-1}$ have been fused into M, thus we have an implicit independency assumption $Z_T \perp Z_{1:T-1} \mid M$ confirming that the assignment between Equation (11) and Equation (12) holds.

We now apply log operation on Equation (12) and create a simple updating formula:

$$l_{v,T} = \log \frac{P(Z_T \mid M_v, M_{-v}, \xi_T)}{P(Z_T \mid \bar{M}_v, M_{-v}, \xi_T)} + l_{v,T-1} \tag{13}$$

$$= \sum_i \log \frac{P(z_{T,i} \mid M_v, M_{-v}, \xi_T)}{P(z_{T,i} \mid \bar{M}_v, M_{-v}, \xi_T)} + l_{v,T-1}. \tag{14}$$

To avoid numerical overflow and increase the robustness, we add a weight coefficient according to [20]:

$$l_{v,T} = \alpha \sum_i \log \frac{P(z_{T,i} \mid M_v, M_{-v}, \xi_T)}{P(z_{T,i} \mid \bar{M}_v, M_{-v}, \xi_T)} + (1-\alpha) l_{v,T-1}. \tag{15}$$

In implementation, we first loop over $z_{T,i}$. For each $z_{T,i}$, we iterate over voxels M_v in the array $\boldsymbol{c}_{T,i}$ along $r_{T,i}$. Firstly, the occupancy state of M_v is set to 1 temporarily, keeping the state of other voxels in $\boldsymbol{c}_{T,i}$ unchanged. Then the sum in Equation (8) is computed, outputting $P(z_{T,i} \mid M_v, M_{-v}, \xi_T)$. Afterwards, the state of M_v is flipped, and the computation of sum is performed again outputting $P(z_{T,i} \mid \bar{M}_v, M_{-v}, \xi_T)$.

The brute force computation requires $O(N^2)$ when there are N voxels in $\boldsymbol{c}_{T,i}$ as it sums over all N voxels in Equation (8). To make the computation more efficient, we use a simple dynamic programming technique. It first stores $\sum_{j=1}^{n} P(z_{t,i}, \boldsymbol{c}_{t,i,j} \mid M, \xi_t)$ for every $n \leq N$, then quickly computes Equation (8) for each M_v by attaching one miss/hit factor to simulate the change of a voxel's occupancy state along with its impact to the voxels it occludes. This will reduce the time to $O(N)$. Detailed algorithm is described in Algorithm 1.

3.5. Surface Extraction

In the probability map maintained for T frames, we assume that the surface lies on the voxels with the max probability, or the max log odds, *i.e.*,

$$\|\nabla l_{v,T}\| = 0. \tag{16}$$

This computation, however, is not numerically stable and filters too many potential surface points. Alternatively, we use ray-casting for surface determination which is compat-

Algorithm 1 DP Bayesian updating

procedure DP UPDATING($z_{t,i}, r_{t,i}, l_M, o_M$)
  ▷ N is the number of searched voxels
  $s[N+1] \leftarrow 0$ ▷ s stores the sum of subarrays
  $p[N+1] \leftarrow 0$ ▷ p stores the probabilities
  $v[N+1] \leftarrow nil$ ▷ v stores indices of found voxels
  for j = 1 to N **do** ▷ Precompute sum of subarrays
    Find the j-th voxel $\boldsymbol{c}_{t,i,j}$ along $r_{t,i}$
    $v[j] \leftarrow \boldsymbol{c}_{t,i,j}$
    $p[j] \leftarrow P(z_{t,i}, \boldsymbol{c}_{t,i,j} \mid M, \xi_t)$
    $s[j] \leftarrow s[j-1] + p[j]$
  end for
  for j = 1 to N **do** ▷ Update l_v
    if $o_{v[j]} = 1$ **then** ▷ Influence of flipping $o_{v[j]}$
      $f_1 \leftarrow 1$
      $f_0 \leftarrow 1/(1 - P_{hit}(M_{v[j]} \mid M, \xi_t))$
    else
      $f_1 \leftarrow 1 - P_{hit}(M_{v[j]} \mid M, \xi_t)$
      $f_0 \leftarrow 1$
    end if
    $P(z_{t,i} \mid M_{v[j]}, M_{-v[j]})$
    $\leftarrow s[j-1] + p[j] + (s[N] - s[j]) \times f_1$
    $P(z_{t,i} \mid \bar{M}_{v[j]}, M_{-v[j]})$
    $\leftarrow s[j-1] + (s[N] - s[j]) \times f_0$
    $l_{v[j]} \leftarrow l_{v[j]} + \log(P(z_{t,i} \mid M_{v[j]}, M_{-v[j]}))$
    $- \log(P(z_{t,i} \mid \bar{M}_{v[j]}, M_{-v[j]}))$
  end for
end procedure

