Correntropy Iterative Closest Point Algorithm for Registering Point Clouds in Noisy Environment

Ashutosh Singandhupe, Hung Manh La *Senior Member, IEEE*, Trung Dung Ngo *Senior Member, IEEE*, Van Ho

***Abstract*— This work focuses on proposing a new method to register two point cloud datasets (*Source* and *Target* point cloud datasets), which are affected by non-Gaussian noise (shot noises at random locations in the data points). We begin by introducing the idea of Correntropy and its usage in registering two point cloud datasets. After the introduction of Correntropy, we propose the fusion of Correntropy with the Iterative Closest Point (ICP) to create the Correntropy ICP (CICP) algorithm for handling scenarios where the raw point cloud data gets affected by non-Gaussian noise. One of the serious problems in the traditional ICP is that the registration results in poor estimation of transformation (rotation and translation) between the *Source* and *Target* datasets when the *Source* is affected by non-Gaussian noise. Our proposed CICP approach attempts to handle this situation. To demonstrate our approach, we inject non-Gaussian noise to the *Source* dataset, and through comparison with the traditional ICP, we evaluate our CICP approach and show its performance vs. the ICP.**

] you

I. Introduction

Registering two point clouds is a significant component in modern computer vision systems and is an active area of research. 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 as the relative error between the 2 point sets is minimized. The problem of registration can also be a significant component in Simultaneous Localization and Mapping (SLAM) and other visual data matching algorithms [1]. The relative transformation computed from 2 point clouds acquired at close time intervals from a 3D sensor can be used to compute the overall trajectory of a system. SLAM has evolved rapidly over the past decade with numerous algorithms that use data from multiple sensors to estimate the state of a system. Modern SLAM algorithms are mostly based on visual data and geometric data of surrounding environment, which can be extracted from sensors like cameras, Lidar, etc. [1], [2].

Early approaches for 3D point cloud registration can be attributed to Besl and Kay [3], [4] and Arun et.al [3], which involves computing the centroid for both *Source* and *Target* dataset. 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 the 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 [5]. 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 [6]–[9].

Despite the various advances in the registration methodologies, there has been very little research work done concerning the situation where the raw point cloud data is noisy. If the noise is Gaussian, we can remove most of it in the data. However, when the noise is non-Gaussian (e.g. shot noises), it becomes difficult to resolve. So it is important to address the problem of handling non-Gaussian noise. 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.

For handling non-Gaussian noise or shot noise we propose a Coorrentropy Iterative Closest Point (CICP) algorithm. In our approach, we inject the raw data of the *Source* dataset with random false values and through comparison with the traditional ICP, we evaluate how well our approach can estimate the transformation with the *Target* dataset. In situation like this we show how the traditional ICP is bound to fail, leading it to false transformation. 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.

The idea of correntropy has seen wide spread use in modern

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).

Corresponding author: Hung La, email: `hla@unr.edu`

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.

The source code of the CICP is available at the ARA lab's github: `https://github.com/aralab-unr/CICP` machine learning, signal processing and other related fields [10]. Correntropy is a similarity measure between 2 random variables and has been implemented with the traditional Kalman filter [11]. Inspired from this correntropy concept, we fuse it with the ICP algorithm, called correntropy ICP or CICP. Our main contribution to this work can be summarized as:

- Introducing the idea of correntropy, which explains the similarity between 2 random variables.
- Incorporating the idea of correntropy into the ICP algorithm where we explain how correntropy can be used in the ICP algorithm.
- Formulating CICP - where we provide a formulation of our CICP algorithm.
- Introducing random shot nose or non-Gaussian vectors in the raw *source* dataset and through comparison with the traditional ICP, we evaluate how well our CICP algorithm estimates the transformation between the *Source* and the *Target* datasets.

The remaining paper is organized as follows: Section II introduces the idea of correntropy and its underlying concepts. Section III describes our implementation of the proposed methodology. The results and evaluation of our proposed methodology is discussed in Section IV. Conclusions are given in Section V.

