Accurate Point Cloud Registration with Robust Optimal Transport

Samuel Bensoussan
samuel.bensoussan@ens-paris-saclay.fr
ENS Paris-Saclay MVA
France

Benjamin Maurel
benjamin.maurel@ensae.fr
ENS Paris-Saclay MVA
France

ABSTRACT

As part of the Geometric Data Analysis course during our MVA curriculum, we performed an extensive and critical analysis of the paper Accurate Point Cloud Registration with Robust Optimal Transport [Shen et al. 2021]. This critique will be divided into three parts:

(1) Context: What is the problem? What are state of the art solutions?
(2) Main Content: What does the authors propose in this article? What is different from previous methods?
(3) Limitations: What can be improved? What are the issues of this new method?

1 CONTEXT

1.1 Introduction to the problem

In computer vision, shape registration refers to the process of aligning two or more shapes in order to compare or analyze them. This is often used in applications such as medical imaging, where it is necessary to compare shapes in order to identify differences or abnormalities. This paper offers a new technique to solve this problem using optimal transport.

One common problem in shape registration is the lack of a consistent frame of reference between the shapes. For example, two different 3D models of the same object may be rotated or translated relative to each other, making it difficult to directly compare or combine them.

To solve this problem, shape registration algorithms typically involve finding a transformation that aligns the shapes in a consistent frame of reference. This can be done through various techniques, such as point matching, feature matching, or optimization techniques that minimize the distance between corresponding points on the shapes. To that extent, optimal transport seems relevant since it provides a generalisation of the distance between points.

Shape registration can also be challenging due to variations in the shape itself, such as deformations or differences in resolution or scale. Dealing with these variations often requires the use of robust techniques that can handle such differences and still achieve a good alignment. This paper also tackle the robust aspect of point cloud registration with a fine resolution regulation.

1.2 An overview of Algorithmic techniques for this problem: focus on ICP

Before we delve into the main content of the article, it's important to review the current state of the art in addressing this type of problem.

The current state of the art in shape registration involves the use of various algorithms and techniques, including:

(1) Iterative Closest Point (ICP) algorithm: This is a widely used method for aligning two point clouds, which are sets of points in 3D space. ICP iteratively adjusts the alignment between the point clouds by minimizing the distance between corresponding points. Coherent Point Drift (CPD) is a variant of this algorithm that is widely used today.
(2) Feature-based methods: These methods rely on identifying and matching distinctive features in the shapes, such as points, lines, or curves, to align the shapes.
(3) Deep learning-based methods: These methods use deep neural networks to learn features and relationships in the shapes, and then use this information to align the shapes.
(4) Any combination of the previous methods

1.3 Focus on ICP and CPD

The ICP algorithm works by iteratively minimizing the distance between corresponding points in the two point clouds. Specifically, it starts with an initial alignment between the point clouds, and then iteratively adjusts the transformation to minimize the distance between corresponding points. This process is repeated until convergence, at which point the algorithm outputs the final transformation that aligns the point clouds.

Mathematically, the ICP algorithm can be described as follows [Arun et al. 1987]:

Given two weighted point clouds $\mathbf{A}$ and $\mathbf{B}$, with N points each, represented as matrices $\mathbf{A} \in \mathbb{R}^{N \times D}$ and $\mathbf{B} \in \mathbb{R}^{N \times D}$, where D is the dimensionality of the points (e.g., D=3 for 3D points).$A = (\alpha_i, a_i)_{i \in 1,..,N}$, $B = (\beta_j, b_j)_{j \in 1,..,N}$, $a_i, b_i \in \mathbb{R}^D$, $\alpha_i, \beta_i \in \mathbb{R}$. The goal is to find a transformation $\mathbf{T}$ that aligns the two point clouds thanks to a rotation and a translation, such that $\mathbf{B} = \mathbf{TA}$.

The ICP algorithm iteratively updates the transformation $\mathbf{T}$ by minimizing the distance between corresponding points in the two point clouds, using the following steps:

(1) For each point $\mathbf{a}_i \in \mathbf{A}$, find the nearest neighbor $\mathbf{b}_i \in \mathbf{B}$ using some distance metric (e.g., Euclidean distance).
(2) Estimate the transformation $\mathbf{T}$ based on the correspondences between the points in $\mathbf{A}$ and $\mathbf{B}$. This can be done using the Singular Value Decomposition as described below.
(3) Transform the point cloud B using the estimated transformation $\mathbf{T}$: $\mathbf{B'} = \mathbf{TB}$.
(4) Repeat steps 1-3 until convergence, at which point the final transformation $\mathbf{T}$ is output.

