DUGMA: Dynamic Uncertainty-Based Gaussian Mixture Alignment

Citation for published version:

Digital Object Identifier (DOI):
10.1109/3DV.2018.00092

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings International Conference on 3D Vision 2018

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
DUGMA: Dynamic Uncertainty-Based Gaussian Mixture Alignment

Can Pu
can.pu@ed.ac.uk

Nanbo Li
nanbo.li@ed.ac.uk

Radim Tylecek
rtylecek@inf.ed.ac.uk

Robert B. Fisher
rbf@inf.ed.ac.uk

School of Informatics, University of Edinburgh

Abstract

Accurately registering point clouds from a cheap low-resolution sensor is a challenging task. Existing rigid registration methods failed to use the physical 3D uncertainty distribution of each point from a real sensor in the dynamic alignment process. It is mainly because the uncertainty model for a point is static and invariant and it is hard to describe the change of these physical uncertainty models in different views. Additionally, the existing Gaussian mixture alignment architecture cannot efficiently implement these dynamic changes.

This paper proposes a simple architecture combining error estimation from sample covariances and dynamic global probability alignment using the convolution of uncertainty-based Gaussian Mixture Models (GMM). Firstly, we propose an efficient way to describe the change of each 3D uncertainty model, which represents the structure of the point cloud better. Unlike the invariant GMM (representing a fixed point cloud) in traditional Gaussian mixture alignment, we use two uncertainty-based GMMs that change and interact with each other in each iteration. In order to have a wider basin of convergence than other local algorithms, we design a more robust energy function by convolving efficiently the two GMMs over the whole 3D space.

Tens of thousands of trials have been conducted on hundreds of models from multiple datasets to demonstrate the proposed method’s superior performance compared with the current state-of-the-art methods. All the materials including our code are available from https://github.com/Canpu999/DUGMA.

1. Introduction

With recent improvements in depth sensing devices and algorithms, 3D point clouds are more accessible to researchers. However, using an expensive high-precision 3D scanner to get accurate and large-scale 3D point clouds is still not popular. Accurate alignment of noisy and partial point clouds with many outliers from cheap low-resolution sensors is still a core technique in various fields, such as computer vision, robotics, virtual and augmented reality.

Finding the accurate transformation between two noisy rigid point clouds is generally hard: true point-to-point correspondences seldom exist, which limits the accuracy of the methods based on ICP [29, 32, 23, 1, 5, 12, 21]. As for the methods based on local descriptors [16, 30, 22, 25], coarse points tend to cause inaccurate local descriptors to mismatch with each other. Also, the variable density (distant areas have a lower point cloud density) will make them unstable. Aligning probabilistic models can effectively mitigate the problems above. Thus, many researchers have been exploring different kinds of probabilistic models [24, 3, 31, 15, 19, 26] to represent the real surface structure.

However, as far as we can tell, no one has incorporated the physical 3D uncertainty distribution information for each point from a real sensor into the probabilistic model, which allows describing the real surface structure more accurately. The challenges mainly lie in two parts: the first is how to get the real uncertainty distribution information from the real sensors. In the recent years, an increasing number of researchers have been investigating how to estimate the uncertainty of the acquired data for different sensors, such as the Kinect sensor [20], the time of flight sensor [7], the structure from motion sensor [10] and the stereo vision sensor [18]. These suggest using physical noise models for each point to represent their individual occurrence probability in 3D space. The second challenge is how to use physical uncertainty information for each
In this work, DUGMA incorporates the 3D uncertainty distribution of each 3D point from a sensor into a dynamic Gaussian mixture alignment system. The figure shows a critical example of aligning two noisy and partial 3D scans with many outliers and different densities from two real low-resolution scanners using our algorithm.

