2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

REM-Seg: A Robust EM Algorithm for Parallel Segmentation and Registration of Point Clouds

Benjamin Eckart[1] and Alonzo Kelly[2]

Abstract— **For purposes of real-time 3D sensing, it is important to be able to quickly register together incoming point cloud data. In this paper, we devise a method to quickly and robustly decompose large point clouds into a relatively small number of meaningful surface patches from which we register new data points. The surface patch representation sidesteps the costly problem of matching points to points since incoming data only need to be compared with the patches. The chosen parametrization of the patches (as Gaussians) leads to a smooth data likelihood function with a well-defined gradient. This representation thus forms the basis for a robust and efficient registration algorithm using a parallelized gradient descent implemented on a GPU using CUDA. We use a modified Gaussian Mixture Model (GMM) formulation solved by Expectation Maximization (EM) to segment the point cloud and an annealing gradient descent method to find the 6-DOF rigid transformation between the incoming point cloud and the segmented set of surface patches. We test our algorithm, Robust EM Segmentation (REM-Seg), against other GPU-accelerated registration algorithms on simulated and real data and show that our method scales well to large numbers of points, has a wide range of convergence, and is suitably accurate for 3D registration.**

I. INTRODUCTION

Establishing and maintaining a consistent world model is one of the cornerstones of mobile autonomy, functioning as an intermediary between raw sensor measurements of the world and high level scene understanding. Spatial maps aid in the tasks of localization, tracking, and object recognition, which in turn are necessary for efficient path planning and higher level perception and intelligence. Today, 3D range sensing is ubiquitous and many modern sensors can generate massive amounts of data, making it challenging to apply standard optimization algorithms over the raw data for purposes of real-time perception. Parametric or geometric representations of surfaces allow a more compact representation than raw point clouds, inducing a more tractable structure to allow real-time computation. These compact representations also allow for more efficient and robust techniques to be applied to the problem of multiple scan registration, a necessary component of any mapping system.

Two broad categories of point cloud registration algorithms exist: those that do direct point-to-point or point-to-local-surface matching, and those that first parametrically transform one or both point clouds into a new object with better computational properties before calculating the rigid transformation. The former category includes the widely used Iterative Closest Point (ICP) algorithm, and the latter includes registration algorithms based on planar segmentation, Gaussian Mixture Models (GMM), or Normal Distance Transforms (NDT).

Historically, registration has been done using the Iterative Closest Point (ICP) algorithm, which is a somewhat *ad hoc* algorithm that iteratively refines the associations of points from a reference scan and points from a new scan [1]. ICP is somewhat ill-posed given the fact that each point cloud consists of non-uniform point samples and thus a given point is very unlikely to have an exact corresponding point in the other point cloud. Furthermore, if outliers are not properly caught and discarded, then trying to minimize outliers using the squared distance will not be robust and can cause the algorithm to diverge. In practice, ICP needs a very good initial estimate for convergence to be possible.

Many variants of ICP have been proposed and used with varying degrees of success [2]. Typically, the modifications center around avoiding the computational expense of having to find nearest neighbors for every point. Some effective advancements toward this goal include the use of Octrees for approximate nearest neighbor calculations or fast projections [3]. However, many other approaches have tried to eschew point-to-point matching in favor of registration methods based on an intermediate parametrized surface representation.

For the second category of algorithms that use surface models, many decompose the environment into planes [4][5][6]. However, assuming the world is planar is possibly too limiting of a restriction, especially in environments with clutter or curved surfaces. Other work has used mixtures of Gaussians [7][8], but many of these more expressive formulations come at a higher computational expense and thus cannot be used for real-time perception algorithms.

The methods described in this paper fall into the latter category. Our goal is to design an algorithm to decompose a point cloud into a relatively few parameters in a data parallel fashion, thereby allowing an efficient implementation on a GPU. We can take advantage of data parallel adaptations of EM for GMM's in the Machine Learning community [9] [10], but modified to be more efficient for the problem of point cloud segmentation, which contains many more points than dimensions.

We give an overview of our proposed method in Section II. The mathematics of the segmentation and registration are given in Sections III and IV, respectively. We detail additional extensions in Section V. Our implementation and algorithm are discussed in Section VI, related work is

[1] [2]B. Eckart and A. Kelly are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA USA {eckart, alonzo} at cmu.edu

mentioned in Section VII, and experiments demonstrating the segmentation, scaling with size, and convergence are shown in Section VIII. Finally, we briefly mention future applications and our conclusions in Section IX.

II. Goals and Proposed Method

Our goals can be divided into two separate, but related, optimization problems: The first problem is to transform a point cloud into a set of parametric surface patches so that the spatial representation of surfaces can be continuous and parametrically sparse. The second problem involves taking such a representation and finding the most likely rigid spatial transformation given a new set of range data representing the same scene. When the rigid spatial transformation is found, the scene and new data can then be fused together. If done in real-time, this process can be iterated as a continuous registration loop in a mapping system, for example, on a moving robot.

Past work has mainly focused on segmentation into planar regions. Many objects, however, do not fit well into this model, especially for scenes with large amounts of clutter. We would like to be able to use a more exspressive function class for our world model. One approach is to model the environment as a set of three-dimensional Gaussians. This model is more general than the planar construction since a Gaussian that is degenerate in one dimension can also approximate a planar surface. The idea that a point cloud can be decomposed into a set of 3D Gaussians sets up the problem as a Gaussian Mixture Model (GMM). In the general Gaussian Mixture Model formulation, we can reinterpret the point cloud data (PCD) as a mixture of points generated from a small number of distinct functions. Given that the functions have a small number of parameters, we can compress the possibly millions of raw 3D points into a collection of easily storable functions that adequately describe the spatial characteristics of the cloud. In other words, if we sample enough from the mixture of functions, we should effectively recreate the PCD.

