Registration of 3D Point Sets Using Correntropy Similarity Matrix

Ashutosh Singandhupe†, Hung Manh La† 1, Trung Dung Ngo, Van Ho

Ashutosh Singandhupe and Dr. Hung La are with the Advanced Robotics and Automation (ARA) Laboratory, Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA

Dr. Trung Dung Ngo is with the More-Than-One Robotics Laboratory, University of Prince Edward Island, Canada

Dr. Van Ho is with the Soft Haptics Laboratory, School of Materials Science, Japan Advanced Institute of Science and Technology (JAIST), Japan

Abstract

This work focuses on Registration or Alignment of 3D point sets. Although the Registration problem is a well established problem and it's solved using multiple variants of Iterative Closest Point (ICP) Algorithm, most of the approaches in the current state of the art still suffers from misalignment when the Source and the Target point sets are separated by large rotations and translation. In this work, we propose a variant of the Standard ICP algorithm, where we introduce a Correntropy Relationship Matrix in the computation of rotation and translation component which attempts to solve the large rotation and translation problem between Source and Target point sets. This matrix is created through correntropy criterion which is updated in every iteration. The correntropy criterion defined in this approach maintains the relationship between the points in the Source dataset and the Target dataset. Through our experiments and validation we verify that our approach has performed well under various rotation and translation in comparison to the other well-known state of the art methods available in the Point Cloud Library (PCL) as well as other methods available as open source. We have uploaded our code in the github repository for the readers to validate and verify our approach https://github.com/aralab-unr/CoSM-ICP

© 2011 Published by Elsevier Ltd.

Keywords: Registration, 3D point cloud, Correntropy criterion, Iterative Closest Point

1. Introduction

Registering two point clouds is an active area of research in the community. Registration is the process of aligning two point cloud datasets of the same scene, which are separated by a certain transformation (rotation and translation). The problem of registration can be explained as finding the transformation between two point clouds such that the relative error between the 2 point sets is minimized. The problem of registration is a significant component in Simultaneous Localization and Mapping (SLAM) for robotics and other visual data matching algorithms. The relative transformation computed from point clouds acquired at close time intervals from a 3D sensor can be used to compute the overall trajectory of a system.

Early approaches for 3D point cloud registration can be attributed to Besl and MacKay and Arun et.al, which consists of computing the centroid for both Source and Target datasets. Each point from both the datasets is subtracted...
from their respective centroids and later on, Singular Value Decomposition (SVD) is used to compute the rotation component. This procedure is performed iteratively until the relative mean square error between the point cloud datasets is minimized, and hence it is termed as Iterative Closest Point (ICP) Algorithm. Since then variants of ICP have been introduced, a point-to-plane variant of ICP operates on taking advantage of surface normal information. Instead of minimizing the root mean square of the individual points as done in the generalized version of the ICP, the point to plane algorithm minimizes error along the surface normal.

Despite the various advances in the registration methodologies, handling the situation where the data set is noisy is still a major concern. If the noise is Gaussian, we can remove most of it from the data. However, when the noise is non-Gaussian (e.g., shot noises), it becomes difficult to resolve. Hence, it is important to address the problem of handling non-Gaussian noise. The idea of Correntropy has seen widespread use in modern machine learning, signal processing, and other related fields. Correntropy is a measure of similarity between two random variables and has been implemented with the traditional Kalman filter. Some recent work has introduced the Correntropy concept to the registration problem to overcome the drawbacks of the ICP algorithm. In the traditional ICP algorithm, if any of the point cloud dataset (Source or Target) is affected by non-Gaussian noise, the ICP fails drastically and affects the overall transformation estimation of the point clouds. In this work, we show through basic examples on how the addition of non-Gaussian noise can affect the traditional ICP as well as compare how our approach can be better than the traditional ICP algorithm and its variants. We also compare our approach with the well known Normal Distribution Transform (NDT) algorithm available in the PCL library.

In this work, we propose a Correntropy Similarity Matrix Iterative Closest Point (CoSM ICP) algorithm. In our approach, we introduce a matrix of size $N \times N$ which maintains a ‘numeric’ relationship between Source and Target dataset points (size of Source dataset and Target dataset is $N$, hence the matrix size is $N \times N$). This numeric relationship is calculated through the well-known correntropy criterion, and it is updated at every iteration. It is calculated only between the nearest points from Source dataset to the Target dataset, thereby making the Similarity Matrix a sparse matrix. Our approach not only provides a significant improvement in the alignment of Source and Target dataset under various rotations and translations, it also proves quite effective against non-Gaussian outliers affecting the dataset. In addition to evaluating our approach in multiple datasets, we also evaluate the method where we inject the Source dataset with random false values and through experimental comparison with the traditional ICP and other approaches, we evaluate how well our approach can estimate the transformation between the ‘infected’ Source and the Target dataset and we see how our approach performs better than the well known state of the art methods. Along side estimating accurate transformations, handling shot-noise or the non-Gaussian noise vectors is a critical component for better transformation estimation and is the driving force for our work.

Inspired by this Correntropy concept, we fuse it with the ICP algorithm, called Correntropy Similarity Matrix ICP or CoSM-ICP. Our main contribution to this work can be summarized as follows:

- We propose a new data registration algorithm in which the Correntropy Similarity Matrix ICP is introduced.
- We evaluate our approach on various randomly generated transformation matrices between Source and the Target.
- Through evaluation on these random transformation matrices we see how well our approach performs better than the other well known state of the art methods.
- We also introduce outliers in the Source dataset and evaluate our approach on the ‘infected’ Source dataset.
- We compare our approach with other approaches on the ‘infected’ Source dataset and see through results how well our approach performs better.