Figure 1: In this work, DUGMA incorporates the 3D uncertainty distribution of each 3D point from a sensor into a dynamic Gaussian mixture alignment system. The figure shows a critical example of aligning two noisy and partial 3D scans with many outliers and different densities from two real low-resolution scanners using our algorithm.
1) Incorporation of the invariant 3D uncertainty distribution information (represented by a Gaussian function with a physical covariance) into the dynamic registration;
2) A bridge to make the two point clouds interact with each other by creating a novel point proximity weight term;
3) A more robust energy function and efficient approximation to the optimization step that greatly reduces computational complexity.

2. Related Work

In 1992, Besl and McKay [1] first introduced the Iterative Closest Point (ICP) algorithm to compute the rigid transformation between two point clouds by minimizing the Euclidean distance between the corresponding points. Since then, a large number of variants have been proposed and the reader could be directed to the survey [21]. To enhance robustness to noise, Segal et al. proposed Generalized-ICP [23] in 2009, which considered the probability distribution of each point but was still based on binary correspondence search. To be robust to occlusion and small partial overlap, researchers [32] built bilateral correspondence using bidirectional distances. To widen the convergence zone, GO-ICP [29] used a branch-and-bound method to avoid getting stuck in local minima. Exact point-to-point correspondences seldom exist and the correspondence definition (two points have the minimum distance rather than are in the same place) is coarse, which makes it hard for the ICP family to achieve accurate results.

The second class of the alignment approaches is feature-based methods, which first extract and match local descriptors (e.g. FPFH [22], SHOT [25]) from two point clouds and then estimate the relative pose using random sampling [22], RANSAC [11], Hough transforms [27], etc. Recently, Zeng et al. [30] used a siamese neural network to learn a local patch descriptor in a volumetric grid to establish the correspondences between two point clouds. Similarly, Elbaz et al. [9] used a deep neural network auto-encoder to encode local 3D geometric structures instead of traditional descriptors. Lei et al. [16] proposed a fast descriptor based on eigenvalues and normals computed from multiple scales to extract the local structure of the point clouds and then recovered the transformation from matches. However, they are sensitive to noisy point clouds. Additionally, the density of the point cloud influences the extraction of local descriptors and even makes them completely break down if the density is too low.

Aligning probabilistic models which represent the structure of the point cloud can efficiently mitigate the problems above. Our method belongs to this class. One key factor to an accurate and robust registration is the data representation used. Since 2003, many approaches based on a variety of probabilistic models have been explored to represent the structure of the point cloud such as Robust Point Matching [6], Kernel Correlation [26], Coherent Point Drift [19]. In 2011, GMMREG [15] used two Gaussian mixture models with the same isotropic covariances for each point and minimized the $L_2$ distance between the two GMMs to get the transformation. The GMM which represented the fixed point cloud was regarded as invariant and thus could not receive the current registration state from the other point cloud. In 2014, Zhou et al. [31] proposed to use the Student’s-t mixture model to represent the point cloud in the registration algorithm. In 2015, Campbell et al. [3] used a Support Vector Machine to learn and construct SVGM (a GMM with non-uniform weights) to represent the point cloud. In the next year, Campbell et al. used SVGM [3] and the architecture in GMMREG [15] to get the globally-optimal transformation using a branch and bound approach in order not to be vulnerable to local minimum. Recently, Straub et al. [24] used a Dirichlet process Gaussian mixture (DP-GMM) and a Dirichlet process von Mises-Fisher mixture (DP-vMF-MM) to represent the geometric information of the point cloud. The mathematical probabilistic model used to represent the point cloud has become considerably complex. Nevertheless, we will use the physical 3D uncertainty distribution from real sensors to construct a simple uncertainty-based GMM to represent the structure of the point cloud, which will fit the real surface geometry better.

The acquisition of a physical 3D uncertainty distribution for each point from a real sensor is a difficult task. With the development of different depth sensors, more effective uncertainty estimation methods for various sensors have been designed. In 2012, Nguyen et al. [26] used the distance and angle between the Kinect sensor and observed surface to estimate both axial and lateral noise distributions. In 2013, Engel et al. [10] used the geometric disparity error and photometric disparity error for the structure from motion sensor to estimate 3D point error. Recently, many researchers [7, 18] have estimated the uncertainty for the ToF (Time of Flight) sensor based on the physical properties of the sensor (e.g. the IR frequency). Meanwhile, [18] developed an empirical uncertainty model for the stereo vision sensor based on the relationship between the local cost and global cost.