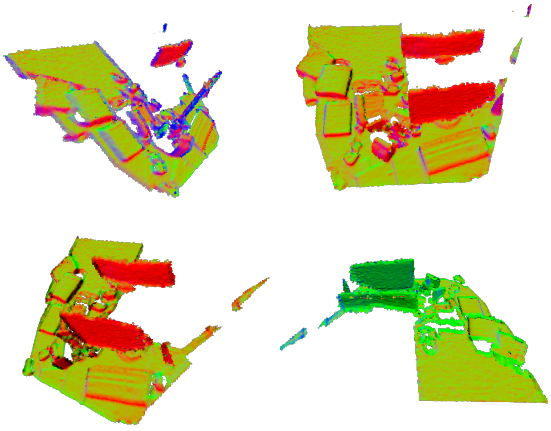

Figure 5. Normal map with pseudo-color at different viewpoints of the reconstructed scene in TUM fr1-xyz.

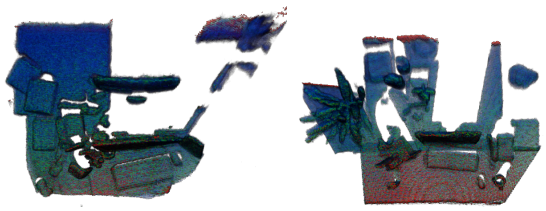

Figure 6. Probability visualization for map reconstructed from TUM fr-xyz1 and fr-xyz2. Red indicates a higher probability, green a medium probability, blue a lower probability.

ible to our updating strategy. Given a certain ray $\boldsymbol{r}$, we calculate directional derivative to approximate Equation (16) by two means,

$$\boldsymbol{r}^T \nabla l_{v,T} = 0, \tag{17}$$

$$\text{or } \frac{\partial l_{v,T}}{\partial \boldsymbol{r}} = 0. \tag{18}$$

These two approaches are equivalent in theory, but not in numerical computations. We have tested both methods:

- Directly compute numerical gradient $\nabla l_{v,T}$ on the whole map using a 3D Sobel operator. For each ray, we compute directional derivative by dot product the direction vector and the 3 dimensional gradient vector according to Equation (17).
- Calculate directional derivative by differentiate $l_{v,T}$ along the ray according to Equation(18) during the ray-casting.

According to experiments, the former approach is more stable and leaves out less surface points. After finding the zero-crossing points of Equation (17), we estimate the surface normals by calculating the gradient of the directional derivative value around; interpolating the gradient with its neighbors is more natural, but is numerically unstable – the norm of gradient around the surface is usually very small, sometimes with a contrary direction, which will lead to counteracting.

It is worth of mention that we can interpret SDF as a derivation of our approach. Consider squared distance to be the $\log$ of Gaussian: projective distance would be its directional derivative:

$$\boldsymbol{r}^T \nabla \log \mathcal{N}(x; z, \sigma) = \frac{\boldsymbol{r}^T (z - x)}{\sigma^2}, \tag{19}$$

the direction of $\boldsymbol{r}$ affects the sign of the result. Involving the updating schema in [15], SDF is the approximation of the directional derivative of a probability function that is the product of a series of Gaussian functions with weights on their exponents.