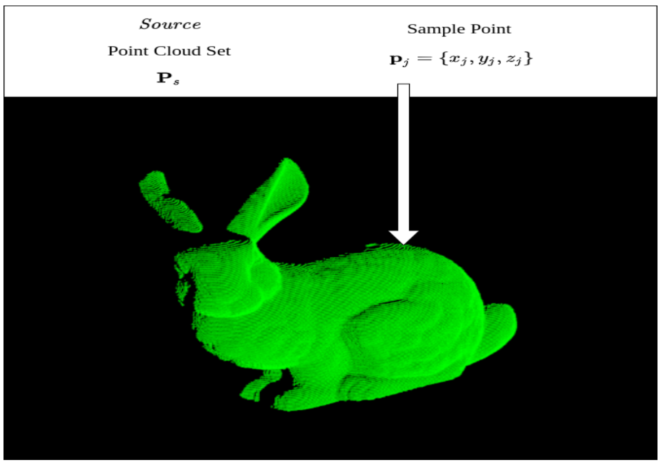
II. Correntropy Criterion

1) Correntropy: We begin by describing the mathematical representation of point cloud data and develop our algorithm based on it. We denote the *source* point cloud as $\mathbf{P}_s$ and the *Target* point cloud as $\mathbf{P}_t$. The points in the point cloud $\mathbf{P}_s$ contains N points in which each point can be referenced as $\mathbf{p}_j$, where $j = 1, ..., N$ and $\mathbf{p}_j = \{x_j, y_j, z_j\}$, where x_j, y_j, z_j denotes the 3D coordinates of the point $\mathbf{p}_j$. Similarly, the points in the point cloud $\mathbf{P}_t$ contains N points in which each point can be represented as $\mathbf{q}_k$ where $k = 1, ..., N$ and $\mathbf{q}_k = \{x_k, y_k, z_k\}$, where x_k, y_k, z_k denotes the 3D coordinates of the point $\mathbf{q}_k$. Figure 1 shows a brief description of the point clouds and Table I denotes the mathematical notations used throughout this work.

We use the correntropy concept between the *Source* point cloud and the *Target* point cloud. Again, the primary purpose is to highlight the significance of correntropy ICP as compared to ICP when the raw *Source* point cloud data gets affected by non-Gaussian noise. Correntropy has proved beneficial to remove large outliers [10]. Correntropy is essentially a similarity measure of two random variables. We use the correntropy concept between the *Source* and the *Target* point clouds. In this scenario, we are measuring the similarity between 2 points $\mathbf{p}_j$ and $\mathbf{q}_k$ from the point clouds $\mathbf{P}_s$ and $\mathbf{P}_t$, respectively. Then the correntropy criterion between the 2 points $\mathbf{p}_j$ and $\mathbf{q}_k$ can be mathematically represented as :

$$V_\sigma(\mathbf{p}_j, \mathbf{q}_k) = E[\kappa_\sigma(\mathbf{p}_j - \mathbf{q}_k)]. \tag{1}$$

(a)



(b)

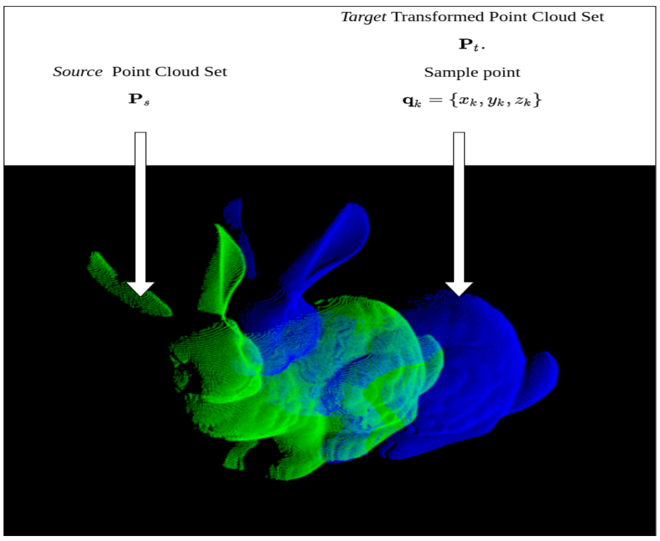

Target Transformed Point Cloud Set
$\mathbf{P}_t$.
Source Point Cloud Set
$\mathbf{P}_s$
Sample point
$\mathbf{q}_k = \{x_k, y_k, z_k\}$

Fig. 1: (a) *Source* point cloud $\mathbf{P}_s$ (bunny rabbit). $\mathbf{P}_s$ contains N points $\mathbf{p}_j$, each of which is a 3D representation x_j, y_j, z_j. (b) Rotated and translated (Transformed) *Target* point cloud $\mathbf{P}_t$. $\mathbf{P}_t$ contains N points $\mathbf{q_k}$, each of which is a 3D representation x_k, y_k, z_k. (Green is the *Source* point cloud and blue is the transformed point cloud or *Target* point cloud).