In summary, there are previously developed methods for both robust rigid point cloud registration and modelling the 3D uncertainty distribution of the points in a 3D point cloud. This paper improves registration accuracy and robustness using an approach that combines these two themes and redesigns a dynamic Gaussian mixture alignment system using invariant 3D uncertainty information from each point cloud.
3. Methodology

First we introduce the change of 3D uncertainty distribution, then build a bidirectional dynamic bridge between the two point clouds, and finally introduce the framework of our math model. Table 1 lists some of the symbols and their notations.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X, Y)</td>
<td>Two point clouds</td>
</tr>
<tr>
<td>(D)</td>
<td>Dimensionality of the point clouds</td>
</tr>
<tr>
<td>(N, M)</td>
<td>Number of points in (X, Y) point cloud</td>
</tr>
<tr>
<td>(x_n)</td>
<td>One point in (X) point cloud</td>
</tr>
<tr>
<td>(y_m)</td>
<td>One point in (Y) point cloud</td>
</tr>
<tr>
<td>(\Sigma_{x_n}, \Sigma_{y_m})</td>
<td>Covariance for point (x_n) and (y_m)</td>
</tr>
<tr>
<td>(I)</td>
<td>Identity matrix</td>
</tr>
<tr>
<td>(0)</td>
<td>Column vector of all zeros</td>
</tr>
</tbody>
</table>

3.1. Change of 3D Uncertainty Distribution

We will use a Gaussian function with a covariance to represent the distribution of one point in 3D space. The specific covariance for each point represents the physical 3D uncertainty distribution for that point from a real sensor.

1. If a point with covariance \(\Sigma_{\text{orig}}\) has been rotated by \(R\), then \(\Sigma = R \Sigma_{\text{orig}} R^T\).

2. A scaling factor for the covariance of a point is proportional to the average minimum distance \(\sigma\) between two point clouds to ensure that the probability of all the points in the other point cloud will not become too small when the two point clouds are far away from each other. See Algorithm 1.

3.2. Dynamic Gaussian Mixture Alignment

The Gaussian function \(g_{x_n}(\tau)\) of the point \(x_n\) predicts the probability that \(x_n\) appears at the position \(\tau\) in its own coordinate system. Based on Gaussian weights around each point, we will define a probability-like function that not only depends on the distribution of the point (represented by isotropic or anisotropic covariance) but also whether a possible corresponding point \(cp_{x_n}\) in the other point cloud is nearby. We model the presence of a corresponding point by a weight function \(w_{x_n}(\tau, cp_{x_n})\) that has significant value only when a potential corresponding point \(cp_{x_n}\) from point cloud \(Y\) is near the position \(\tau\). A similar definition holds for \(g_{y_m}(\tau), w_{y_m}(\tau, cp_{y_m})\). Thus either point cloud can receive and send current state information from or to the other point cloud bi-directionally to evaluate the current registration quality.

In the analysis below, we assume the \(Y\) point cloud has been already transformed from the initial point cloud \(Y_0\) by rotation \(R\) and translation \(t\) (which then become the domain for the optimization of the evaluation function). The product \(g_{x_n}(\tau)g_{y_m}(\tau)\) represents \(x_n, y_m\) appearing at the same position \(\tau\) in the same coordinate system. Thus, it encodes the underlying prior knowledge, i.e. \(x_n, y_m\) are possible corresponding points from two point clouds. In other words, any two points from the fixed and moving point cloud can be a corresponding pair in our system and the likelihood depends on the probability of correspondence that \(x_n, y_n\) appear at the same position \(\tau\), which is different from soft assignment [13] in essence.