4. Experiments

We test our framework on the TUM [19] and the KITTI [6] datasets. For the TUM dataset, we focus on fusing data from one depth scanner at various viewpoints; for the KITTI dataset, we target on fusing data from various scanners at one viewpoint. Our results are compared against KinectFusion (KF) [15] we implemented with the tracking stages eliminated, as poses are provided. It can be regarded as a typical implementation of generally used SDF [3]. We also run Octomap [8]'s open source code on the KITTI dataset.

In both experiments, our programs run on a machine with an Intel(R) Xeon(R) E5-1650 v3@3.50GHz CPU and a GeForce GTX TITAN X GPU. All the parallelizable operations including data fusion, gradient calculation and ray-casting are implemented on GPU; computation of error model for each point is currently on CPU.

4.1. Experiments on TUM

We test our approach on fr1-xyz and fr2-xyz. In both experiments, our program receives 640×480 depth images along with the ground truth of the sensor's pose. There still exists error of ground truth since the timestamps of the poses and the depth images are not perfectly aligned; we choose the part of data where the motion of Kinect is not severe to reduce pose error.

We allocate a 512^3 volume of 8mm's voxel size for both our method and KF. For our approach, we adopt the axis sigma described in [16] as the error prior; to determine $\boldsymbol{c}_{t,i}$ for each $z_{t,i}$, we search $\pm 5\sigma_{t,i}$ around each $z_{t,i}$. Correspondingly, we run KF twice, setting the truncate distance in TSDF to be $\min 5\sigma_{t,i}$ and $\max 5\sigma_{t,i}$.

Figure 7 shows the results of our approach and KF. Our method provides a more comprehensive initialization for both datasets, while KF produces some holes in this stage. Admittedly, KF generates more smooth surfaces after the fusion; our method however provides more natural edges without black offsets; the result of KF is severely affected by the choice of truncating distance.

To demonstrate that our surface extraction algorithm is able to find the correct surface at any viewpoints, we store the volume and display the surface with a moving virtual camera; the normal images of extracted surfaces are shown in Figure 1 and 5.

We also draw the probability map of the reconstruction results in Figure 6 with a simple implementation of the volume rendering technique [4] that render traced voxels according to their probability value (after the conversion from log odds to probability). As σ is relatively small for data points, the probability range is not very large, thus displayed color is not very distinctive; nevertheless, it is not hard to recognize that the occluded regions with more complicated structures are more uncertain. At the border of the maps, there are some singular points with high probability due to a