Equ. (1) also refers to in common literature as a cross-correntropy of two random variables [11]–[13]. In Equ. (1), *E[.]* refers to the expectation of the variable, and κ_σ denotes the kernel function. We can manually choose the size of the kernel function (also referred as bandwidth σ). In our approach, we use the Gaussian kernel function where we define the correntropy function as:

$$V_\sigma(\mathbf{p}_j, \mathbf{q}_k) = \frac{1}{N}\sum_{i=1}^{N} G_\sigma(\mathbf{p}_j - \mathbf{q}_k), \tag{2}$$

TABLE I: Variables and it's usages

Variables	Usages
$\mathbf{P}_s$	*Source* Point Cloud Set.
$\mathbf{P}_t$	*Target* Point Cloud Set.
N	Total number of points in point clouds $\mathbf{P}_s$ and $\mathbf{P}_t$.
j, k	Index of every point in point cloud where $j, k = 1, .., N$.
$\mathbf{p}_j$	Individual 3D point (x_j, y_j, z_j) in a point cloud $\mathbf{P}_s$, or $\mathbf{p}_j \in \mathbf{P}_s$.
$\mathbf{q}_k$	Individual 3D point (x_k, y_k, z_k) in a point cloud $\mathbf{P}_t$, or $\mathbf{q}_k \in \mathbf{P}_t$.
G_σ	Gaussian kernel function with bandwidth σ.
$\mathbf{R}$	Rotation matrix.
$\mathbf{tr}$	Translation vector.
ε^2	Root mean square error.
s	Scaling factor.
$\mathbf{c}_n(u)$	Corresponding points in the *Source* dataset.
$\mathbf{d}_n(v)$	Corresponding points in the *Target* dataset.

From Equ.(2),

$$G_\sigma(\mathbf{p}_j - \mathbf{q}_k) = exp(-\frac{\|\mathbf{p}_j - \mathbf{q}_k\|^2}{2\sigma^2}), \tag{3}$$

and σ is the bandwidth or the kernel size of the Gaussian kernel. From Equ. (3) it becomes evident that if $\mathbf{p}_j = \mathbf{q}_k$ Gaussian correntropy is maximum, and the Gaussian correntropy function is positive and bounded [11]–[13].

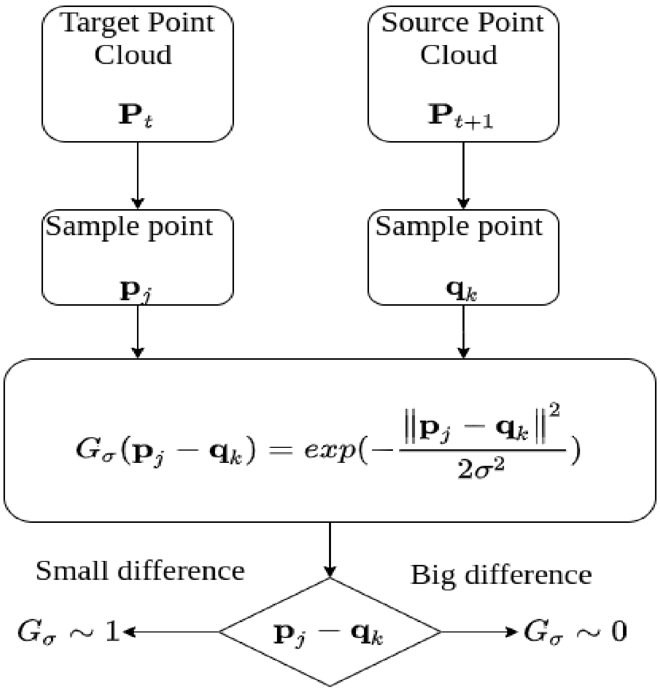


Fig. 2: Correntropy Criterion. Here, when the difference between the points $(\mathbf{p}_j - \mathbf{q}_k)$ is small, i..e., when they are similar, and the Gaussian kernel function G_σ approaches to 1. On the contrary, when the difference is large, the Gaussian kernel function approaches to 0.