3.3. The description of our model

Based on the previous discussion, we design the uncertainty-based GMM as follows:

\[
G^{I, 0}_{X}(\tau) = \sum_{n=1}^{N} w_{x_n}(\tau, cp_{x_n}) g_{x_n}(\tau),
\]

\[
G^{R, t}_{Y}(\tau) = \sum_{m=1}^{M} w_{y_m}(\tau, cp_{y_m}) g_{y_m}(\tau),
\]}
where the gaussian kernels are given by

\[ g_{xn}(\tau) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_{xn}|}} e^{-\frac{1}{2}(\tau-x_n)^T\Sigma_{xn}^{-1}(\tau-x_n)} \] (3)

\[ g_{ym}(\tau) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_{ym}|}} e^{-\frac{1}{2}(\tau-y_m)^T\Sigma_{ym}^{-1}(\tau-y_m)} \] (4)

\[ w_{xn}(\tau, cp_{xn}) = e^{-\frac{1}{2}(cp_{xn}-\tau)^T\Sigma_{xn}^{-1}(cp_{xn}-\tau)} \] (5)

\[ w_{ym}(\tau, cp_{ym}) = e^{-\frac{1}{2}(cp_{ym}-\tau)^T\Sigma_{ym}^{-1}(cp_{ym}-\tau)} \] (6)

\[ G^{10}_X(\tau) \] denotes the GMM from the fixed point cloud \( X \) and \( G^{R,t}_Y(\tau) \) represents the GMM from the moving point cloud \( Y \) after rotation \( R \) and translation \( t \). Thus \( y_m = R y_m + t \), \( \Sigma_{ym} = R \Sigma_{ym0} R^T \), \( |\Sigma_{ym}| = |R \Sigma_{ym0} R^T| = |\Sigma_{ym0}| \) due to \( |R| = 1 \). \( \Sigma_{ym0}^{-1} = (R \Sigma_{ym0} R^T)^{-1} = R \Sigma_{ym0}^{-1} R^T \) due to \( RR^T = I \). At all times, \( x_n = x_{n0} \) and \( \Sigma_x = \Sigma_{x0} \). Each point has its own covariance.

Integrating the product of the two dynamic GMMs (representing the overlapping effect of the two point clouds) over the whole 3D space, as we shall show in our experiments, makes the energy function more robust, accurate and have a wider convergence basin compared with [19, 15, 4].

We now formulate the optimization over rotation \( R \) and translation \( t \) as an EM-like process. First, an energy function is defined as the following

\[ E = \int P(\tau) \log \left[ G^{10}_X(\tau)G^{R,t}_Y(\tau) \right] d\tau, \] (7)

where \( \tau \) integrates over all the domain of the point clouds; \( P(\tau) \) is the probability that there is a point at the position \( \tau \). We design it as the sum of the probability that all the possible corresponding pairs appear at the position \( \tau \) in

\[ P(\tau) = \sum_{i=1}^{N} \sum_{j=1}^{M} P(\tau, x_i, y_j) = \sum_{i=1}^{N} \sum_{j=1}^{M} g_{x_i}(\tau)g_{y_j}(\tau). \] (8)

Equation (7) can be rewritten as

\[ E = \int P(\tau) \log \left[ \sum_{i=1}^{N} \sum_{j=1}^{M} P^{R,t}(\tau, x_i, y_j) \right] d\tau, \] (9)

with a combined term

\[ P^{R,t}(\tau, x_i, y_j) = w_{x_i}(\tau, cp_{x_i})g_{x_i}(\tau)w_{y_j}(\tau, cp_{y_j})g_{y_j}(\tau). \] (10)

By the definitions above, the weight term \( w_{x_i}(\tau, cp_{x_i}) \) is nearly zero when point \( x_i \) is far from any point in \( \{y_j\} \). This allows us to avoid having to compute correspondences by using all \( y_j \) in place of \( cp_{x_i} \) (and similarly for \( cp_{y_j} \)) and simplify \( P^{R,t} \) with

\[ P^{R,t}(\tau, x_i, y_j) = w_{x_i}(\tau, y_j)g_{x_i}(\tau)w_{y_j}(\tau, x_i)g_{y_j}(\tau). \] (11)