In our case, we are trying to maximize the overall likelihood of a set of Gaussians having produced the given point cloud. Each Gaussian has nine free parameters representing its mean (three) and covariance (six, due to symmetry). Since both the parameters of the Gaussians and the correspondence variables to the Gaussians per point in the cloud are unknown, we solve this problem using the well-known Expectation Maximization (EM) algorithm [11].

Our given GMM formulation serves the basis for our scene. Given a new set of N 3D points, we can look at the probability of our scene having generated those points according to our surface model. In our problem, we wish to optimize over some spatial transformation in 3D space. Thus, we can minimize over the negative log-likelihood of the data with respect to the transformation parameters.

Because the particular problem of point cloud data involves a vast amount of points, we can parallelize the computation of the gradient by computing as much of the gradient as possible independently with CUDA threads and then doing a logarithmic number of reductions to obtain the final gradient step. Minimizing over the negative log-likelihood will provide the MLE spatial transformation of the data to the scene, and will allow us to register together successive 3D measurements and recover the motion of the sensor.

III. Segmentation

In our world model, we assume that the PCD has been generated by a discrete set of 3D Gaussians. Thus, given N points, $\mathbf{Z} = \{\mathbf{z}_i = (x_i, y_i, z_i)^T\}$ and a set of J 3D Gaussians with $\mathbf{\Theta} = \{\mathbf{\Theta}_j = (\pi_j, \mu_j, \mathbf{\Sigma}_j)\}$, our function describing the generation of points is a linear combination of Gaussians,

$$p(\mathbf{z}_i|\mathbf{\Theta}) = \sum_j \pi_j \mathcal{N}(\mathbf{z}_i|\mathbf{\Theta}_j) \tag{1}$$

with $\sum_j \pi_j = 1$ representing a mixture distribution. Given that the PCD is a set of *iid* points, our total likelihood of seeing the PCD from our world model is $p(\mathbf{Z}|\mathbf{\Theta}) = \prod_i p(\mathbf{z}_i|\mathbf{\Theta})$.

To produce a likelihood function, we need to introduce correspondence variables c_{ij} for each measurement that serve as "soft" labels for the cluster to which the point belongs. In other words, each point i will have j correspondence variables with values ranging from 0 to 1, with the intuition that 1 means "definitely belongs to this cluster" and 0 means "definitely does not belong to this cluster". Under this formulation, the likelihood function is simply the product of the conditional joint conditional probabilities of all the measurements and their correspondences.

We note that the expression $p(\mathbf{z}_i|\mathbf{\Theta}) = \sum_j \pi_j \mathcal{N}(\mathbf{z}_i|\mathbf{\Theta}_j)$ can be seen as a marginalization of a joint distribution $p(\mathbf{z}_i, \mathbf{c_i})$ where $p(c_{ij} = 1|\mathbf{\Theta}_j) = \pi_j$ and thus $p(\mathbf{z}_i|\mathbf{\Theta}) = \sum_j p(c_{ij} = 1|\mathbf{\Theta}_j) p(\mathbf{z}_i|c_{ij} = 1, \mathbf{\Theta}_j)$. Here $\mathbf{c_i}$ is a binary vector representing the correspondence of a point with a given cluster j.

The total joint log likelihood of the data given the model is therefore,

$$\ln p(\mathbf{Z}, \mathbf{C}|\mathbf{\Theta}) = \sum_i \ln\{\sum_j p(c_{ij} = 1|\mathbf{\Theta}_j)\mathcal{N}(\mathbf{z}_i|\mathbf{\Theta}_j)\} \tag{2}$$

Maximizing over the total data log-likelihood under the parameters Θ and $\mathbf{C}$ cannot be done in closed form due to the sum present inside the logarithm, but if we knew the correspondence variable for each $\mathbf{z}_i$, we could easily maximize this likelihood by setting $\mathbf{\Theta}_j$ to the sample mean and sample covariance of all points for which $c_{ij} = 1$. Alternately, if we knew $\mathbf{\Theta}$, we could maximize the likelihood using Bayes' rule and calculating,

$$\begin{aligned} p(c_{ij}|\mathbf{z}_i, \mathbf{\Theta}) &= \frac{p(\mathbf{z}_i|c_{ij}, \mathbf{\Theta}_j)p(c_{ij}|\mathbf{\Theta}_j)}{p(\mathbf{z}_i|\mathbf{\Theta})} && (3) \\ &= \frac{\pi_j \mathcal{N}(\mathbf{z}_i|\mathbf{\Theta}_j)}{\sum_k \pi_k \mathcal{N}(\mathbf{z}_i|\mathbf{\Theta}_k)} && (4) \end{aligned}$$

To summarize the above equation, the probability of a certain point z_i belonging to cluster j is the relative likelihood of cluster membership given the likelihood over the point in all other clusters.

Since we do not know the correspondence labels c_{ij}, nor do we know the model parameters $\mathbf{\Theta}_j$, we can use the EM algorithm to iteratively improve both the labels and cluster parameters alternately. In other words, we iteratively hold $\mathbf{C}$ as a constant and optimize over $\mathbf{\Theta}$ and then hold $\mathbf{\Theta}$ as a constant and optimize over $\mathbf{C}$.