Fig. 2 shows the basic understanding of correntropy criterion. We can see that when the difference between the points $(\mathbf{p}_j - \mathbf{q}_k)$ is small, i.e., they are similar, and the Gaussian kernel function G_σ approaches to 1. On the contrary, when the difference is large, the Gaussian kernel function approaches 0. This essentially means that when the data point $\mathbf{q}_k$, which could be a potentially changed data point, is far varied from $\mathbf{p}_j$ then the Gaussian kernel signifies that the points $\mathbf{p}_j$ and $\mathbf{q}_k$ are not similar. Intuitively, this similarity is dependent on the value of σ. This technique allows us to reject the faulty data point (shot noise or non-Gaussian noise) $\mathbf{q}_k$. This idea of calculating the maximum similarity as given in Equ. (3) is called the maximum correntropy criterion.

When the value of $\mathbf{q}_k$ approaches ∞ then the kernel function approaches 0, i.e.,

$$\lim_{\mathbf{q}_k \to \infty} G_\sigma = 0. \tag{4}$$

Intuitively, when the Gaussian Kernel shrinks to 0, the correntropy approaches the value $p(\mathbf{p}_j = \mathbf{q}_k)$ [14].

III. Proposed CICP Method

1) Correntropy 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. By exploring the idea of correntropy in relation to the ICP algorithm, we see how the correntropy criterion can be used to handle non-Gaussian noise present in a raw point cloud dataset. Traditional ICP [3] describes the alignment of 2 point clouds so that the Mean Square Error (MSE) between the 2 point sets is minimized. The MSE is the key criterion in ICP and its variants. Traditionally, the idea of ICP revolves around the problem of aligning 2 points sets $\mathbf{P}_s$ and $\mathbf{P}_t$. We denote the 2 point sets as $\mathbf{P}_s = \{[\mathbf{p}_j]_{j=1}^N\}$ (here $\{\}$ denotes a point set) and $\mathbf{P}_t = \{[\mathbf{q}_k]_{k=1}^N\}$, respectively. Our goal is to best align the *Source* point set $\mathbf{P}_s$ to the *Target* point set (or a model point set) $\mathbf{P}_t$. 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 $(\mathbf{tr})$ and rotated $(\mathbf{R})$ such that *Source* point set $\mathbf{P}_s$ is best aligned with $\mathbf{P}_t$ given that it satisfies a particular criterion (Root Mean Square Error

(RMSE) threshold. This process is commonly known as the registration. So, the registration between the *Source* $\mathbf{P}_s$ and the *Target* $\mathbf{P}_t$ is defined as finding the translation $\mathbf{tr}$ and the rotation $\mathbf{R}$ component such that $\mathbf{P}_s$ is in best alignment with $\mathbf{P}_t$. In our scenario we introduce shot noises or non-Gaussian noise in the *Source* point set $\mathbf{P}_s$ to illustrate the importance of correntropy. A random noise vector $\mathbf{a} = [a_1, a_2, a_3, ..., a_l]$ is introduced in the *Source* dataset $\mathbf{P}_s$, where l is the number of data points in the point cloud dataset we want to change (alter their orignal raw values). $\mathbf{a}$ is a collection of random values generated from normal distribution. We then select random data points in the *Source* dataset $\mathbf{P}_s$ and add the values of $\mathbf{a}$ to the original *Source* data points. In essence, if we select randomly k data points then the resulting data points in the point cloud $\mathbf{P}_s$ becomes $\mathbf{P}_s = \mathbf{P}_s + \rho\mathbf{a}$. Now, if $\rho = 0$, no noise is introduced and if $\rho = 1$, noise is introduced. For the rest of the mathematical formulation we assume that the *Source* point set $\mathbf{P}_s$ is 'infected' with noise vector $\mathbf{a}$, i.e., $\mathbf{P}_s = \mathbf{P}_s + \rho\mathbf{a}$ where $\rho = 1$. Now, in relation to correntropy, the problem can be equivalently formulated as:

$$\underset{s,\mathbf{R},\mathbf{tr}}{max} \sum_{j,k=1}^{N} exp(-\left\|\mathbf{q}_k - (s\mathbf{R}\mathbf{p}_j + \mathbf{tr})\right\|^2 / 2\sigma^2), \quad (5)$$