We maximize Equation (9) to get the estimated rotation and translation by minimizing its negative. We adopt the EM algorithm [8, 2] to solve for \( R, t \). Its main idea is: guess the values of the parameters firstly in the last iteration (denoted by ‘old’) and calculate the posteriori probability \( P^{old}(x_i, y_j \mid \tau) \) using Bayes’ theorem then, which corresponds to the expectation stage. After that, minimize the expectation of the negative log-likelihood function \( \mathcal{L} \) to find the parameters in the current iteration (denoted by ‘new’), which corresponds to the maximization stage. Thus, we get

\[ \mathcal{L} = - \sum_{i=1}^{N} \sum_{j=1}^{M} B(\tau, x_i, y_j) \log(\tilde{P}^{new}(\tau, x_i, y_j)) d\tau, \] (12)

\[ B(\tau, x_i, y_j) = P(\tau) P^{old}(x_i, y_j \mid \tau). \] (13)

Neglecting the constant term and using \( P(\tau) \approx P^{old}(\tau) \), we simplify the target function to

\[ \mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{M} P^{old}(\tau, x_i, y_j) M a h^{new}(\tau, x_i, y_j) d\tau, \] (14)
where a term similar to Mahalanobis distance is obtained:
\[
Mah_{new}(\tau, x_i, y_j) = \frac{1}{2}(\tau - x_i)^T(\Sigma_{x_i}^{-1} + \Sigma_{y_j}^{-1})(\tau - x_i) \\
+ \frac{1}{2}(\tau - y_j)^T(\Sigma_{x_i}^{-1} + \Sigma_{y_j}^{-1})(\tau - y_j).
\] (15)

As we will justify below, there is no real benefit to integrate the whole 3D space, because the values of the Gaussian functions are only significant near the data points themselves. In fact, because most values are quite low, we approximate the integral by a sum at only the data points to speed up the algorithm drastically (unlike [4]). Thus we need evaluate only each term \(P^{\text{old}}(\tau, x_i, y_j)Mah_{new}(\tau, x_i, y_j)\) at the positions of \(x_i\) and \(y_j\), which will reduce the time complexity greatly to only \(O(MN)\). Applying this simplification, the approximated energy function becomes
\[
\tilde{\mathcal{L}} = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{\tau \in \{x_i, y_j\}} P^{\text{old}}(\tau, x_i, y_j)Mah_{new}(\tau, x_i, y_j).
\] (16)

Expanding the last sum and uniting like terms we get
\[
\tilde{\mathcal{L}} = \sum_{i=1}^{N} \sum_{j=1}^{M} C^{\text{old}}_{i,j} \cdot (y_j - x_i)^T(\Sigma_{x_i}^{-1} + \Sigma_{y_j}^{-1})(y_j - x_i),
\] (17)

where
\[
C^{\text{old}}_{i,j} = (2\pi)^{-D}|\Sigma_{x_i}|^{-\frac{1}{2}}|\Sigma_{y_j}|^{-\frac{1}{2}}
\left(e^{-\frac{1}{2}(y_j - x_i)^T \Sigma_{x_i}^{-1}(y_j - x_i)} + e^{-\frac{1}{2}(x_i - y_j)^T \Sigma_{y_j}^{-1}(x_i - y_j)}\right).
\] (18)

The \(x_i, y_j, \Sigma_{x_i}\), and \(\Sigma_{y_j}\) in \(C^{\text{old}}_{i,j}\) are from the previous iteration. We then minimize Equation (17) over the rotation \(R\) and translation \(t\) domain, using interior point optimization [28] as summarized in Algorithm 1.

---

**Algorithm 1 DUGMA Point Cloud Registration.**

**Input:** Two point clouds \(X, Y\) and their covariances \(\Sigma_{x}, \Sigma_{y}\), initial transformation \(R = I, t = 0\).