Let's define

$$\gamma_j(\mathbf{z}_i) \overset{\text{def}}{=} p(c_{ij}=1|\mathbf{z}_i) \tag{5}$$
$$\hat{\gamma}_j(\mathbf{z}_i) \overset{\text{def}}{=} p(\mathbf{z}_i|c_{ij}=1)p(c_{ij}=1) \tag{6}$$

The gamma function can be viewed as the conditional expectation $\mathbb{E}[c_{ij}]$ over the posterior $p(c_{ij}|\mathbf{z}_i,\mathbf{\Theta})$. Gamma hat is the unnormalized probability $p(c_{ij}|\mathbf{z}_i)$.

A. Parallelization

The point cloud segmentation problem is nicely parallelizable due to the relative lack of non-interactions among points, low-dimensional parameter space, and relatively many data points compared to model parameters. Additionally, whereas many segmentation algorithms must rely on nearest neighbor calculations at some point (region growing) or operations over local neighborhoods, each point in the point cloud for EM segmentation needs to compare itself to one of a few cluster parameters. Thus, we do not need to impose any special spatial data structure like octrees to be able to quickly calculate nearest neighbors. The only interaction among the points comes during the calculation of the normalization terms of the M step, which is the sum of the expectations of the correspondence variable. In the M step on the k^{th} iteration, we calculate

$$\hat{N}_j^{(k)} = \sum_i \gamma_j(\mathbf{z}_i) \tag{7}$$
$$\mu_j^{(k)} = \frac{1}{\hat{N}_j^{(k)}}\sum_i \gamma_j(\mathbf{z}_i)\mathbf{z}_i \tag{8}$$
$$\mathbf{\Sigma}_j^{(k)} = \frac{1}{\hat{N}_j^{(k)}}\sum_i \gamma_j(\mathbf{z}_i)(\mathbf{z}_i-\mu_{\mathbf{j}}^{(\mathbf{k})})(\mathbf{z}_i-\mu_{\mathbf{j}}^{(\mathbf{k})})^T \tag{9}$$
$$\pi_j^{(k)} = \frac{\hat{N}_j^{(k)}}{N} \tag{10}$$

Each $\hat{\gamma}_j(\mathbf{z}_i)$ can be found independently and in parallel as long as $\mathbf{\Theta}$ is known. Note that we can rewrite our calculation of $\gamma_j(\mathbf{z}_i)$ as

$$\gamma_j(\mathbf{z}_i) = \frac{\hat{\gamma}_j(\mathbf{z}_i)}{\sum_k \hat{\gamma}_k(\mathbf{z}_i)} \tag{11}$$

The normalized gammas can be found in a logarithmic number of steps by doing a sum reduction.

Note that we can remove the dependence on $\mu_j^{(k)}$ when calculating $\mathbf{\Sigma}_{\mathbf{j}}^{(\mathbf{k})}$ by rewriting the calculation as

$$\mathbf{\Sigma}_{\mathbf{j}}^{(\mathbf{k})} = \frac{\sum_i \gamma_j(\mathbf{z}_i)\mathbf{z}_i\mathbf{z}_i^T}{\hat{N}_j^{(k)}} - \frac{(\sum_i \gamma_j(\mathbf{z}_i))(\sum_i \gamma_j(\mathbf{z}_i))^T}{\hat{N}_j^{(k)}} \tag{12}$$

Given this reformulation, the parameter updates can also be calculated using a single sum reduction.

Past work from the machine learning community has looked at data parallel algorithms for EM to find GMM parameters [9] [10]. Unlike these works, we do not restrict our covariances to be spherical or axis-aligned elliptical, allowing non-zeros on the non-diagonal. We can calculate the full 3D covariance matrix since the problem of point cloud registration is relatively low dimensional (where the dimension $d \ll N$). In the previous work in this area, the authors assumed that inverting a high dimensional matrix for the expectation calculation would be prohibitively expensive if the matrix is non-sparse. Since we are dealing with 3D data, we can afford this luxury. Furthermore, we need the Gaussians to correspond with real 3D objects that often are not aligned nicely against the principle axes.

IV. Registration

Given that we have a world model parameter set $\mathbf{\Theta}$, We can describe the registration problem as follows: Given a set of 3D points, we can look at the probability of a scene having generated those points according to some surface model. If the probability of a point is $p(\mathbf{z_i}|\mathbf{\Theta})$, where $\mathbf{\Theta}$ is the set of parameters describing the scene, then the total probability of an entire scan of data is $p(\mathbf{Z}|\mathbf{\Theta}) = \prod_i p(\mathbf{z_i}|\mathbf{\Theta})$. In our problem, we wish to optimize over some spatial transformation in 3D space. Parametrizing our rigid transformation by a 3D vector $\mathbf{t} = (t_x, t_y, t_z)^T$ and a unit quaternion $\mathbf{q} = (u, v, w, s)^T$, we can minimize over the negative log-likelihood of the data with respect to the transformation parameters,

$$\min_{\mathbf{q},\mathbf{t}} -\ln p(T(\mathbf{Z}|\mathbf{q},\mathbf{t})|\mathbf{\Theta}) = -\sum_i \ln p(T(\mathbf{z}_i|\mathbf{q},\mathbf{t})|\mathbf{\Theta})$$