The remaining paper is organized as follows: Section introduces the idea of Correntropy and its underlying concepts. Section describes our proposed method and it’s implementation. The results and evaluation of our proposed method are discussed in Section. Conclusions and future work discussion are given in Section V.
Table 1: Nomenclature

| Variables | Usages |
|-----------|--------|
| $P_s$    | Source Point Cloud Set. |
| $P_t$    | Target Point Cloud Set. |
| $N$      | Total number of points in point clouds $P_s$ and $P_t$. |
| $j, k$   | Index of every point in point cloud where $j, k = 1, \ldots, N$. |
| $p_j$    | Individual 3D point $(x_j, y_j, z_j)$ in a point cloud $P_s$, or $p_j \in P_s$. |
| $q_k$    | Individual 3D point $(x_k, y_k, z_k)$ in a point cloud $P_t$, or $q_k \in P_t$. |
| $G_\sigma$ | Gaussian kernel function with bandwidth $\sigma$. |
| $R$      | Rotation matrix. |
| $tr$     | Translation vector. |
| $\epsilon^2$ | Root mean square error. |
| $s$      | Scaling factor. |
| $SM$     | Similarity Matrix. |
| $s_{cen, t}$ | Centroids computation of Source and Target datasets. |

2. Correntropy Criterion

We begin by describing the mathematical representation of point cloud data and develop our algorithm based on it. We denote Source point cloud as $P_s$ and Target point cloud as $P_t$. The points in the point cloud $P_s$ contain $N$ points in which each point can be referenced as $p_j = \{x_j, y_j, z_j\}$, where $x_j, y_j, z_j$ denotes the 3D coordinates of the point $p_j$. Similarly, the points in the point cloud $P_t$ contain $N$ points in which each point can be represented as $q_k = \{x_k, y_k, z_k\}$, where $x_k, y_k, z_k$ denotes the 3D coordinates of the point $q_k$. Throughout the paper, we describe a matrix as bold and capitalized letters, a vector is represented as bold and lowercase letters and the rest of the variables (single dimension) are represented as lowercase variables. Figure 1 shows a brief description of the point clouds and Table 1 denotes the mathematical notations used throughout this work.

![Figure 1](image-url)  

Figure 1: (a) Red is the Source point cloud $P_s$ (bunny rabbit). $P_s$ contains $N$ points $p_j \ (j = 1, \ldots, N)$, each of which is a 3D representation $x_j, y_j, z_j$. White is the Target point cloud $P_t$. $P_t$ contains $N$ points $q_k \ (k = 1, \ldots, N)$, each of which is a 3D representation $x_k, y_k, z_k$. Same goes for the dragon dataset in (b).

We use the Correntropy concept between the Source point cloud and the Target point cloud. Correntropy has proved beneficial to remove large outliers. Correntropy is essentially a similarity measure of two random variables. We use the Correntropy concept between Source and Target point clouds. In this scenario, we are measuring the similarity
between points $p_j$ and $q_k$ from the point clouds $P_s$ and $P_t$, respectively. Then the Correntropy criterion between points $p_j$ and $q_k$ can be mathematically represented as:

$$V_{\sigma}(p_j, q_k) = E[\kappa_{\sigma}(p_j - q_k)].$$

(1)

Equ. (1) also refers to in common literature as a cross-Correntropy of two random variables. In Equ. (1), $E[.]$ refers to the expectation of the variable, and $\kappa_{\sigma}$ denotes the kernel function. We can manually choose the size of the kernel function (also referred to as bandwidth $\sigma$). In our approach, we use the Gaussian kernel function where we redefine the Correntropy function as:

$$V_{\sigma}(p_j, q_k) = \frac{1}{N} \sum_{j=1}^{N} G_{\sigma}(p_j - q_k).$$

(2)

From Equ.(2),

$$G_{\sigma}(p_j - q_k) = \exp\left(-\frac{\|p_j - q_k\|^2}{2\sigma^2}\right),$$

(3)

where $\sigma$ is the bandwidth or the kernel size of the Gaussian kernel. From Equ. (3) it becomes evident that if $p_j = q_k$, Gaussian Correntropy is maximum, and the Gaussian Correntropy function is positive and bounded.

Figure 2: Correntropy Criterion. Here, when the difference between the points ($p_j - q_k$) is small, i.e., when they are similar, and the Gaussian kernel function $G_{\sigma}$ approaches to 1. On the contrary, when the difference is large, the Gaussian kernel function approaches to 0.

Fig. 2 shows the basic understanding of the Correntropy criterion. We see that when the difference between the points ($p_j - q_k$) is small, i.e., they are similar, and the Gaussian kernel function $G_{\sigma}$ approaches to 1. On the contrary, when the difference is large, the Gaussian kernel function approaches 0. This essentially means that when a potentially changed data point (for e.g., $q_k$), is far varied from $p_j$, the Gaussian kernel signifies that the points $p_j$ and $q_k$ are not similar. Intuitively, this similarity is dependent on the value of $\sigma$. This technique allows us to reject the faulty data point (shot noise or non-Gaussian noise) $q_k$. This idea of calculating the maximum similarity as given in Equ. (3) is called the maximum Correntropy criterion.
When the value of \( q_k \) approaches \( \infty \) then the kernel function approaches 0, i.e.,

\[
\lim_{q_k \to \infty} G_{\sigma} = 0.
\]

Intuitively, when the Gaussian Kernel shrinks to 0, the Correntropy approaches the value \( \rho(p_j = q_k) \).

3. Proposed Method

3.1. Correntropy Similarity Matrix with Iterative Closest Point Algorithm

Given the general description of the Correntropy concept in the previous section, we extend this idea of Correntropy to the well known ICP algorithm. Traditional ICP \(^3\) describes the alignment of point clouds such that the Mean Square Error (MSE) between point sets is minimized. The MSE is the key criterion of convergence in ICP and its variants.

ICP revolves around the problem of aligning point sets \( P_s \) and \( P_t \). We denote the point sets as \( P_s = \{p_j\}_{j=1}^N \) (here \( \{ \) denotes a point set) and \( P_t = \{q_k\}_{k=1}^N \), respectively. Our goal is to best align Source point set \( P_s \) to Target point set (or a model point set) \( P_t \). This leads to finding the appropriate rotation and translation between the two point sets, which essentially means how much units the Source dataset needs to be translated (\( \text{tr} \)) and rotated (\( R \)) such that Source point set \( P_s \) is best aligned with \( P_t \) given that it satisfies a particular criterion (Root Mean Square Error (RMSE) threshold).

This process is commonly known as registration between the Source \( P_s \) and the Target \( P_t \) as defined as finding the translation \( \text{tr} \) and the rotation \( R \) component such that \( P_s \) is in best alignment with \( P_t \).

Now, the problem can be written as: find the best \( R \) and \( \text{tr} \) such that the MSE, \( \varepsilon^2 \), between 2 point clouds is minimized. This problem can be mathematically written as,

\[
\varepsilon^2 = \sum_{j,k=1}^N \left\| p_j - (sRq_k + \text{tr}) \right\|^2,
\]

where \( s \) is the scaling component. \( p_j \in P_s \) is the set of all the points in the Source point set, and \( q_k \in P_t \) is the set of all the points in the Target point set (\( j, k = 1, ..., N \)).