---

1. **Algorithm EM-like optimization, repeat until convergence:**
2. **procedure** E-STEP
3. \(Y \leftarrow RY + t\) \(\triangleright Upate \ Y, \sigma, \Sigma_{y}, C^{\text{old}}_{i,j}\)
4. \(\sigma \leftarrow \frac{1}{M} \sum_{j=1}^{M} d_{\text{min}}(y_j, X)\) \(\triangleright Minimum \ distance\)
5. \(\Sigma_y \leftarrow \sigma R \Sigma_y R^T\)
6. \(C^{\text{old}}_{i,j} \leftarrow \text{compute Eq. (18)}\)
7. **procedure** M-STEP \(\triangleright Solve \ for \ R, t\)
8. Use interior point algorithm to solve Eq. (17):
9. \((R, t) \leftarrow \arg \min_{R,t} \tilde{\mathcal{L}}\)

---

4. **Experiments**

We implemented our algorithm using Matlab and Cuda C++. We ran all the algorithms on a laptop with Intel Core i7-7820HK processor (quad-core, 8MB cache, up to 4.4GHZ) and NVidia Geforce GTX 1080 with 8GB GDDR5X. To test the accuracy and robustness of our algorithm, our proposed method is compared with relevant recent algorithms from the top journals and conferences: CPD [19], GMMREG [15], BDICP [32], GOICP [29], GOGMA [4], 3DMATCH [30]⁴, FDCP [16]². All the code is directly from the authors. We did not compare ours with [9] because we could not get our re-implemented algorithm to work well based on their partial released code. The Stanford 3D Scanning Repository [17] and our new 3D dataset have been used to do performance comparison of the algorithms. After that, 30 real scenes with ground truth from multiple Kinect sensors in our new dataset³ have been used to show the approach works on par or better in a real application compared with the rest.

---

⁴We only compared ours with 3DMATCH in the Kinect data application with their pre-trained weights.

²FDCP has compared their results with [24] so we neglected [24].

³[https://github.com/Canpu999/DUGMA](https://github.com/Canpu999/DUGMA)
4.1. Simulation

To synthesize the two point clouds to register, we randomly choose a model from the datasets above for two point clouds firstly. Then a different random large segment of each point cloud is removed completely to simulate occlusion. After that, the two occluded models are sampled differently, which simulates the resolution of different sensors in real scenarios. Also, different anisotropic Gaussian noise with random standard deviations and zero mean has been added to each point to simulate the complex noise in real environments resulting from known and unknown factors. The variances of all the noise on each axis have been stored in the covariances accurately. Next, outliers have been added into both point clouds to simulate outliers acquired by the sensors. Finally, an initial rigid transformation is applied to the moving point cloud.

The experiments are divided into four groups given the four influence factors or variables: noise, outliers, occlusion, and initial rotation. In each group of experiments, one controlled variable will be changed and the values of the other variables will be picked randomly from a default range. The experiment is conducted 3 times at each controlled value for each of 100 shapes with a random perturbation each time, see Algorithm 2. The maximum iteration value for all is 100. For FDCP, we set gridStep=1.5 and Rho=0.1 to make it robust to different densities. For GOGMA we set the scale parameter for SVM (0.5,0.5)) to limit GOGMA’s running time to around 100 seconds per registration. For GOICP, Mean Squared Error (MSE) convergence threshold MSEThresh=0.2. The rest of the parameters share default values in their open code.

We use $||t_{gt} - t_{est}||_F$, $||I - R_{gt}R_{est}^{-1}||_F$ [14] to estimate the quality of the registration, where $R_{gt}$, $t_{gt}$ are the ground truth and $R_{est}$, $t_{est}$ are estimated results respectively and $|| \bullet ||_F$ is the Frobenius norm.

Algorithm 2 Controlled and random variables process. For each method, 14700 trials have been done.