Using the EM algorithm, our surface model is just the collection of Gaussians and their respective mixing parameters, $\mathbf{\Theta} = \{\mathbf{\Theta}_j = (\pi_j, \mu_j, \mathbf{\Sigma}_j)\}$, so our data log-likelihood becomes

$$\ln p(T(\mathbf{X}|\mathbf{q},\mathbf{t})|\mathbf{\Theta}) = \sum_i \ln\{\sum_j \pi_j \mathcal{N}(\tilde{\mathbf{z}}_i|\mathbf{\Theta}_j)\} \tag{13}$$

where $\tilde{\mathbf{z}}_i = R(\mathbf{q})\mathbf{z}_i + \mathbf{t}$ and $R(\mathbf{q})$ is the 3x3 rotation matrix derived from the unit quaternion $\mathbf{q}$. Note the similarity to the likelihood function given by Equation 2 maximized during the segmentation step. In this step, however, we have a fixed $\mathbf{\Theta}$ and we optimize over $\mathbf{q}$ and $\mathbf{t}$. As before, we cannot find a closed form solution due to the sum inside the logarithm, but we can easily take the gradient of the negative log likelihood with respect to each transformation parameter,

$$\nabla_{\mathbf{t}} = \frac{1}{N}\sum_i \frac{\sum_j \hat{\gamma}_j(\tilde{\mathbf{z}}_i)(\tilde{\mathbf{z}}_i - \mu_j)^T \mathbf{\Sigma}_j^{-1}}{\sum_j \hat{\gamma}_j(\tilde{\mathbf{z}}_i)} \tag{14}$$

$$\nabla_{\mathbf{q}} = \frac{1}{N}\sum_i \frac{\sum_j \hat{\gamma}_j(\tilde{\mathbf{z}}_i)(\tilde{\mathbf{z}}_i - \mu_j)^T \mathbf{\Sigma}_j^{-1}\mathbf{\Xi}_{\tilde{\mathbf{z}}_i}}{\sum_j \hat{\gamma}_j(\tilde{\mathbf{z}}_i)} \tag{15}$$

where $\mathbf{\Xi}_{\tilde{\mathbf{z}}_i}$ is the skew symmetric matrix of $\tilde{\mathbf{z}}_i$, the transformed point,

$$\mathbf{\Xi}_{\tilde{\mathbf{z}}_i} = \begin{bmatrix} 0 & -\tilde{z}_i & \tilde{y}_i \\ \tilde{z}_i & 0 & -\tilde{x}_i \\ -\tilde{y}_i & \tilde{x}_i & 0 \end{bmatrix}$$

The gradient with respect to the quaternion only gives updates to (u, v, w), with s fixed to 1. For more details, see [12].

We note that, like the E step of the segmentation algorithm, the $\hat{\gamma}_j$ values can be calculated in parallel and then summed as a reduction.

V. Extensions

As the algorithm has been presented thus far, we have a few free parameters such as the number of clusters, the step size to take along the gradient, and possible singularities when inverting the covariance matrix of a cluster. For noisy data, outliers should also be handled. We give our extensions to solve these problems in the next few sections.

A. Oversegmentation

One important parameter to consider when segmenting is J, the amount of clusters to use. To obviate the need for model selection or some other type of parameter search, we simply over-segment and then drop clusters without a lack of adequate support. This process also prevents singularities when doing EM for the GMM parameters, since clusters with low support produce low-rank covariance matrices. Since the final surface description is the linear superposition of Gaussians, it is perfectly acceptable in terms of the gradient calculation, for example, to have overlapping Gaussians describe the same surface.

B. Robust Boundaries

Unfortunately, range data is often corrupted by several sources of noise, from spurious readings or discretization errors in the sensor. Similarly, operations on point cloud data are often burdened by occlusion and sampling sparsity when the objects are far away or at bad grazing angles to the sensor. These aspects make the task of finding and filtering outliers very important, as they may adversely affect the performance of the point cloud segmentation and registration algorithm.

To reduce computational costs and enforce hard boundaries, some past work has used non-overlapping labels when doing EM for purely planar decomposition [4]. To achieve this, during the EM algorithm, all correspondence variables are set zero if they do not have the maximal expectation over all clusters. This method closely resembles the elliptical k-means algorithm, where the Mahalanobis distance from the center of the cluster is used instead of the Euclidean distance.

We adopt a similar approach, but make some important modifications to filter out outliers. Instead of a hard boundary, or a purely soft boundary as the normal EM approach makes, we employ *robust* thresholding, where a correspondence is zero if its expectation is less than or equal to ξ times the maximum expected correspondence, where $\xi \in [0, 1]$. Our approach is a generalization of hard and soft boundaries. When $\xi = 0$, this algorithm reduces to soft boundaries, and when $\xi = 1$, this corresponds to the hard boundaries of [4].

C. Annealing and Line Search

The EM algorithm, although it is guaranteed to converge, will often converge to a local optimum instead of the global optimum. One problem with running the gradient descent algorithm over a GMM without annealing is that extremely planar surfaces will be described by nearly 2D Gaussians. In other words, the minimum eigenvalue of the covariance matrix will approach zero. In terms of surface reconstruction, this effect is desirable, since the surface will be very well-described by a degenerate Gaussian. In terms of gradient descent, however, a nearly 2D Gaussian will produce very steep peaks on the likelihood surface that do not give very useful gradient information when the current iterated solution is far away. As a solution gets close to a peak, the gradient will be nearly zero up until the solution is near the mean, where the gradient will be an extremely high value and possibly cause overshooting or instability.