Our work is built on the work of Arun et al.\(^3\) and it is well known from their work that initially we compute the corresponding points between Source and Target point sets. The Corresponding points in this context are computed as shortest point from each point in the Source dataset to the Target dataset. This can be efficiently computed using k-d tree data structure. We introduce an additional parameter using the Correntropy criterion to find the similarity metric between points. Now, we say that for a Source point index \( i \) the corresponding point index in the Target point set is \( c(i) \) (\( c \) is vector of corresponding points from Source to Target). From Equ. we compute the similarity between the two corresponding points \( P_s \) and \( P_t \) as

\[
d = P_s(i) - P_t(c(i)),
\]

\[
G_{\sigma}(P_s(i) - P_t(c(i))) = \frac{1}{(2\pi\sigma)^{D/2}} \exp\left(-\frac{d^2}{2\sigma^2}\right).
\]

Here, \( D \) is the dimensions and \( D = 3 \) in this case.\(^{23}\) Now, we can create a similarity matrix between the Source and the Target points based on the similarity metric computed in Equ.\(^6\). Intuitively, the size of the similarity matrix is \( N \times N \). In every iteration we initialize the value of this similarity matrix as zeros and update them in every iteration based on the similarity metric computed above in Equ.\(^6\). It is updated as shown below:

\[
\text{SM}(i, c(i)) = G_{\sigma}(P_s(i) - P_t(c(i))),
\]

\[
\text{SM}(c(i), i) = G_{\sigma}(P_s(i) - P_t(c(i))),
\]

where SM is the similarity matrix which we intend to use in the computation of the rotation component. The centroid of both the point clouds i.e., the Source and Target is computed as given by Equ.\(^8\)
Figure 3: Similarity Matrix. ’*’ represent the Correntropy values.

\[
\begin{align*}
\text{Target points index} & = \begin{bmatrix}
0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\
N \times N
\end{bmatrix}
\end{align*}
\]

where \( s_{cen} \) and \( t_{cen} \) are the computed centroid of Source and Target data set respectively. The difference between the computed centroid \( s_{cen} \) and the individual Source points is computed. The same procedure is followed for the Target points. It is given in Equ. 9

\[
\begin{align*}
p_i' &= p_i - s_{cen}, \\
q_j' &= q_j - t_{cen}.
\end{align*}
\]

The Singular Value decomposition (SVD) for finding the rotation component is based on first finding the \( 4 \times 4 \) matrix as shown in Equ. 10 where we introduce our Similarity Matrix

\[
H = \sum_{i=1}^{N} p_i' S M q_i'^T.
\] (10)

The size of the \( SM \) matrix is \( N \times N \) where as the size of the \( p' \) matrix is \( 4 \times N \) (under homogeneous transformation) which results in \( 4 \times N \) matrix. The result when multiplied with \( q'^T \) (which is a \( N \times 4 \) matrix) will have the final result in a \( 4 \times 4 \) matrix form which is the size of the Matrix \( H \). The rest of the algorithm is the same as the traditional algorithm where we compute the SVD of \( H \) as given by Equ. 11

\[
H = U \Lambda V^T.
\] (11)

Now the Rotation component is calculated as given by Equ. 12

\[
R = V U^T.
\] (12)

The determinant of \( R \) must be a positive integer. The translation component is simply computed as the difference of the centroids computed in Equ. 9 Algorithm describes the entire procedure of our proposed method.
Algorithm 1 CoSM Algorithm

1: function ReadDataSets($P_s$, $P_t$) \hfill \triangleright Read the Source and the Target datasets.
2: \hspace{1em} while not converged do
3: \hspace{2em} $c \leftarrow \text{ComputeCorrespondence} (P_s, P_t)$. \hfill \triangleright Compute Correspondence
4: \hspace{2em} $SM = \text{zeros}(N, N)$ \hfill \triangleright Initialize $N \times N$ Similarity Matrix to all zeros.
5: \hspace{2em} for $i \leftarrow 1$ to $N$ do
6: \hspace{3em} $d = P_s(i) - P_t(c(i))$. \hfill \text{(Line 7 and 8)}
7: \hspace{3em} Compute $G_\sigma$ as shown in Equ.6.
8: \hspace{3em} $SM(i, c(i)) = G_\sigma(P_s(i) - P_t(c(i)))$. \hfill \text{(Line 8)}
9: \hspace{3em} $SM(c(i), i) = G_\sigma(P_s(i) - P_t(c(i)))$. \hfill \text{(Line 9)}
10: \hspace{2em} end for
11: \hspace{2em} Compute centroid as given in Equ.8.
12: \hspace{2em} Compute the difference as given in Equ.9.
13: \hspace{2em} Compute the $H$ Matrix as given in Equ.10.
14: \hspace{2em} Compute SVD of $H$ as given in Equ.11.
15: \hspace{2em} Compute $R$ as given in Equ.12.
16: \hspace{2em} Compute $tr$ as difference of centroids.
17: \hspace{2em} end while
18: \hspace{1em} end function

![Iterations vs Rank of Similarity Matrix](image)

Figure 4: As the iterations increases, the rank of the matrix approaches $N$. Here $N$ is 1360.

3.2. Properties of the Similarity Matrix

3.2.1. SM Matrix is a sparse Matrix

Since at every iteration we initialize the values to zeros and based on Equ.7 we fill the Correntropy similarity values at only specified location, the matrix is sparse. Line 7 and 8 in Algorithm 1 clearly indicated that we update only those indices in the Source and the Target point indexes which are close to each other. Fig.3 shows a simple structure of the similarity matrix.

3.2.2. The rows and columns are linearly independent

Every element in the SM matrix or $SM(i, j)$ represents the Correntropy relationship between the $i$th Source point and the $j$th Target point. Since we are computing the similarity of only the closest point from the Source index to the Target index, every other indices that particular row/column is intuitively zero. This clearly means for every row index in the Source point set there is only one component in the Target dataset with a similarity metric which assures one to one relationship between every Source point to every Target point. The points in the Source point set are transformed based on this rotation matrix after every iteration. This process is repeated every iteration and one can find that the number of points with closer similarity metric ($\sim 1$) increases. Fig.4 shows that as the iterations increases the rank approaches $N$. 

7
3.2.3. Similarity Matrix is a Mirror-Symmetric Matrix.

From Equ. [7] one can verify that if the matrix is diagonally separated, then it’s a mirror image of each other (Mirror-Symmetric Matrices) which means $\text{SM} = \text{SM}^T$.