where σ is the bandwidth of the given Gaussian kernel, which can be user defined and s is the scaling factor which we assume it to be 1 in our case. We iteratively perform this procedure in order to minimize the error between the 2 point clouds. For every iteration n in CICP, firstly, we use the nearest neighbour search to compute the closest point in one set to every other point in other set. Let $\mathbf{c}_n(u)$ be the set of point in the *Source* point set which is closest to the *Target* point set. Similarly, let $\mathbf{d}_n(v)$ be the set of points in the *Target* point set, which is closest to the *Source* point set. Both the vectors $\mathbf{c}_n(u)$ and $\mathbf{d}_n(v)$ can be computed as follows:

$$\mathbf{c}_n(u) = \underset{u,v \in \{1,2,...,N\}}{min} (\mathbf{R}_{n-1}\mathbf{p}_u + \mathbf{tr}_{n-1} - \mathbf{q}_v) \quad (6)$$

$$\mathbf{d}_n(v) = \underset{u,v \in \{1,2,...,N\}}{min} (\mathbf{R}_{n-1}\mathbf{q}_v + \mathbf{tr}_{n-1} - \mathbf{p}_u). \quad (7)$$

Intuitively, $\mathbf{c}_n(u)$ and $\mathbf{d}_n(v)$ represent the point correspondences in both the point dataset, respectively, and let the total number of corresponding data points in both the datasets be $\tilde{N}$. min returns the relative minimum distance between the vector points. Incorporating correntropy, the rotation and the translation components can be computed as,

$$F(\mathbf{R}, \mathbf{tr}) = (1/\tilde{N}) \sum_{u,v=1}^{\tilde{N}} exp(\frac{-\left\|\mathbf{d}_n(v) - (s\mathbf{R}\mathbf{c}_n(u) + \mathbf{tr})\right\|^2}{2\sigma^2}), \quad (8)$$

Taking spatial derivative along $\mathbf{tr}$ and equating it to 0 we can calculate the translation vector $\mathbf{tr}$ as,

$$\frac{\partial F(\mathbf{R}, \mathbf{tr})}{\partial \mathbf{tr}} = 0. \quad (9)$$

Let $X = \frac{-\|\mathbf{d}_n(v) - (s\mathbf{R}\mathbf{c}_n(u) + \mathbf{tr})\|^2}{2\sigma^2}$, then from Equ. (9) we have:

$$\frac{1}{\tilde{N}\sigma^2} \sum_{u,v=1}^{\tilde{N}} exp(X)(\mathbf{d}_n(v) - (s\mathbf{R}\mathbf{c}_n(u) + \mathbf{tr})) = 0. \quad (10)$$

Then from Equ. (10) we can find $\mathbf{tr}$ as

$$\mathbf{tr}^* = \frac{1}{N} \sum_{u,v=1}^{\tilde{N}} (\mathbf{d}_n(v) - s\mathbf{R}\mathbf{c}_n(u)) \quad (11)$$

Now, the centroid for each point set can be computed as,

$$\mathbf{p}_{cen_n} \sim \mathbf{c}_n(u) - \frac{\sum_{u,v=1}^{\tilde{N}} G_\sigma(\mathbf{e}_n)\mathbf{c}_n(u)}{\sum_{u,v=1}^{\tilde{N}} G_\sigma(\mathbf{e}_n)}, \quad (12)$$

$$\mathbf{q}_{cen_n} \sim \mathbf{d}_n(v) - \frac{\sum_{u,v=1}^{\tilde{N}} G_\sigma(\mathbf{e}_n)\mathbf{d}_n(v)}{\sum_{u,v=1}^{\tilde{N}} G_\sigma(\mathbf{e}_n)}, \quad (13)$$

where $\mathbf{e}_n = \mathbf{d}_n(v) - s\mathbf{R}\mathbf{c}_n(u)$. Now the rotation matrix, $\mathbf{R}$, can be calculated by computing the singular value decomposition (SVD) of a matrix $\mathbf{H}$ where,

$$\mathbf{H} = \sum_{u,v=1}^{\tilde{N}} \mathbf{p}_{cen_n} G_\sigma(\mathbf{e}_n) \mathbf{q}'_{cen_n}. \quad (14)$$

where $'$ is the transpose, and it is very similar to the well known algorithm given by [3]. Computing the SVD of $\mathbf{H}$ we get 3 components: $\mathbf{V}$,$\mathbf{U}$ and $\mathbf{D}$, where $\mathbf{V}$ is a $m \times m$ real or complex matrix, $\mathbf{D}$ is a $m \times n$ diagonal matrix, and $\mathbf{U}$ is an $n \times n$ real or complex matrix [3]. Now, $\mathbf{R} = \mathbf{U}\mathbf{V}'$. If the determinant of $\mathbf{R} = 1$ then $\mathbf{R} = \mathbf{U}\mathbf{V}'$ else it is an invalid rotation. Then $\mathbf{tr}$ can be calculated from Equ. 11. This process is iterated and the rotation and the translation is updated until the error ε^2 reaches a certain user defined threshold. The Rotation $\mathbf{R}$ and the translation $\mathbf{tr}$, forms the transformation matrix $\mathbf{M}_t$.