We adopt an annealing function to smooth out local minima and steep peaks formed by planar Gaussians. In our case, we add a decaying exponential function that adds a multiple of the identity matrix to the covariance matrix for each cluster,

$$\mathbf{\Sigma}_j^* := \mathbf{\Sigma}_j + \lambda(T)\mathbf{I} \tag{16}$$

where, if T is our current iteration

$$\lambda(T) = \epsilon + k_1 e^{-T/k_2} \tag{17}$$

This provides a "fattening" of the GMM clusters so as to have shallower gradients in an attempt to not oscillate between local minima. In our experiments, we found a wide array of values for k_1, k_2 to be acceptable. We use $k_1 = 0.01$ and $k_2 = 20$, with $\epsilon = 1e - 5$.

As an additional protection against taking steps that may cause oscillations or jumps to non-optimal solutions, we do a line search over our step size. Note that this search requires repeated computation of the log likelihood of a given rigid transformation parameter set, which can be done fully in parallel for each $(\mathbf{z}_i, \mathbf{\Theta}_j)$ tuple and then summed as a reduction.

VI. Implementation and Algorithm

As a shorthand, we will refer to our algorithm as REM-Seg, or Robust Expectation Maximization Segmentation.

At a high level, the REM-Seg algorithm does the following:

E step: We measure the probability of each point having been generated by each cluster, given its current mean and covariance. We then assign the correspondence for each point as its expectation or as zero, according to the highest probability cluster and ξ. At the end of the E step, we have a new set of correspondence values for each point.

M step: We gather each point weighted by its expectation and calculate the most likely mean and covariance given all the points that were assigned to the cluster. At the end of the M step, we end up with a new set of parameters for the Gaussians.

Gradient step: We transform each point according to our current transformation guess and then calculate the annealed gradient with respect to the unit quaternion and translation vector representing the 6-DOF spatial transformation. We then line search to find the step size to increase the overall data log likelihood.

See Algorithm 1 for pseudocode.

```
Data: Two sets of PCD to be registered: $\mathbf{Z}$, $\mathbf{Z_2}$
Result: $\mathbf{q}$, $\mathbf{t}$
Initialize
$\mathbf{\Theta}_j = \{\pi_j, \mu_j, \mathbf{\Sigma}_j\} = \{1/J, \mathbf{r}, I\} \forall j \in \{1, ..., J\}$, where
$\mathbf{r}$ is a random vector inside of the unit cube.;
while not converged do
    Calculate correspondences $\gamma_j(\mathbf{z}_i) \forall \mathbf{z}_i \in \mathbf{Z}$ (in
    parallel, Equation 11);
    Apply robust thresholding to $\gamma_j(\mathbf{z}_i)$ (in parallel);
    Update $\mathbf{\Theta}$ (parallel reduction, Equations 7-10);
    Drop unsupported $\mathbf{\Theta}_j$;
end
Initialize $\mathbf{q} = (0, 0, 0, 1)^T$ and $\mathbf{t} = (0, 0, 0)^T$;
while not converged do
    Apply annealing step to $\Sigma_j, \forall j = \{1, .., J\}$
    (Equations 16,17);
    Calculate $\nabla_q, \nabla_t$ given $\mathbf{Z_2}$ and current $\mathbf{q}, \mathbf{t}$ (in
    parallel, Equations 14,15);
    Perform line search to find $\alpha$ (parallel
    log-likelihood calculations, Equation 13);
    $q := q + \alpha \nabla_q$ (parallel reduction);
    $t := t + \alpha \nabla_t$ (parallel reduction);
end
```

Algorithm 1: Pseudocode for REM-Seg

VII. Related Work

A. ICP

First developed by Besl and McKay [1], ICP works to iteratively associate points between two point clouds and the minimize the squared error between the associated points under rigid transformation parameters. The basic steps are as follows:

1) Sample points from each cloud (or use entire cloud)
2) For each point in a given cloud, find a corresponding point (e.g. via Euclidean distance)
3) Assign weights to the correspondences
4) Reject outlier pairs
5) Apply an error metric to the current transform (e.g. sum of squared error)
6) Use an optimization technique to minimize the error metric (e.g. SVD, gradient descent, Newton's method, etc.)

ICP is the *de facto* standard for many registration applications, but can easily succumb to misalignment when point matching between clouds is inappropriate, for example, if two surfaces are sampled very differently or there is a large transformation between clouds.

B. Point Cloud Parametrizations

Many registration algorithms first begin with an explicit or implicit parametrization of the point cloud into a form that provides continuous point estimates. Many of the fastest parametrization are in the form of planar models. The task of point cloud registration is then reduced to finding the best transformation of a set of planes [4][5][6]. Given a set of new points, the corresponding point in the scene can simply be the closest surface in terms of point-to-plane distance. In this way, the matching process has better convergence properties than ICP, since the matching is from points to corresponding surfaces and not simply to other sampled points. Furthermore, since many indoor environments are roughly rectilinear, the planar decomposition of the environment given a point cloud is somewhat justified. However, due to clutter, occlusion, and non-uniform sampling density, it may be inappropriate to rely on planar segmentation when a richer representation is required.

One may also use distance transforms or Gaussian Mixture Models over grids to describe the point cloud [7] [8]. These techniques have the added benefit of being able to capture a richer set of world models. Normal Distance Transform methods discretize the point cloud into cubic regions (voxels) and compute Gaussian distribution parameters for each voxel [13]. Given that each voxel now contains a probabilistic interpretation of a point falling into it, the likelihood has a well-defined gradient and the optimization over the rigid transformation can be more robust than a method like ICP. Imposing a grid presents certain technical issues and computational limitations, however, since the grid boundaries may lie across objects, and small grid sizes incur large amounts of GMM parameters.