4. Results

4.1. Evaluation on Datasets with no outliers.

We validated our approach on 3 simple and well known datasets: the Bunny Rabbit, the Happy Buddha and the Dragon dataset. In addition, we have also performed our evaluation on a Lidar dataset (KITTI dataset) since it captures data on a large scale. We perform our experiment on Intel i7 2.9 Ghz CPU with 32 Gb of RAM. We implemented our approach in the PCL library (version 1.9) and released our code in the github repository (https://github.com/aralab-unr/CoSM-ICP). In this section, we did not add any random outliers, and we introduce random rotation and translation between the Source and the Target point sets. First we capture a sample pcd file and transform the acquired point cloud with a random transformation matrix where all the individual components of the transformation matrix were randomly generated with different seeds (we assume throughout the paper that the units for angle are in radians). We name the original point cloud as read from the pcd file as Target and the randomly transformed point cloud as the Source. The problem statement is to find the relative transformation between the Source and the Target such that the Source is aligned with the Target. For the above steps, the algorithm is described in Algorithm 2.

Algorithm 2: Transform the point cloud with a Random Transformation Matrix.

1: function ReadPCD(filename:pcd,P).
2: $a_{ax} =$ Random angular component along the X-axis (-6.28 to 6.28 radians).
3: $a_{ay} =$ Random angular component along the Y-axis (-6.28 to 6.28 radians).
4: $a_{az} =$ Random angular component along the Z-axis (-6.28 to 6.28 radians).
5: $tr_{tx} =$ Random translation along the X-axis.
6: $tr_{ty} =$ Random translation along the Y-axis.
7: $tr_{tz} =$ Random translation along the Z-axis.
8: $M \leftarrow \text{GenerateTransformationMatrix}(a_{ax}, a_{ay}, a_{az}, tr_{tx}, tr_{ty}, tr_{tz})$
9: $P_s \leftarrow \text{TransformPointCloud}(P_t, M)$
10: return $P_s$
11: end function

![](https://github.com/aralab-unr/CoSM-ICP)

Figure 5: Rotated and translated point cloud (white is the original point cloud (Target), and Red is the transformed point cloud (Source)) with rotation of $(r, p, y) = (0.314, 0, 0)$ and translation of 0.05 unit in the z-axis. (a) shows the bunny rabbit point cloud down sampled using voxel grid filtering of leaf size 0.005. (b) shows the dragon point cloud down sampled using voxel grid filtering of leaf size 0.005. (c) shows the happy buddha point cloud down sampled using voxel grid filtering of leaf size 0.005.
Figure 6: Transformation from Source to Target: \((r, p, x, y, z) = (0.314, 0, 0, 0, 0.05)\). White point cloud: Original point cloud (Target). Red point cloud: Source point cloud and Green point cloud: Source transformed point cloud after 10 iterations using (a) ICP Standard SVD (RMSE: 6.25494e-06) (b) ICP Point to Plane (RMSE: 2.77175e-16) and (c) CoSM ICP (RMSE: 3.34226e-06) on the Bunny Rabbit dataset. Similarly we perform 50 iterations on the Dragon dataset using (d) ICP Standard SVD (RMSE: 2.86089e-14) (e) ICP Point to Plane (RMSE: 3.93822e-16) and (f) CoSM ICP (RMSE: 2.72838e-15) and 35 iterations on the Happy Buddha dataset using (g) ICP Standard SVD (RMSE: 1.72851e-13) (h) ICP Point to Plane (RMSE: 3.91357e-16) and (i) CoSM ICP (RMSE: 2.46336e-14).
To validate our approach on the datasets, we collected the results on a simple scenario where the Source is transformed manually with a simple transformation matrix which has less variation in rotation and translation \(((r, p, y, x, y, z) = (0.314, 0, 0, 0, 0.05))\). We reduce the number of point clouds to decrease the computational time by performing Voxel grid filtering (leaf size 0.005) on the point clouds. The original number of points in the bunny point cloud sets is 40256 points, and after applying voxel grid filtering we get the number of points reduced to 1360 points. Similarly for the Dragon dataset, the original number of points is 43467 points, and after voxel grid filtering the number of points is reduced to 1593 points. The original number of points for the Happy Buddha dataset is 76166 points, and after voxel grid filtering we get 1090 points. The Source and the Target point clouds for the datasets are shown in Fig. 5.

After applying few iterations, our approach aligns equally well (RMSE: 2.46336e-14) in comparison to the well known Voxel grid filtering (leaf size 0.005) on the point clouds. The results for different datasets are shown in Fig. 6. We compared the Root Mean Square Error (RMSE) on this particular transformation and we see that ICP point to plane converges faster than other methods including ours (in 50 iterations). Fig. 7 shows the RMSE comparisons of different approaches on different datasets for this simple transformation. In this example, ICP point to plane seems to converge faster than our method since the rotation and the translation component is very small. The next step is to evaluate how well our method performs as compared to the other methods when the rotation is large.

Fig. 8 shows the results of different approaches including ours when the rotation and the translation component of the transformation matrix is quite large. In this example we use the Bunny Rabbit dataset where the Source is transformed from the Target as:

\((r, p, y, x, y, z) = (-1.32811, -5.87854, 2.12814, -0.874, -0.433, 0.221)\).

Clearly the translation component is small as compared to the rotation component. In this example, our method performs well in comparison to the other methods with RMSE as 4.76074e-06 in our method after 20 iterations. It clearly shows that the Source is aligned very well with our approach where as in other methods the Source failed to align with the Target point cloud. We compared the RSME results with other approaches like GICP, NDT and ICP-nonlinear. Fig. 9 shows the RMSE results for various methods on the Bunny Rabbit dataset. It is clear that our method performs well in comparison to others. In addition, Table 2 shows the RMSE comparison of various methods applied on the Bunny Rabbit dataset with various rotation and translation.

In Fig. 10 we use the Dragon dataset where the Source is transformed from the Target as:

\((r, p, y, x, y, z) = (2.39318, -5.02554, -2.69076, 0.000, -0.003, 0.003)\).

Again, our method performs well in comparison to other methods with RMSE as 8.66664e-08 in our method after 25 iterations. We arrive at a similar conclusion that the Source is aligned very well with our approach where as in other methods the Source failed to align with the Target point cloud. Fig. 11 shows the RMSE results for various methods on the Dragon dataset. In addition, Table 3 shows the RMSE comparison of various methods applied on the Dragon dataset with various rotation and translation.

In Fig. 12 we use the Happy Buddha dataset where the Source is transformed from the Target as \((r, p, y, x, y, z) = (-4.50504, 1.31677, 4.83251, -0.023, -0.019, -0.008)\). Again, our method performs well in comparison to the rest of the methods with RMSE as 2.26675e-06 in our method after 25 iterations. We arrive at a similar conclusion that the Source is aligned very well with our approach where as in other methods the Source failed to align with the Target point cloud. Fig. 13 shows the RMSE results for various methods on Dragon dataset. In addition, Table 4 shows the RMSE comparison of various methods applied on Dragon dataset with various rotation and translation.
Figure 7: Convergence of different Registration Methods (ICP Standard SVD, ICP point to plane, GICP and CoSM ICP) on different datasets on the simple transformation \((r, p, y, x, z) = (\pi/10, 0, 0, 0, 0.05)\). (a) Shows RMSE comparison on Bunny Rabbit dataset. (b) Shows RMSE comparison on the Dragon dataset. (c) RMSE comparison on the Happy Buddha dataset.