The CICP algorithm can be described in steps as mentioned in Algorithm 1.

IV. Results

Through simple datasets [15] we demonstrate how well the CICP algorithm performs better than the ICP algorithm in case the source dataset $\mathbf{P}_s$ is corrupted by non-Gaussian noise. We also see how the different values of σ in the Gaussian kernel can affect the CICP registration results. In this experiment we use two sample point clouds: a bunny rabbit (Fig. 3-a) and a buddha (Fig. 4-a).

We compare both the ICP and the CICP registration results on the bunny rabbit point clouds, and we see that ICP does not improve the registration results (improving RMSE) in the presence of noise vectors even after multiple iterations (it appears to give a constant RMSE after multiple iterations).

We then evaluate the CICP registration results (RMSE) under different values of σ. Based on the nature of noise, we can tune the value of σ to give better registration results. In Table

Algorithm 1: CICP Algorithm

```
1:  *Source* point cloud ($\mathbf{P}_s$) $\rightarrow \{\mathbf{p}_j\}_{j=1}^N$
2:  *Target* Point Cloud ($\mathbf{P}_t$) $\rightarrow \{\mathbf{q}_k\}_{k=1}^N$
3:  Initialize $\mathbf{R}_0 = \mathbf{I}$ for $k = 0$.
4:  **for** $n = 1, 2, ....K$ **do**
5:      Compute the corresponding points $\mathbf{c}_n(u)$ and $\mathbf{d}_n(v)$ from Equ. (6) and Equ. (7).
6:      Compute $\mathbf{e}_n = \mathbf{d}_n(v) - s\mathbf{R}\mathbf{c}_n(u)$ to be used in the Gaussian kernel.
7:      Compute centroid $\mathbf{p}_{cen_n}$ and $\mathbf{q}_{cen_n}$ from Equ. (12) and Equ. (13), respectively .
8:      Compute the rotation matrix $\mathbf{R}$ from Equ. (14).
9:      Compute translation parameter $\mathbf{tr}^*$ from Equ. (11)
10:     Compute the Mean Square Error (MSE) $\varepsilon^2$.
11:     Create empty transformation matrix $\mathbf{M}$, which is composed of the rotation $\mathbf{R}$ and the translation vector $\mathbf{tr}^*$.
12: **end for**
13: Return pairwise transformations $\mathbf{M}$ calculated by the CICP algorithm.
```

II we evaluated multiple values of and compared the results. In this case we use the bunny point cloud example where 6% of the data points are affected with noise. In CICP we can tune the value of σ to get variant RMSE. In Table III we generate multiple noise vectors on the bunny dataset as mentioned in Section III. Noise vector 1 affects 18% of the data, Noise vector 2 affects 15% of the data, Noise vector 3 affects 11% of the data, and Noise vector 4 affects 9% of the data. For CICP we used a constant σ value of 0.05. From Table III it is evident that CICP algorithm performs better than the traditional ICP algorithm in presence of non-Gaussian noise.

TABLE II: RMSE comparison of CICP with different values of sigma under the presence of non-Gaussian noise for bunny dataset.

Sigma (σ)	RMSE	iterations
2	0.00516	19
1	0.0037	23
0.1	0.0039	22
0.01	0.034	38

TABLE III: RMSE comparison of ICP under the presence of non-Gaussian noise for bunny dataset.

Noise vectors	RMSE (ICP)	RMSE (CICP)
Noise vector 1	0.0667	0.00913
Noise vector 2	0.0429	0.00851
Noise vector 3	0.0390	0.00801
Noise vector 4	0.0234	0.00768

On the buddha point cloud dataset (Fig. 4) we perform CICP on the noisy *Source* dataset. In this case we set the value of σ to be 0.01. We introduce attack vectors, which affects 6% of the *Source* point cloud data. We compare both the ICP and the CICP registration results, and we see that ICP does not improve the registration results (improving RMSE) in the presence of noise vectors even after multiple iterations.