C. Softassign and EM-ICP

EM-ICP uses annealing with isotropic Gaussian noise around each point with point correspondences as a hidden variable solved via EM. Similarly, Softassign also uses multiply labeled correspondences. Both are similar to ICP using a Mahalanobis distance metric. The problem is that the probability of a point matching to another point nearby depends also on the local surface characteristics. That is, a point matching to a plane should be more probable if the translation is not along normal, but along with the two principal axes of the plane. Also, GMM segmentation is useful in its own right, allowing for a low-memory representation of the PCD. Few GMM parameters make the algorithm

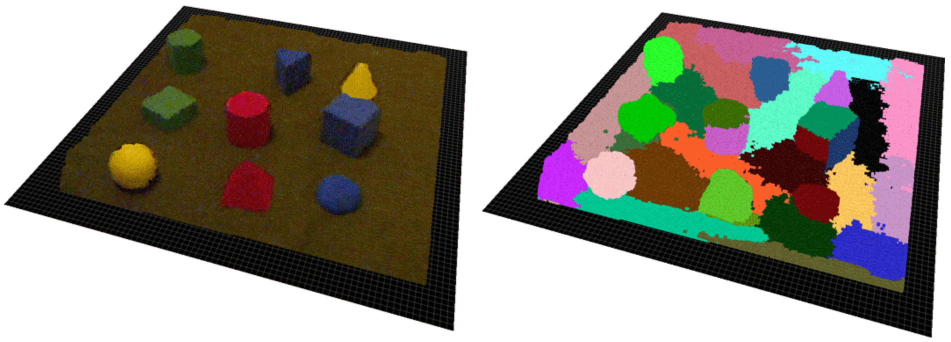


Fig. 1: Example surface patches with each point colored by its most likely cluster. Note that since the GMM formulation is additive with respect to each cluster, over-segmenting serves to more accurately construct a PDF of scene.

more efficient since new points only need to be compared to $J \ll N$. This computational gap can be narrowed with maximum distance thresholding and k-D trees, except for the case of EM-ICP, when annealing, since the PCD must be decimated to avoid the need for a search over all other points.

D. GMM-Reg

GMM-Reg is another registration algorithm based upon the idea of using Gaussian Mixture Models as the base structure for a continuous optimization procedure [14][15]. Given a Gaussian with fixed bandwidth around each point for both model and scene, GMM-Reg finds the maximum cross correlation between the two distributions.

VIII. Experiments

For the following experiments, we use two datasets: a cross-section of the Stanford Bunny [16] and a point cloud generated from a Kinect of a table-top scene with various geometric shapes on top. A depiction of the table-top scene is given by the left image in Figure 1. Both datasets are scaled to fit within the unit cube.

We compare our algorithm, REM-Seg, with ICP, EM-ICP [17] [18], Softassign [19], and GMM-Reg [14]. Additionally, we test our method against GPU-accelerated versions of EM-ICP and Softassign [20]. The CPU versions of EM-ICP and ICP are accelerated with OpenMP. The algorithms to which we compared REM-Seg are all freely available to download. In particular, our implementation of ICP was taken from work described in [20]. All experiments were done on a quad-core CPU (Intel Q6600) and Nvidia GPU (GTX 680).

A. Segmentation

The Kinect table-top scene is shown in Figure 1. The colorized point cloud is shown on the left and the segmentation is shown on the right. The full scene contains over 50,000 points. We color the points according to the maximal expected cluster membership. Note that the GMM algorithm represents an "over-segmentation" in that some homogenous surfaces are broken up into separate clusters. For purposes of segmentation for perception tasks, one could trivially join together like clusters. For instance, clusters representing the ground plane will all have a very similar normal, and so a normal space clustering could be used in conjunction with Euclidean distance to join them. For purposes of registration, over-segmentation does not have any adverse effects since the GMM model adds individual cluster probabilities. In fact, in the limit of one cluster per point, the REM-Seg algorithm reduces to an algorithm somewhat similar to EM-ICP.

We can see that the sample segmentation of Figure 1 has some nice properties compared with similar planar decompositions. When a facet is large enough, it is described by a mostly flat Gaussian. Examples of this effect can be seen in the cube shape and the top of the larger cylinder. However, when the shape does not have as many supporting points, it is described by a single 3D Gaussian, as shown by the smaller shapes, such as the sphere. Describing the sphere by a 3D Gaussian provides a much better gradient than a planar segmentation of the same object.

B. Registration Quality

To test the convergence or "goodness-of-fit" properties of the various algorithms, using the Stanford Bunny, we randomly and uniformly sampled 100 rigid transformations from the space of transformations bounded by rotations of less than 15 degrees around any single axis, and translations bounded by 2 times the extent of the dataset in each dimension. Each algorithm was given the same randomly transformed and subsampled cloud and run until convergence. The squared error (SE) was then calculated. Figure 2 shows the distribution of the SE for each algorithm. The x-axis is the SE and the y-axis represents the number of trials that fell into the particular bin. We also repeated these experiments with the Kinect table-top scene with similar results, so for space reasons we have omitted them.