Figure 8: RMSE comparison of various methods on the Bunny Rabbit dataset. (b) shows the zoomed in version of (a).

Table 2: RMSE comparison of different methods with different values of rotation and translation after 50 iterations on Bunny Rabbit dataset. Note: Rotation component is in radians.

| Transformation(r, p, y, x, z) | ICP Standard SVD | ICP Point to Plane | GICP | ICP Non-Linear | NDT | CoSM ICP |
|-------------------------------|------------------|--------------------|------|----------------|-----|----------|
| \((-0.866749, -2.6182, -0.318386, -2.1561, -1.25001, -4.8753)\) | 0.000428388 | 88.0856 | 0.0011591 | 0.000206335 | 15.5141 | 1.81086e-13 |
| \((5.08434, -3.9644, -2.66895, 2.45251, -6.82633, 1.41512)\) | 0.000274292 | 268.684 | 81.2352 | 0.000301596 | 81.2352 | 3.09662e-13 |
| \((2.39384, -2.57132, 0.66973, 0.876204, -2.893912, 2.68268)\) | 0.000177495 | 147.987 | 58.9048 | 58.9048 | 1.75634e-13 |
| \((6.10518, -0.249119, 2.41527, 1.99458, 8.99637, 1.20097)\) | 0.000177495 | 147.987 | 58.9048 | 58.9048 | 1.75634e-13 |
| \((1.17438, -9.5203, -4.13622, 4.6532, 2.8659, 0.0542642)\) | 0.000283007 | 859.114 | 28.7628 | 28.7628 | 1.98801e-13 |
| \((-0.866749, -2.6182, -0.318386, -2.1561, -1.25001, -4.8753)\) | 0.000471378 | 58.1505 | 55.996 | 55.996 | 2.63875e-13 |
Figure 9: White Point Cloud: Original Point Cloud (Target). Red Point Cloud: Source point cloud. Green Point Cloud: Source transformed point cloud after applying iterations. Source is transformed from the Target as \((r, p, y, x, y, z) = (-1.32811, -5.87854, 2.12814, -0.874, -0.433, 0.221))\). (a)-(c) shows the convergence of the Source point clouds after 5, 10 and 20 iterations using the standard ICP, respectively (RMSE after 20 iterations is 0.000358474). (d)-(f) shows the same using ICP Point to Plane (RMSE after 20 iterations is 0.00216325). (g)-(i) shows the same using CoSM ICP (RMSE after 20 iterations is 4.76074e-06).
Figure 10: White point cloud: Original Point Cloud (Target). Red point cloud: Source point cloud. Green point cloud: shows the Source transformed point cloud after applying different registration methods. Source is transformed from the Target as \((r, p, y, x, y, z) = (2.39318, -5.02554, -2.69076, 0.001, -0.005, 0.003))\). (a)-(c) shows the convergence of the Source point clouds after 5, 10 and 25 iterations using the standard ICP respectively (RMSE after 25 iterations is 0.000198014). (d)-(f) shows the same using ICP Point to Plane (RMSE after 25 iterations is 0.000136451). (g)-(i) shows the same using CoSM ICP (RMSE after 25 iterations is 8.66664e-08).

For evaluating on the Lidar dataset we applied a voxel grid filtering of leaf size 1.5. The original point cloud size is 115150 points and after voxel grid filtering we get 1497 points. In Fig. 12 we use the Lidar dataset where the Source is transformed from the Target as \((r, p, y, x, y, z) = (-4.27108, -0.505914, 0.0988647, 10.938, -10.532, 17.832))\). Again, our method performs well in comparison to the rest of the methods with RMSE as 2.26675e-06 in our method after 25 iterations. We arrive at a similar conclusion that the Source is aligned very well with our approach where as in other methods the Source failed to align with the Target point cloud. Fig 15 shows the RMSE results for various methods on the Lidar dataset. In addition, Table 5 shows the RMSE comparison of various methods applied on the Lidar dataset with various rotation and translation.

For evaluating on the Lidar dataset we applied a voxel grid filtering of leaf size 1.5. The original point cloud size is 115150 points and after voxel grid filtering we get 1497 points. In Fig. 12 we use the Lidar dataset where the Source is transformed from the Target as \((r, p, y, x, y, z) = (-4.27108, -0.505914, 0.0988647, 10.938, -10.532, 17.832))\). Again, our method performs well in comparison to the rest of the methods with RMSE as 2.26675e-06 in our method after 25 iterations. We arrive at a similar conclusion that the Source is aligned very well with our approach where as in other methods the Source failed to align with the Target point cloud. Fig 15 shows the RMSE results for various methods on the Lidar dataset. In addition, Table 5 shows the RMSE comparison of various methods applied on the Lidar dataset with various rotation and translation.
4.2. Evaluation on datasets with outliers.

In this section we evaluated our approach in cases where the data has outliers. We randomly injected data values at random indexes in the Source point set. The algorithm for injecting random outliers is shown in Algorithm 3.

Algorithm 3 Transform the point cloud with a Random Transformation Matrix.

1: function InjectOutliers(Ps).
2:   ridx = Pick random index in the Source point set.
3:   trx = Random translation along the X-axis.
4:   try = Random translation along the Y-axis.
5:   trz = Random translation along the Z-axis.
6:   Pps[ridx].x = Ps[ridx].x + trx
7:   Pps[ridx].y = Ps[ridx].y + try
8:   Pps[ridx].z = Ps[ridx].z + trz
9:   return Pps.
10: end function

In this experiment we selected the percentage of points in the Source dataset to be affected by outliers. We added the outliers after performing voxel grid filtering of leaf size 0.005 since one can still encounter outliers after pre-

Table 4: RMSE comparison of different methods with different values of rotation and translation (from Source to Target) after 50 iterations on the Happy Buddha dataset.