1: for controlled_variable := start by step to end do
2:   for shape := 1 to 100 do
3:     for instance := 1 to 3 do
4:       Produce data with controlled and random variable
5:     Do registration (different algorithms)
6:   Calculate the registration error
From the Stanford 3D Scanning Repository (50) and our new dataset (50) we got 100 models from various views of different objects and scenes. Each was downsampled to about 1000 points with different densities. Figure 2 shows 6 example models from different scenes and objects.

We apply different effects to simulate the real factors in the real environment. Figure 3 shows examples of the effects. In our experiment, the sampling rate is set to 90% and 85% for the fixed and moving point cloud, respectively. Table 2 gives specific information about the parameters.

Table 2: Range for random and controlled factors

<table>
<thead>
<tr>
<th>Factor</th>
<th>random range</th>
<th>controlled range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial rotation</td>
<td>[-20°, 20°]</td>
<td>[-60°, 60°]; step=8°</td>
</tr>
<tr>
<td>Outliers</td>
<td>[0, 500] that is, [0, ≈33%]</td>
<td>[0, 2000] that is, [0, ≈67%]; step=200</td>
</tr>
<tr>
<td>Noise standard deviation</td>
<td>[0, 0.2] × radius of point cloud</td>
<td>[0, 0.3] × radius of point cloud; step=0.03</td>
</tr>
<tr>
<td>Occlusion</td>
<td>[0, 15%]</td>
<td>[0, 30%]; step=0.03</td>
</tr>
</tbody>
</table>

Figure 4 shows one successful registration in a real garden. After registration, we could see the hedges and trees overlap well although there is a big patch of occlusion in both two point clouds, many outliers and noise.

Figure 4: A successful registration in a real garden.

When the initial rotation angle value is the controlled variable, it ranges from [-60°, 60°], with an 8° step. In the experiments, the specific rotation angle around each axis is chosen as 0 or the initial rotation angle value randomly. Figure 5 shows that beyond -40° or 40°, the proposed algorithm breaks down because the iteration count exceeds the maximum. But within [-40°, 40°], our algorithm is much more stable and accurate compared with the rest. In Figures 5, 6, 7, 8, ‘Time’ refers to the average running time per registration. If rotation and translation error is below 0.2 and 0.1 respectively in a trial, the trial is a success (third plot). When occlusion rate is the controlled variable, Figure 6 shows within 25%, the proposed algorithm performs well.

Judging that GOGMA needs much more time (about 1000 sec for a trial) to achieve a much better performance and behaved poorly in the experiments above, we will neglect GOGMA in the remaining noise and outlier experiments but later compare the proposed algorithm with it in the small dataset registration experiment.

When outliers are the controlled variable, we use covariances generated in the same manner as the true data points. Figure 7 shows the proposed algorithm has superior performance again. When the noise level is the controlled variable, Figure 8 shows robust and accurate performance compared with the rest.

4.2. Real data from multiple Kinect sensors

The simulation experiments above show our algorithm works well with very accurate covariance estimates. In the real case, it is hard to get very accurate covariance estimates. In this real application, we estimate an inaccurate covariance for each 3D point from a Kinect sensor to test our algorithm. We design the uncertainty of each valid 3D point acquired by the
Kinect sensor based on the depth value $d$ and the angle $\alpha$ between the camera and the normal of the surface [20].

$$U(\alpha, d) = \exp[w_1(1 - \cos \alpha) + w_2d]$$

(19)

We use $w_1 = 1.6658$ and $w_2 = 0.2776$ by letting $U(\pi/3, 0) = U(0, 3) = 2.2$. The number 2.2 is set manually and the algorithm works well if that number is in the range [1,10] (known by our experiments). Then we simply multiplied the uncertainty and the identity matrix to estimate a coarse covariance for each point. Future work will explore more accurate real covariances to represent the 3D uncertainty distribution.

We tested our algorithm using two point clouds from two Kinect sensors. The ground truth of the rotation and translation between the two Kinect sensors is known by calibration. Figure 1 (a), (b) show the scene before and after registration using our algorithm. Figure 1 (c) adds the colour texture information into the two registered point clouds.