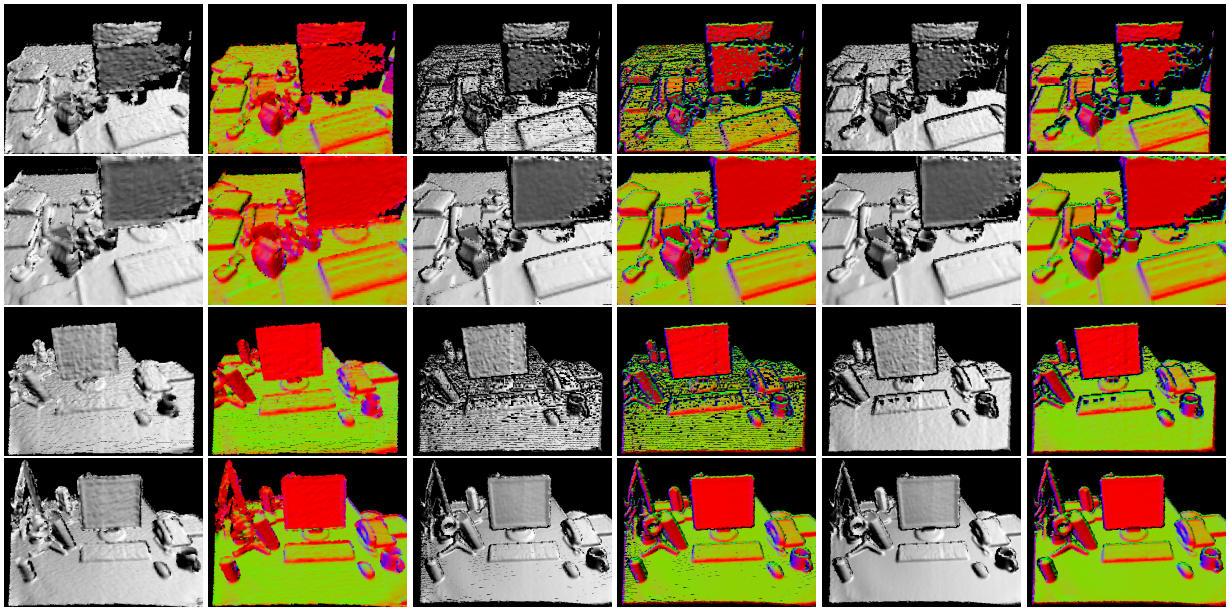

Figure 7. Result comparison between our method and KF. From left to right: Phone shading surface images with pseudo-color surface normal images of our approach; of KF with truncating distance $= 3\times$ voxel-size; of KF with truncating distance $= 5\times$ voxel-size. From top to bottom: 1st frame for TUM fr1-xyz; 20th frame for TUM fr1-xyz; 1st frame for TUM fr2-xyz; 60th frame for TUM fr2-xyz.

Method	Fusing	Gradient	Ray-casting	Total
Ours	0.0031	0.0144	0.0156	0.0331
KF	0.0121	-	0.0134	0.0255

Table 1. Running time analysis on TUM fr1-xyz. Performance is similar on TUM fr2-xyz.

lack of supplementary measurement around: only hit factor is considered in such regions. This effect can be eliminated by adding more observations around such places.

Table 1 shows the average running time for every stage including the data transferring time from CPU to GPU. Our approach works faster on fusing stage, as we only iterate over voxels around observed 3D points, while KF iterates over all voxels in the volume. As the zero-crossing point detection is more complicated, our approach is slower in ray-casting stage. For implementation simplicity, we compute gradient over all voxels, which can be optimized; KF does not explicitly requires this stage. Although our approach is not carefully optimized, *e.g.*, ray-casting does not skip the vacant areas but traces voxels one-by-one, gradient solver iterates over voxels that are not updated, it is still able to achieve real time.

4.2. Experiments on KITTI

We test our approach on a pair of aligned 1226×370 depth images generated from a LiDAR and a stereo camera from KITTI's Stereo Evaluation 2012 where the scene contains several cars parking in the street. The depth image from the stereo camera is generated by Displets [7]. We conduct this experiment to prove the availability of our framework to fuse data from 3D sensors with very different properties (sparse but accurate against dense but uncertain). As the volumetric representation we used at current can only cover a region with limited space while the sensor motion is very fast in the dataset, we do not test on continuous frames.

We allocate a 512^3 volume with 8cm's voxel size for both our method and KF. For our approach, we choose a fixed σ for LiDAR as it is relatively tiny; the σ for data from the stereo camera are computed in prior according to [5]. As the σ diverse severely, we choose $\max\{5\sigma_{t,i}, 3 \times \text{voxel-size}\}$ as the searching range. For KF, we test truncating distance from $1 \times$ voxel-size to $10 \times$ voxel-size, and choose the best $5 \times$ voxel-size.