| Transformation(p, y, z) | ICP Standard SVD | ICP Point to Plane | GICP | ICP Non-Linear | NDT | CoSM ICP |
|-------------------------|------------------|-------------------|-----|----------------|-----|---------|
| (0.2398, -2.1852, -0.5040) | 0.00016037 | 325.136 | 469.843 | 55.3922 | 55.3922 | 1.41924e-13 |
| (-1.5869, 2.6094, -0.5603) | 0.000142378 | 39.1843 | 310.756 | 115.142 | 115.142 | 3.78298e-13 |
| (-2.7911, 4.3194, -2.1571) | 0.000121835 | 26.6913 | 65.4895 | 115.142 | 115.142 | 1.94386e-13 |
| (-0.7441, -1.8619, -1.7826) | 0.000192252 | 27.6866 | 71.9632 | 71.9632 | 71.9632 | 1.80283e-13 |
| (5.1726, 1.5723, 1.9238) | 0.000157366 | 116.57 | 27.6866 | 27.6866 | 27.6866 | 1.38844e-13 |
Figure 12: White point cloud: Original point cloud (Target). Red point cloud: Source point cloud. Green: Source transformed point cloud after applying different registration methods. Source is transformed from the Target as \((r, p, y, x, y, z) = (-4.50504, 1.31677, 4.83251, -0.023, -0.019, -0.008)\). (a)-(c) shows the convergence of the Source point clouds after 5, 10 and 25 iterations using the standard ICP, respectively (RMSE after 25 iterations is 0.000128905). (d)-(f) shows the same using ICP Point to Plane (RMSE after 25 iterations is 0.000124717). (g)-(i) shows the same using CoSM ICP (RMSE after 25 iterations is 2.26675e-06).

processing the data. The Source point cloud is transformed from the Target point cloud with the transformation \((r, p, y, x, y, z) = (-1.32811, -5.87854, 2.12814, -0.874, -0.433, 0.221)\). The Source point cloud is indeed affected by outliers, and we compare the results with different approaches on different datasets when 10%, 25% and 50% of the Source data points are affected.

We compared our approach with other methods as well. Table 6 shows the RMSE comparison with other datasets where 10% of the Source dataset (after voxel grid filtering) is is affected. Clearly, our methods outperforms other approaches under various rotations and translations. Our results are consistent and keeps performing better even when 25% and 50% of the Source dataset are affected as shown in [7] and [8].
Figure 13: RMSE comparison of various methods on the Happy Buddha dataset. (b) shows the zoomed in version of (a).

Table 5: RMSE comparison of different methods with different values of rotation and translation (from Source to Target) after 50 iterations on the KITTI Lidar dataset. Note: Rotation component is in radians

| Transformation (r, p, y, x, y, z) | ICP Standard SVD | ICP Point to Plane | GICP | ICP Non-Linear | NDT | CoSM ICP |
|---------------------------------|-----------------|------------------|------|----------------|-----|----------|
| [-2.36822,-0.72688,2.59612,-0.08886,-0.68882,-1.68381] | 0.000411282 | 711.375 | 114.581 | 0.000260968 | 114.581 | 0.000171509 | 6.158345e-05 |
| [-2.67216,-3.73756,-0.90248,-4.97289,-4.04282,-4.09878] | 0.000257398 | 374.895 | 0.000460177 | 23.8551 | 119.48 | 7.70255e-05 | 8.26112e-05 |
| [-1.56059,4.39878,-2.80996,-3.91466,-1.31099,-3.34176] | 0.000257819 | 304.884 | 37.13 | 0.000478805 | 37.13 | 0.000210153 | 6.492456e-05 |

Table 6: (10% of the Source data contains outliers) RMSE comparison of different methods with different values of rotation and translation (from Source to Target) after 50 iterations on different datasets. Note: Rotation component is in radians

| Transformation (r, p, y, x, y, z) | ICP Standard SVD | ICP Point to Plane | GICP | ICP Non-Linear | NDT | CoSM ICP |
|---------------------------------|-----------------|------------------|------|----------------|-----|----------|
| [-0.17274,-6.52683,2.47671,0.85269,0.31915,0.90184] | 0.00020957 | 76.301 | 128.453 | 0.000246111 | 128.453 | 0.000201795 | 7.942776e-05 |
| [-0.620192,-5.82537,2.70454,3.85906,2.74563,-5.42011] | 0.000270745 | 45.484 | 101.049 | 23.8551 | 101.049 | 7.942776e-05 | 8.03656e-05 |
| [-1.35321,4.87181,-3.45812,-7.74232,5.24125,-6.44992] | 0.00021795 | 24.134 | 125.922 | 0.000223347 | 125.922 | 0.000171509 | 7.67166e-05 |

Table 7: (25% of the Source data contains outliers) RMSE comparison of different methods with different values of rotation and translation (from Source to Target) after 50 iterations on different datasets. Note: Rotation component is in radians

| Transformation (r, p, y, x, y, z) | ICP Standard SVD | ICP Point to Plane | GICP | ICP Non-Linear | NDT | CoSM ICP |
|---------------------------------|-----------------|------------------|------|----------------|-----|----------|
| [-1.20164,-0.24146,3.21611,-2.09067,5.95335,-6.68423] | 0.000259019 | 116.015 | 98.0711 | 0.000258207 | 98.0711 | 0.000194223 | 8.3469e-05 |
| [-1.14973,-4.81181,3.49819,9.95025,0.96451,8.4972] | 0.000266775 | 103.766 | 269.049 | 0.000226339 | 269.049 | 0.000177326 | 7.24041e-05 |
Figure 14: (From KITTI dataset) White point cloud: Original Point Cloud (Target). Red point cloud: Source point cloud and Green point cloud: Source transformed point cloud after applying different registration methods. Source is transformed from the Target as \((r, p, x, y, z) = (-4.27108, -0.505914, 0.0988647, 10.938, -10.532, 17.832)\). (a)-(c) shows the convergence of the Source point clouds after 7, 15 and 33 iterations using the standard ICP, respectively (RMSE after 33 iterations is 13.7695). (d)-(f) shows the same using ICP Point to Plane (RMSE after 33 iterations is 12.6385). (g)-(i) shows the same using CoSM ICP (RMSE after 25 iterations is 1.01689e-08).

4.3. Effect of \(\sigma\).

As shown in the previous section, the registration works well when the value of \(\sigma\) is 100. With lower values of \(\sigma\) like 0.01 it aligned well for small rotation and translation, however for large rotation and translation between the Source and the Target it took a large number of iterations to converge. For even smaller values of \(\sigma\) like 0.001, and for larger rotation and translation, alignment did take place. Moreover, the iterations was larger (\(\sim 500\)), and the computed transformation was prone to errors.
Figure 15: RMSE comparison of various methods on the KITTI Lidar dataset.