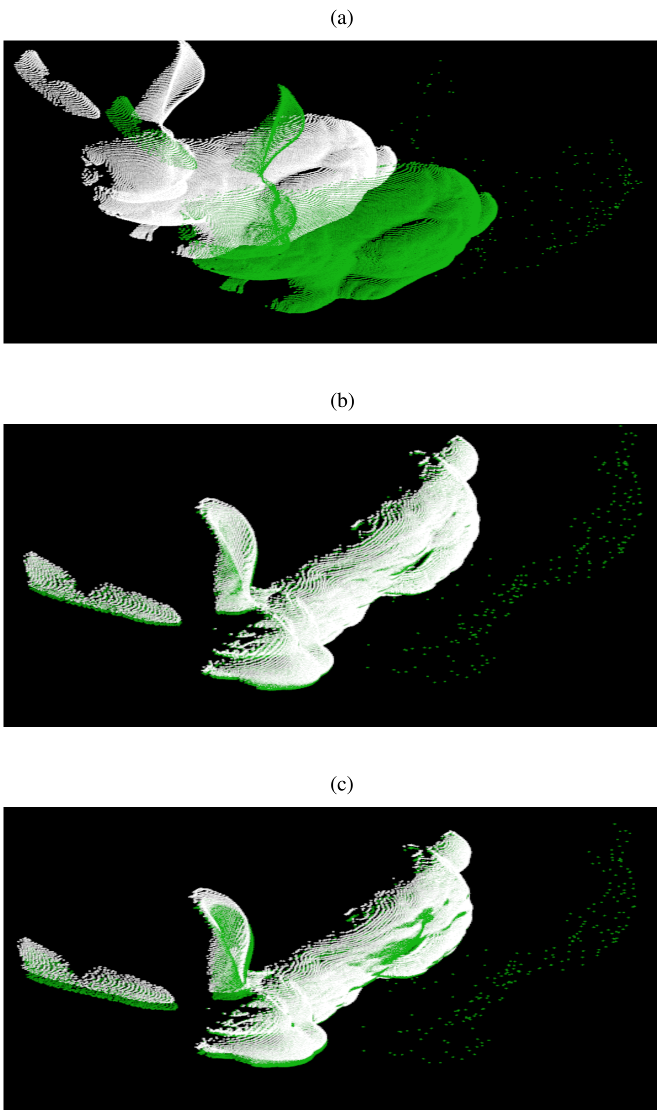
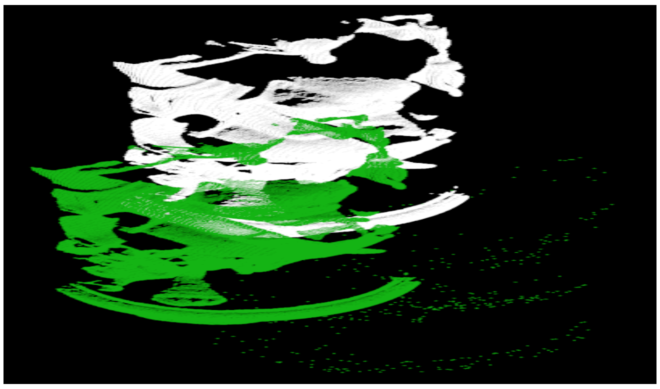


Fig. 3: (a) Rotated and translated point cloud (white is the original point cloud(*Target*), and green is the transformed point cloud (*Source*)) with zero rotation and translation of 0.04 unit in the z-axis). *Source* is injected with noise vectors at random data points (6% of the data points are affected.) (b) CICP registration result with RMSE of 0.00516. (c) ICP registration result with RMSE of 0.0821

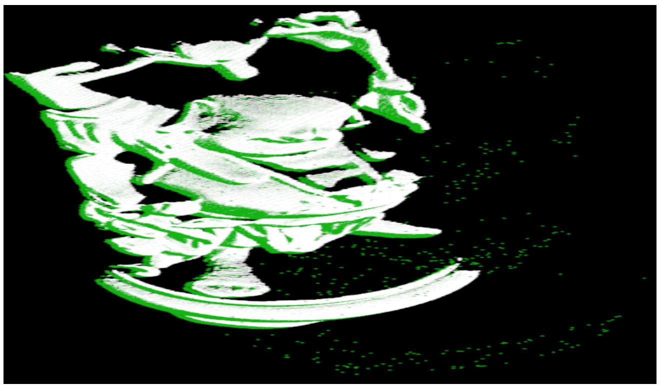
V. Conclusions

In this work, we have proposed a novel Correntropy Iterative Closest Point (CICP) algorithm. We have shown how the proposed CICP performs better than the traditional ICP in response to noise vectors. To evaluate our approach we use