Figure 8 illustrates the results of our approach and KF. The adaptivity of our framework is apparent: although noise exists, the fusion generally preserves the details from the more accurate LiDAR (*i.e.*, the wheels and windows of the car), while absorbs blocks of data from the stereo camera. In contrast, KF fails to extract dense surfaces from the sparse LiDAR data. In spite of the fact that we put a lot of weight on the signed distance computed from the LiDAR data, the effect of LiDAR can hardly be recognized after the fusion. In addition, we visualize the probability map created by separate 3D sensors before and after fusion. According to the bottom line of Figure 8, two sensors compensate for

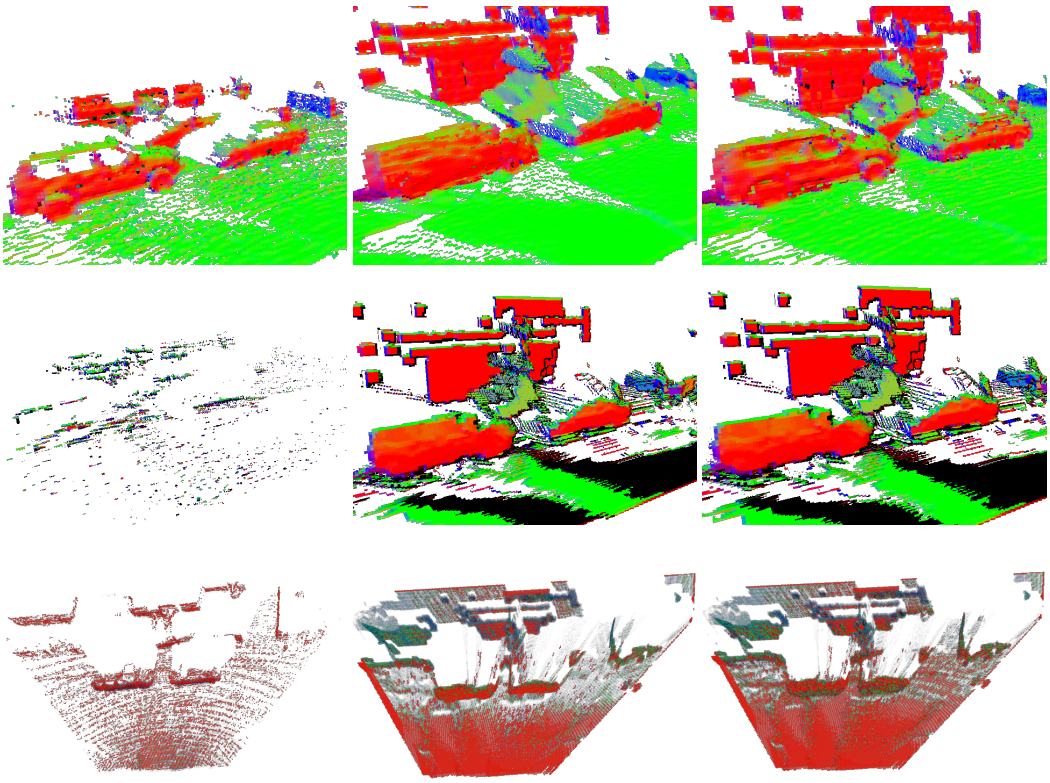

Figure 8. Results on KITTI. From left to right, result of LiDAR data only; stereo camera data only; fused data. From top to bottom: first two lines, pseudo-color surface normal map extracted from the map of our approach, of KF; last line: probability visualization of our method.

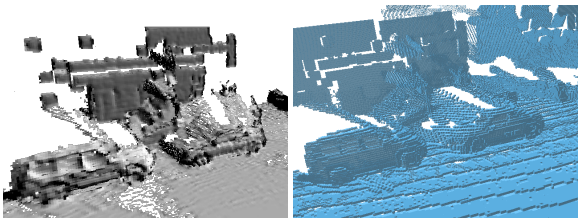

Figure 9. Result comparison between our method and Octomap. Ours provides a Phong shaded image; Octomap only provides a voxel-occupancy visualization without surface extraction or normal estimation.

each other during the fusion, proving the effectiveness of involving the sensors' error prior.