We found that our convergence was much more robust to random rigid transformations than ICP, Softassign, and EM-ICP. Even with potentially large transformations, our algorithm found a solution close to zero SE, whereas the other algorithms diverged. ICP, especially, succumbed to many local optima that represented globally non-optimal solutions. Our method most closely resembles GMM-Reg in terms of convergence, though 15 of the 100 GMM-Reg

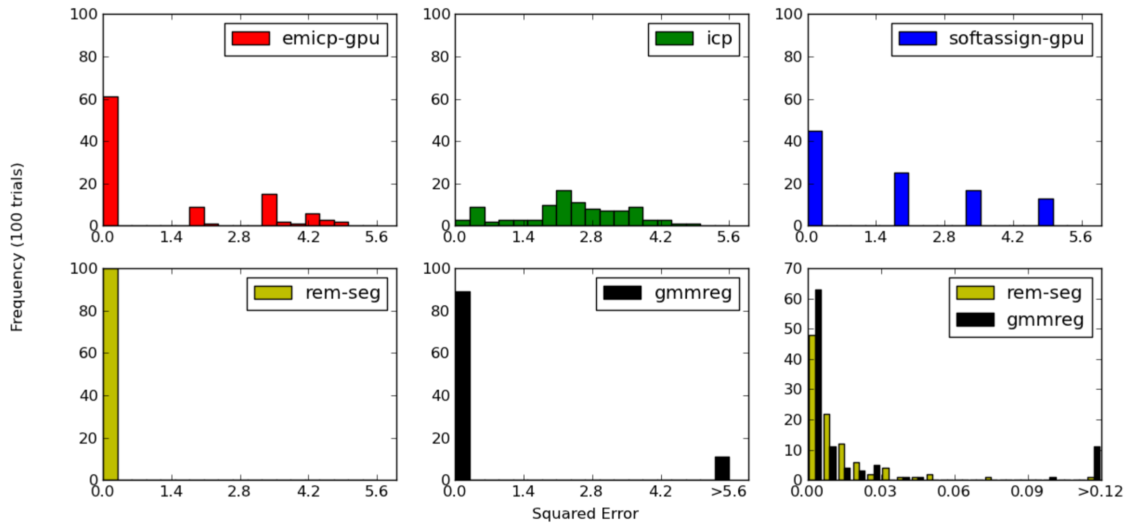
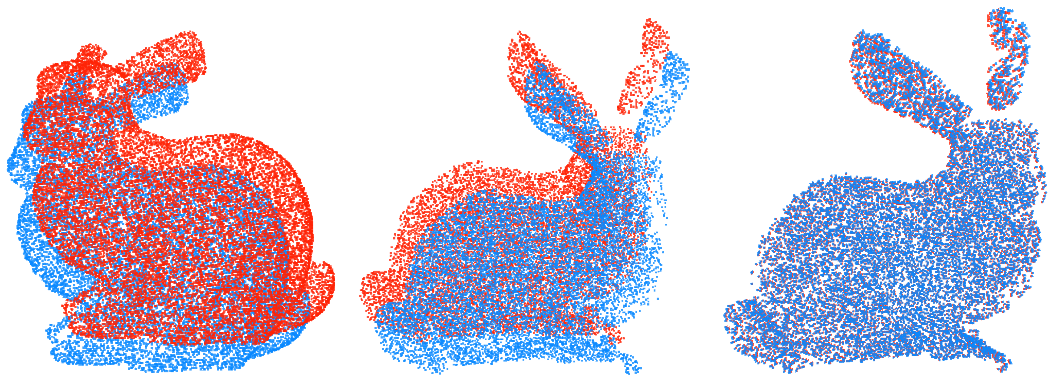


Fig. 2: Goodness of fit using the Bunny dataset. These convergence results are from 100 random rigid transformations. REM-Seg consistently converges to near-zero error while the other algorithms tend to get stuck at local minima. The bottom right subfigure shows the two best algorithms with a zoomed x-axis.



(a) Sq. Error ≈ 0.16 (b) Sq. Error ≈ 0.13 (c) Sq. Error ≈ 7.5e-3

Fig. 3: Example Registration Results

trials diverged to a wildly wrong registration result, while Rem-Seg fell into local minima that were closer to zero. Figure 3 shows example cases for various squared errors. Note that Figure 3a and 3b represent cases that are less than 0.20, compared to the local minima encountered by EM-ICP, ICP, and Softassign, which often were 3.0 or more. Figure 3c represents a correctly converged case.

C. Scaling and Speed

Using the Kinect tabletop dataset, we randomly subsampled from its points and ran our registration algorithm over varying amounts of points. For each data point, we averaged the time to convergence over 10 random rigid transformations of the scene.

The timing results can be seen in Figure 4. The figure on the left shows the y-axis as a log scale, giving the time to convergence. As the number of points increases, we can see that the scaling of all algorithms but EM-ICP and REM-Seg are poor. Note that regular ICP is the fastest of all algorithms when the number of points is very small, but the convergence is typically poorer. The right plot is the same data but on a linear y-axis to show more clearly the difference between the two best algorithms. We can clearly see that the scaling difference grows superlinearly, favoring REM-Seg. Since the GPU-accelerated EM-ICP relies mainly on accelerated linear algebra routines, our REM-Seg is faster since it presents a naturally data parallel solution for the GMM problem. Though not shown in the Figure, we can run REM-Seg over much larger point clouds, achieving sub-second registrations up to just under one million points. In general, the execution times scale roughly logarithmically until the parallelism limit is reached on the GPU, after which the scaling is mostly linear. With our current hardware, this limit is reached at around 100,000 points.