(a)



(b)



(c)

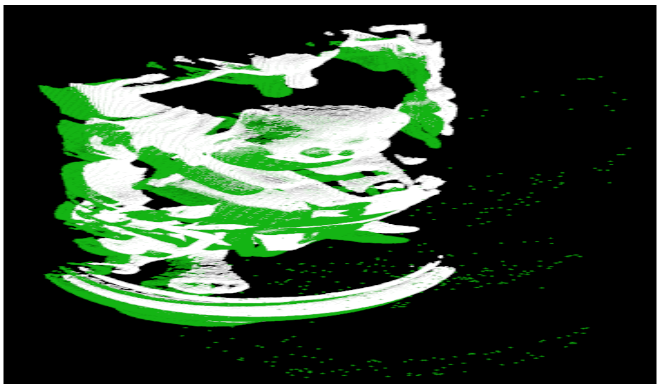


Fig. 4: (a) Rotated and translated point cloud (white is the original point cloud (*Target*), and green is the transformed point cloud (*Source*) with zero rotation and translation of 0.07 unit in the z-axis). *Source* is injected with noise vectors at random data points (6% of the data points are affected.) (b) CICP registration result with RMSE of 0.000412. (c) ICP registration result with RMSE of 0.000793

two point cloud datasets and, we introduce shot noise vectors into the *Source* datasets. We then evaluate the transformation matrix of both ICP and CICP algorithms and through results (RMSE comparision) we highlight the advantages of using Correntropy with different values of σ. Our future work will be focusing on using raw Lidar data points and introducing noise into them. We plan to use the idea of Correntropy to find the similarity between the Lidar point clouds acquired over a period of time and then computing the relative transformation between the point clouds.

References

[1] A. Singandhupe and H. M. La, "A review of slam techniques and security in autonomous driving," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, Feb 2019, pp. 602–607.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec 2016.

[3] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 5, p. 698–700, May 1987. [Online]. Available: https://doi.org/10.1109/TPAMI.1987.4767965

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

[5] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," 01 2004.

[6] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low drift, robust, and fast," in *IEEE Intern. Conf. on Robotics and Automation (ICRA)*, Seattle, WA, May 2015.

[7] T. Y. Tang, D. J. Yoon, F. Pomerleau, and T. D. Barfoot, "Learning a Bias Correction for Lidar- only Motion Estimation," *15th Conf. on Computer and Robot Vision (CRV)*, 2018.

[8] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: real-time dense monocular SLAM with learned depth prediction," *CoRR*, vol. abs/1704.03489, 2017. [Online]. Available: http://arxiv.org/abs/1704.03489

[9] A. Sehgal, A. Singandhupe, H. M. La, A. Tavakkoli, and S. J. Louis, "Lidar-monocular visual odometry with genetic algorithm for parameter optimization," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, D. Ushizima, S. Chai, S. Sueda, X. Lin, A. Lu, D. Thalmann, C. Wang, and P. Xu, Eds. Cham: Springer International Publishing, 2019, pp. 358–370.

[10] A. Singandhupe and H. M. La, "Mcc-ekf for autonomous car security," *Proceedings of the 4th IEEE International Conference on Robotic Computing (IRC)*, Nov 2020.

[11] S. Fakoorian, A. Mohammadi, V. Azimi, and D. Simon, "Robust Kalman-Type Filter for Non-Gaussian Noise: Performance Analysis With Unknown Noise Covariances," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 9, 05 2019. [Online]. Available: https://doi.org/10.1115/1.4043054

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

[13] B. Chen, X. Liu, H. Zhao, and J. C. Principe, "Maximum correntropy kalman filter," *Automatica*, vol. 76, pp. 70 – 77, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000510981630396X

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

[15] Stanford, "The stanford 3d scanning repository," http://graphics.stanford.edu/data/3Dscanrep/, 2020.