In the experiment, the two point clouds have been downsampled to $\sim$4K points or so using the grid average method. The downsampling was small from $\sim$20K to $\sim$4K (rather than 640*480 to $\sim$4K): the Kinect scan was cropped to the fixed scenes. Then we applied the same initial rotation to all the algorithms and reduced the scale parameter for SVM (0.08, 0.08) in GOGMA and $\text{MSE}_{\text{thresh}}=0.001$ in GOICP to make them get their best performance. Here we present results from 30 scenes in our new dataset. We calculated the error and running time based on only successful registrations (rotation and...
translation error is below 0.2 and 0.1). After all the algorithms have converged, our successful rate (most important) ranks first (96%). The estimated mean rotation of our algorithm (0.04) is lowest, see Table 3. Otherwise, our algorithm is about 7 times faster than GOGMA whose success rate (93%) ranks second.

4.3. Additional Experiments

We have also done 14700 similar trials using 100 2D fish models from the Gatorbait100\(^4\) database and have used the other 2D point clouds with 100 different shapes (face, umbrella, computer etc.) to test our sensitivity to shapes. We have used 100*5 2D models from Gatorbait100 database with 5 different densities to test our sensitivity to density. All the results are equally robust. We have also tested our method without any uncertainty information and replaced each covariance matrix with an identity matrix. The results show our method with uncertainty information is better than that without uncertainty and both are better than the other comparison algorithms. For more details, see our supplementary materials \(\text{https://github.com/Canpu999/DUGMA}\).

\(^4\)http://www.rvg.ua.es/graphs/dataset01.html
Table 3: Experiment Results for the Real Application

<table>
<thead>
<tr>
<th>Method</th>
<th>Error Mean (R)</th>
<th>Error Std (R)</th>
<th>Error Mean (t)</th>
<th>Error Std (t)</th>
<th>Suc. rate</th>
<th>Time s/trial</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPD</td>
<td>0.05</td>
<td>0.03</td>
<td>0.03</td>
<td>0.01</td>
<td>50%</td>
<td>42.0</td>
</tr>
<tr>
<td>Gmmreg</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>BDICP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>GOICP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>GOGMA</td>
<td>0.04</td>
<td>0.03</td>
<td>0.03</td>
<td>0.02</td>
<td>93%</td>
<td>1125</td>
</tr>
<tr>
<td>3dmatch</td>
<td>0.08</td>
<td>0.04</td>
<td>0.04</td>
<td>0.02</td>
<td>23%</td>
<td>6.6</td>
</tr>
<tr>
<td>FDCP</td>
<td>0.06</td>
<td>0.02</td>
<td>0.04</td>
<td>0.01</td>
<td>40%</td>
<td>8.2</td>
</tr>
<tr>
<td>Ours</td>
<td>0.04</td>
<td>0.03</td>
<td>0.04</td>
<td>0.02</td>
<td>96%</td>
<td>163</td>
</tr>
</tbody>
</table>

5. Conclusion

The proposed algorithm is simpler and more effective than the previous algorithms. We incorporated the 3D uncertainty distribution into a simple dynamic Gaussian mixture alignment. The obvious difference between our algorithm and the previous ones is that it needs covariances at each point as input, which requires error models of how to estimate the real covariance for each kind of sensor. All the experiments we have done show that the proposed method is very robust and accurate and works well. In the future, we will only use the points from set \( Y \) in the neighbourhood of \( x_i \) to approximate all the points in set \( Y \) in Equation (17) to reduce the time complexity to \( O(N) \).

6. ACKNOWLEDGMENTS

The research is funded by TrimBot2020 project [Grant Agreement No. 688007] in the European Union Horizon 2020 programme. We thank Dylan Campbell ([4, 3]), Andy Zeng [30], Huan Lei [16], Jihua Zhu [32] for providing help when we did the comparison experiments. We thank Dylan Campbell, Marcelo Saval-Calvo for giving us good advice on this paper.

References