Though GMM-Reg was comparable to REM-Seg in terms of the goodness-of-fit of the final registration, the fact that it does not currently run on parallel hardware is a severe limitation when processing many points. Furthermore, the

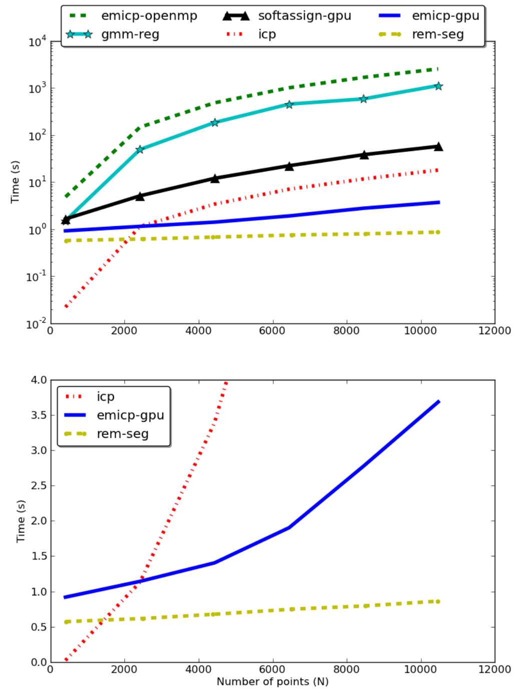


Fig. 4: Timing results as the number of points in the cloud to be matched to the scene increases for the Kinect dataset. The top plot shows the time to convergence on a log scale, while the bottom plot shows the same data on a linear y-axis among ICP, EM-ICP-GPU, and REM-Seg.

related methods rely on Gaussians around every point in the cloud, while REM-Seg efficiently compresses the clouds into a sparse amount of parameters.

IX. Future Work and Conclusion

Given that the REM-Seg algorithm can segment the data into spatially consistent chunks, we could potentially extract surface statistics from these chunks and use them as the basis for algorithms that rely on surface-based signatures, such as certain types of landmark-based SLAM or object detection and recognition.

GPU's and other many-core architectures offer many benefits when dealing with point cloud processing algorithms that have been designed to be data parallel. We have shown one such segmentation and registration algorithm, REM-Seg, to scale very nicely with the size of a point cloud and to have a wide range of convergence over rigid transformations. Segmenting the point cloud first before applying a registration algorithm has the nice property of being robust to noise as well as allowing tractable and very efficient parallel gradient calculations. Overall, we feel that data parallel algorithms are a natural choice when coupled with many-core architectures for large-scale point processing.

References

[1] P. Besl and H. McKay, "A method for registration of 3-D shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, no. 2, pp. 239–256, 1992.

[2] N. Mitra, N. Gelfand, H. Pottmann, and L. Guibas, "Registration of point cloud data from a geometric optimization perspective," in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. Nice, France: ACM, 2004, pp. 22–31. [Online]. Available: http://dx.doi.org/10.1145/1057432.1057435

[3] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE, 2001, pp. 145–152.

[4] C. Martin and S. Thrun, "Real-time acquisition of compact volumetric 3D maps with mobile robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, Washington, DC, USA, Aug. 2002, pp. 311–316. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1013379

[5] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun, "Using EM to learn 3D models of indoor environments with mobile robots," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, p. 329336. [Online]. Available: http://portal.acm.org/citation.cfm?id=645530.655822

[6] S. Thrun, C. Martin, Y. Liu, D. Hahnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard, "A Real-Time Expectation-Maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 433–442, 2004. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1303689

[7] Y. Tsin and T. Kanade, "A correlation-based approach to robust point set registration," *Computer Vision-ECCV 2004*, pp. 558–569, 2004.

[8] B. Jian and B. Vemuri, "A robust algorithm for point set registration using mixture of gaussians," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1246–1251.

[9] N. Kumar, S. Satoor, and I. Buck, "Fast parallel expectation maximization for gaussian mixture models on gpus using cuda," in *High Performance Computing and Communications, 2009. HPCC'09. 11th IEEE International Conference on*. IEEE, 2009, pp. 103–109.

[10] L. Machlica, J. Vanek, and Z. Zajic, "Fast estimation of gaussian mixture model parameters on gpu using cuda," in *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2011 12th International Conference on*. IEEE, 2011, pp. 167–172.

[11] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[12] M. Wheeler and K. Ikeuchi, *Iterative estimation of rotation and translation using the quaternion*. School of Computer Science, Carnegie Mellon University, 1995.

[13] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2003, pp. 2743–2748.

[14] B. Jian and B. C. Vemuri, "Robust point set registration using Gaussian mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1633–1645, 2011. [Online]. Available: http://gmmreg.googlecode.com

[15] ——, "A robust algorithm for point set registration using mixture of Gaussians." in *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, 2005, pp. 1246–1251.

[16] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 303–312.

[17] S. Granger and X. Pennec, "Multi-scale em-icp: A fast and robust approach for surface registration," *Computer VisionECCV 2002*, pp. 69–73, 2006.

[18] Y. Liu, "Automatic registration of overlapping 3d point clouds using closest points," *Image and Vision Computing*, vol. 24, no. 7, pp. 762–781, 2006.

[19] S. Gold, A. Rangarajan, C. Lu, S. Pappu, and E. Mjolsness, "New algorithms for 2d and 3d point matching:: pose estimation and correspondence," *Pattern Recognition*, vol. 31, no. 8, pp. 1019–1031, 1998.

[20] T. Tamaki, M. Abe, B. Raytchev, and K. Kaneda, "Softassign and em-icp on gpu," in *Networking and Computing (ICNC), 2010 First International Conference on*. IEEE, 2010, pp. 179–183.