Table 8: (50% of the Source data contains outliers) RMSE comparison of different methods with different values of rotation and translation (from Source to Target) after 50 iterations on different datasets. Note: Rotation component is in radians

| Bunny Dataset | Transformation(x,y,z) | ICP Standard SVD | ICP Point to Plane | GICP | ICP Non-Linear | NDT | CoSM ICP |
|---------------|-----------------------|------------------|-------------------|------|----------------|-----|----------|
| Source        | Target                | RMSE             | RMSE              | RMSE | RMSE           | RMSE| RMSE     |
| (-4.21267, -5.1913, 0.510754, 4.78028, -3.81009, -8.99842) | (-1.90501, 2.94602, -4.15732, -4.38121, -8.43926, 0.951326) | 0.000552246 | 176.793 | 118.74 | 0.000705962 | 118.74 | 0.000367226 |
| (-3.86175, 2.38854, -4.81381, 2.95017, -6.77983, -2.26068) | 0.000615907 | 83.665 | 61.5916 | 0.000610313 | 61.5916 | 0.00036124 |
| Dragon Dataset | (-4.41725, 2.40238, 0.642351, 3.74842, -9.32348, -9.14091) | (-3.29757, 5.37519, 4.68352, 5.59739, -1.76298, -7.59656) | (-0.27986, 2.66004, 5.61896, -0.89552, -2.83092, 4.9319) | 0.000487516 | 188.167 | 185.885 | 0.000510906 | 185.885 | 0.000372615 |
| (-3.88617, 5.23854, -4.81381, 2.95017, -6.77983, -2.26068) | 0.000615907 | 83.665 | 61.5916 | 0.000610313 | 61.5916 | 0.00036124 |
| Happy Buddha Dataset | (5.471, 1.1243, -0.67022, 8.42324, -5.29389, -3.70151) | (1.77436, -2.70017, -4.70393, 1.9514, 4.74639, -8.0765) | (-3.01961, 2.91297, -5.51066, 6.01917, 2.99075, 7.78239) | 0.000462191 | 510.054 | 111.689 | 0.000489964 | 111.689 | 0.000384029 |
| Source        | Target                | RMSE             | RMSE              | RMSE | RMSE           | RMSE| RMSE     |
| (-3.86175, 2.38854, -4.81381, 2.95017, -6.77983, -2.26068) | 0.000615907 | 83.665 | 61.5916 | 0.000610313 | 61.5916 | 0.00036124 |
| Happy Buddha Dataset | (5.471, 1.1243, -0.67022, 8.42324, -5.29389, -3.70151) | (1.77436, -2.70017, -4.70393, 1.9514, 4.74639, -8.0765) | (-3.01961, 2.91297, -5.51066, 6.01917, 2.99075, 7.78239) | 0.000462191 | 510.054 | 111.689 | 0.000489964 | 111.689 | 0.000384029 |
Figure 16: CoSM Results when Source is infected with noise. White point cloud: Original Point Cloud (Target). Red point cloud: Infected Source point cloud. Green point cloud: Source transformed point cloud after applying iterations. Transformation from Source to Target: $(r, p, x, y, z) = (1.6585448, 0.6090677, -1.204856, 0.00, -0.003, 0.003))$. (a), (b) and (c) shows the result of CoSM ICP on Bunny dataset when 10%, 25% and 50% of the data is affected with outliers. Their respective RMSE’s are $8.64002e-05$, $0.000203088$ and $0.000384025$. (d), (e) and (f) shows the result of CoSM ICP on the Dragon dataset when 10%, 25% and 50% of the data is affected with outliers. Their respective RMSE’s are $6.91792e-05$, $0.000163833$ and $0.000368516$. (g), (h) and (i) shows the result of CoSM ICP on the Happy Buddha dataset when 10%, 25% and 50% of the data is affected with outliers. Their respective RMSE’s are $9.29047e-05$, $0.000205528$ and $0.000386966$. For each case we performed around 30 iterations.
5. Discussion

We have found that using a Correntropy Relationship matrix proved to be very effective in the problem of registration. Through experiments we have evaluated that our approach is not only robust to large rotation and translation, but also robust to outliers. We have compared our approach to other state of the art methods like GICP, NDT, ICP Non-Linear etc. and proved it’s efficiency under large transformation matrices. It is quite important to note that the size of the Correntropy Matrix is dependent on the size of the dataset, which means that if we have larger datasets, our approach would be computationally very expensive for each iteration (~ 7ms for a dataset of size 10000 points for each iteration). In addition to our previously mentioned results, we have also evaluated on different datasets using multiple runs. We define a run such that in each run, we generate a random transformation matrix and we compute a transformation matrix from randomly generated translation and rotation components and transform the original point cloud (which we call it as Target) to another point cloud (which we call it as Source). We then employ several state of the art methods like GICP, NDT, ICP Non-Linear NDT and CoSM ICP iteratively on the Source point cloud. In this case we perform the iteration 50 times during each run and collect the final RMSE’s from each of the methods. The components of the random transformation matrix (translation and rotations (x, y, z, r, p, y)) are generated independently with different seeds (seeds are defined by the system’s clock). The mean for all the components is zero. The standard deviation for the translation components along x, y, z is 10, where as for the angular components it is 6.28317 radians (360 degrees) along each axes. This guarantees that we could potentially cover all the possible orientation along all axes. On the Bunny dataset we found that ICP Point to Plane had a very large RMSE (mean RMSE for ICP Point to Plane is \(1.9668e + 03\) in 100 runs) which means the Source dataset was not at all aligned to the Target dataset at all. However GICP and NDT better than ICP Point to Plane in most of the cases (mean RMSE of GICP in 100 runs is 75.6727 and the mean RMSE of NDT in 100 runs is 76.2536). However in certain transformations GICP and NDT performs well with RMSE of \(8.8582e - 04\) and \(5.6122e - 04\) respectively. On the other hand if we compare the average RMSE of Standard ICP, ICP Non-linear and CoSM ICP we see that their average RMSE’s in 100 runs is \(2.4156e - 04\), \(2.3781e - 04\) and \(3.2489e - 07\) respectively. It is very evident that CoSM ICP is better than the other approaches (by more than ~ 100%). Fig. 17(a) shows the distribution of errors (RMSE’s) of various methods in 100 runs on the Bunny Rabbit dataset. Fig. 17(b) shows the RMSE deviation of Standard ICP, ICP Non-Linear and CoSM ICP. In the Dragon dataset the average RMSE’s of Standard ICP, ICP Point to Plane, GICP, ICP Non-Linear, NDT and CoSM ICP is \(1.1372e - 04\), \(177.0397\), \(86.8183\), \(1.4154\), \(87.5327\) and \(2.6989e - 13\) respectively. In this case as well CoSM outperformed the other approaches by more than ~ 100%. It is again evident from Fig. 18(a) and Fig. 18(b). On the Happy buddha dataset, the average RMSE’s of Standard ICP, ICP Point to Plane, GICP, ICP Non-Linear, NDT and CoSM ICP is \(9.4830e - 05\), \(414.0633\), \(93.0712\), \(1.1289e - 04\), \(93.3621\) and \(5.6055e - 13\) respectively in 100 runs. Again it is validated that CoSM ICP performs better than the rest of the approaches by more than ~ 100%. Fig. 19(a) and Fig. 19(b) clearly validates our claim for the happy buddha dataset. Readers can note that the top and the bottom of the box plot shown represent the 25th and 75th percentiles respectively and the middle red line shows the median of the error (average RMSE’s) across 100 runs.