We also compare our results in Figure 9 with Octomap, setting the same voxel size for both approaches. Octomap does not provide methods for direct surface extraction or normal estimation, but only a voxel-wise occupancy visualization. Thus we merely illustrate our Phong shaded surface image against the result of theirs. It is hard to judge the quality of the results with different appearances of data; at least our method provides a consistent and direct solution to extract the surface from a probability map.

5. Conclusions and Future Work

We propose an adaptive framework to fuse 3D data from various perspectives and different sensors into a probability map. The framework models uncertainty of sensor's reading at the data-level, hence is adaptive to sensors: given the error model of a sensor that can be determined in prior, it directly fuse the data from a new 3D sensor without the need to redesign a higher level PGM. This incremental framework is simple yet effective, being able to handle data with diverse properties; surfaces can be reconstructed from the probability maps by a compatible surface extraction approach with reasonable quality.

In the future, this work might include several improvements and extensions: Firstly, we want to adjust the details in the generative model to make the system more stable in numerics; secondly, we intend to replace the plain voxel representation into an octree [24] or a hash table [17] to optimize the memory usage. Finally, the proposed probability function by nature involves the pose ξ of a sensor. By considering ξ as an unknown variable along with M, this model might provide an integrated solution for both localization and mapping.

References

[1] R. Cabezas, O. Freifeld, G. Rosman, and J. W. Fisher. Aerial Reconstructions via Probabilistic Data Fusion. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[2] B. Choo, M. Landau, M. DeVore, and P. Beling. Statistical Analysis-Based Error Models for the Microsoft KinectTM Depth Sensor. *Sensors*, 14(9):17430–17450, 2014. 3

[3] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *ACM SIGGRAPH*, 1996. 2, 6

[4] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. In *ACM SIGGRAPH*, 1988. 6

[5] J. Engel, J. Sturm, and D. Cremers. Semi-dense Visual Odometry for a Monocular Camera. In *IEEE International Conference on Computer Vision*, 2013. 7

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 6

[7] F. Guney and A. Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 7

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. 2, 3, 6

[9] X. Hu and P. Mordohai. Robust Probabilistic Occupancy Grid Estimation from Positive and Negative Distance Fields. In *IEEE International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 2012. 2

[10] Y. M. Kim, C. Theobalt, J. Diebel, and J. Kosecka. Multiview image and tof sensor fusion for dense 3d reconstruction. In *IEEE International Conference on Computer Vision*, 2009. 2

[11] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning Mobile Phones into 3D Scanners. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2, 4

[12] K. Konolige. Improved Occupancy Grids for Map Building. *Autonomous Robots*, 4(4):351–367, 1997. 2

[13] G. Marin, P. Zanuttigh, and S. Mattoccia. Reliable Fusion of ToF and Stereo Depth Driven by Confidence Measures. In *European Conference on Computer Vision*, 2016. 2

[14] R. A. Newcombe, D. Fox, and S. M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, 2011. 2, 6

[16] C. V. Nguyen, S. Izadi, and D. Lovell. Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking. In *IEEE International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 2012. 3, 6

[17] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32(6):1–11, 2013. 8

[18] K. Pathak, A. Birk, and J. Poppinga. 3d forward sensor modeling and application to occupancy grid based sensor fusion. In *IEEE International Conference on Robotics and Automation*, 2007. 2

[19] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IEEE International Conference on Intelligent Robot Systems*, 2012. 6

[20] S. Thrun. Learning Occupancy Grid Maps with Forward Sensor Models. *Autonomous Robots*, 15(2):111–127, 2003. 2, 4, 5

[21] A. O. Ulusoy, M. J. Black, and A. Geiger. Patches, planes and probabilities: A non-local prior for volumetric 3D reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[22] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Robotics: Science and Systems*, 2015. 2

[23] O. J. Woodford and G. Vogiatzis. A Generative Model for Online Depth Fusion. In *European Conference on Computer Vision*, 2012. 2

[24] M. Zeng, F. Zhao, J. Zheng, and X. Liu. Octree-based fusion for realtime 3D reconstruction. *Graphical Models*, 75(3):126–136, 2013. 8