It is significant to note that our approach has proved beneficial in solving the registration problem under various rotations and translations as compared to other well-known methods. We encourage the community to validate our approach on their own datasets.
Figure 17: Average RMSE’s of various methods in 100 runs (Bunny Rabbit dataset). Each run consists of random rotation and translation between the *Source* and the *Target*.

Figure 18: Average RMSE’s of various methods in 100 runs (Dragon dataset). Each run consists of random rotation and translation between the *Source* and the *Target*.

Figure 19: Average RMSE’s of different methods in 100 runs (Happy Buddha dataset). Each run consists of random rotation and translation between the *Source* and the *Target*.
6. Conclusions and Future Work

One of the key things we have addressed is to find an accurate estimate of the transformation between the Source and the Target point clouds under various rotations and translations. In this work, we have verified our approach in various datasets, where the Source is transformed from the Target using various randomly generated transformation matrices. Through the obtained results we can see that our method has performed better than the other state of the art approaches. In addition, we also evaluate our approach when the Source dataset is affected by noise generated with random intensity and affecting random data points. In this scenario as well, we have verified that our proposed approach has outperformed most of the other state of the art approaches where the ‘infected’ Source dataset is transformed randomly from the Target dataset. We strongly insist our readers to evaluate our approach located in our Github repository. In addition to our current approach, we also aim to evaluate our approach in SLAM where the poses computed from our approach can be used to localize a robot (both aerial and ground robot) as well as build a map of the environment. For further evaluation, we plan to test our approach in the well-known KITTI SLAM dataset and other online datasets.

Acknowledgement

This work is partially supported by the U.S. National Science Foundation (NSF) under grants NSF-CAREER: 1846513 and NSF-PFI-TT: 1919127, and the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (USDOT/OST-R) under Grant No. 69A3551747126 through INSPIRE University Transportation Center. The views, opinions, findings and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the NSF and USDOT/OST-R.

References

[1] A. Singandhupe, H. M. La, A review of slam techniques and security in autonomous driving, in: 2019 Third IEEE International Conference on Robotic Computing (IRC), 2019, pp. 602–607. doi:10.1109/IRC.2019.00122
[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, T. Reid, J. J. Leonard, Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, IEEE Trans. on Robotics 32 (6) (2016) 1309–1332. doi:10.1109/TRO.2016.2624754
[3] K. S. Arun, T. S. Huang, S. D. Blostein, Least-squares fitting of two 3-d point sets, IEEE Trans. Pattern Anal. Mach. Intell. 9 (5) (1987) 698–700. doi:10.1109/TPAMI.1987.4767965
URL https://doi.org/10.1109/TPAMI.1987.4767965
[4] P. J. Besl, N. D. McKay, A method for registration of 3-d shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (2) (1992) 239–256.
[5] K.-L. Low, Linear least-squares optimization for point-to-plane icp surface registration.
[6] J. Zhang, S. Singh, Visual-landar odometry and mapping: Low drift, robust, and fast, in: IEEE Intern. Conf. on Robotics and Automation (ICRA), Seattle, WA, 2015.
[7] T. Y. Tang, D. J. Yoon, F. Pomerleau, T. D. Barfoot, Learning a Bias Correction for Lidar- only Motion Estimation, 15th Conf. on Computer and Robot Vision (CRV).
[8] K. Tateo, F. Tombari, I. Laina, N. Navab, Cnn-slam: Real-time dense monocular slam with learned depth prediction, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 6565–6574.
[9] A. Sehgal, A. Singandhupe, H. M. La, A. Tavakkoli, S. J. Louis, Lidar-monocular visual odometry with learned depth prediction, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 6565–6574.
[10] X. Zhang, L. Jian, M. Xu, Robust 3d point cloud registration based on bidirectional maximum correntropy criterion, PhoS one 13 (5) (2018) e0197542.
[17] G. Xu, S. Du, D. Cui, S. Zhang, B. Chen, X. Zhang, J. Xue, Y. Gao, Precise point set registration using point-to-plane distance and correntropy for lidar based localization, in: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2018, pp. 734–739.

[18] X. Zhang, L. Jian, M. Xu, Robust 3d point cloud registration based on bidirectional maximum correntropy criterion, PLOS ONE 13 (2018) e0197542. [doi:10.1371/journal.pone.0197542]

[19] G. Xu, S. Du, D. Cui, S. Zhang, B. Chen, X. Zhang, J. Xue, Y. Gao, Precise point set registration using point-to-plane distance and correntropy for lidar based localization, in: 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 734–739. [doi:10.1109/IVS.2018.8500525]

[20] R. Izanloo, S. A. Fakoorian, H. S. Yazdi, D. Simon, Kalman filtering based on the maximum correntropy criterion in the presence of non-gaussian noise., in: CISS, IEEE, 2016, pp. 500–505.

[21] B. Chen, X. Liu, H. Zhao, J. C. Principe, Maximum correntropy kalman filter, Automatica 76 (2017) 70 – 77. [doi:https://doi.org/10.1016/j.automatica.2016.10.004]
URL http://www.sciencedirect.com/science/article/pii/S000510981630396X

[22] J. C. Principe, Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives, 1st Edition, Springer Publishing Company, Incorporated, 2010.

[23] J. Principe, J. Iii, Information-theoretic learning, Advances in unsupervised adaptive filtering.

[24] A. Segal, D. Hähnel, Generalized-icp, 2009. [doi:10.15607/RSS.2009.V.021]

[25] P. Biber, W. Strasser, The normal distributions transform: a new approach to laser scan matching, in: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), Vol. 3, 2003, pp. 2743–2748 vol.3. [doi:10.1109/IROS.2003.1249285]

[26] A. W. Fitzgibbon, Robust registration of 2d and 3d point sets, Image and Vision Computing 21 (13) (2003) 1145–1153, british Machine Vision Computing 2001. [doi:https://doi.org/10.1016/j.imavis.2003.09.004]
URL https://www.sciencedirect.com/science/article/pii/S0262885603001835