To compute the transformation $\mathbf{T}$ in step (2), it is common to use the following algorithm:

(1) Compute A_c and B_c the centered version of A and B by subtracting the centroid of each shape:

$$A_c = A - \sum_{i=1}^{N} \alpha_i * a_i$$

$$B_c = B - \sum_{j=1}^{M} \beta_j * b_j$$

(2) Compute the DxD matrix H:

$$H = A_c^T B_c$$

(3) Compute the Singular Value Decomposition of H:

$$H = USV^t$$

(4) Compute the rotation matrix:

$$R = VU^t$$

(5) Compute the translation :

$$T = \sum_{i=1}^{N} \alpha_i * a_i - R \sum_{j=1}^{M} \beta_j * b_j$$

Overall, the ICP algorithm is a widely used method for aligning point clouds, and it can be quite effective in certain cases, particularly when the point clouds are relatively well-aligned and have a large number of corresponding points. However, it can be sensitive to initialization and may not always converge to the optimal solution. The accuracy of shape registration depends on the quality of the input shapes and the chosen registration technique.

One key difference between this approach and optimal transport is the use of nearest neighbor instead of the bijectivity constraint. In the next section, we will delve into the implications of this difference and how it contributes to the benefits of optimal transport.

The Coherent Point Drift (CPD) method is a variant of the Iterative Closest Point (ICP) algorithm that adds regularization to the optimization process. Instead of minimizing point-wise distances between the two point clouds, CPD minimizes the Kullback-Leibler divergence, which measures the difference in shape and distribution between the point clouds. CPD can be useful in scenarios where the ICP algorithm may produce suboptimal results due to the presence of noise or outliers in the data.

Formally, the CPD algorithm can be described as follows:

Given two point clouds A and B, with N points each, represented as matrices $\mathbf{A} \in \mathbb{R}^{N \times D}$ and $\mathbf{B} \in \mathbb{R}^{N \times D}$, where D is the dimensionality of the points (e.g., D=3 for 3D points). The goal is to find a transformation $\mathbf{T}$ that aligns the two point clouds, such that $\mathbf{B'} = \mathbf{TA}$ is the transformed point cloud B.

The CPD algorithm minimizes the Kullback-Leibler divergence between the two point clouds, using the following steps:

(1) Initialize the transformation $\mathbf{T}$ with some initial guess (e.g., the identity transformation).
(2) Compute the transformed point cloud B': $\mathbf{B'} = \mathbf{TA}$
(3) Compute the Kullback-Leibler divergence between the transformed point cloud B' and the original point cloud B:

$$D_{KL}(\mathbf{B}, \mathbf{B'}) = \sum_{i=1}^{N} \mathbf{B}_i \log \frac{\mathbf{B}_i}{\mathbf{B'}_i}$$

(4) Compute the gradient of the Kullback-Leibler divergence with respect to the transformation $\mathbf{T}$:

$$\frac{\partial D_{KL}}{\partial \mathbf{T}} = \sum_{i=1}^{N} \left(\mathbf{B}_i - \mathbf{B'}_i\right) \frac{\partial \mathbf{B'}_i}{\partial \mathbf{T}}$$

(5) Update the transformation $\mathbf{T}$ using gradient descent:

$$\mathbf{T} = \mathbf{T} - \alpha \frac{\partial D_{KL}}{\partial \mathbf{T}}$$

where α is the learning rate.

It is often useful to compare and contrast different algorithms in order to understand their similarities and differences, and to determine the most suitable approach for a given problem. By describing the Coherent Point Drift (CPD) and Iterative Closest Point (ICP) algorithms in relation to the optimal transport method discussed in the original paper, we want to provide readers with a more thorough understanding of the pros and cons of each approach and how they may be applied in practice.

Although it will not be developed in the rest of this report, we also wanted to mention the importance of feature extractions in the shape registration problem. This method greatly increases the performance of matching methods. Features extractions may include points, lines, curves, edges, corners, or other geometric or topological structures that are distinctive and can be used to distinguish one shape from another.

Feature extraction can be performed using various techniques such as edge detection, corner detection, or curve fitting, and the choice of technique depends on the characteristics of the shapes and the desired level of precision and accuracy in the alignment process. Today, this is mainly done by points neural networks such as PointNet [Ruizhongtai Qi et al. 2016] [Qi et al. 2017] cited in the article. Once the features have been extracted, they can be used to compute a similarity measure between the shapes and guide the alignment process. This features can be added to the input dimension to make the matching more accurate. That is why, even if we are in the case of 3D points clouds, in the algorithm we denote as D the dimension of a point: we no longer optimize in 3D but for a dimension D which can be much larger than 3.

Today, all the methods that achieve state of the art performance use points neural networks such as PointNet at least to extract features.

2 MAIN CONTENT

2.1 Reminder

This article decided to focus on the matching feature part, more precisely on methods derived from Optimal Transport (OT) to improve the quality of matching between two shapes.

Shape matching refers to the problem of aligning two point clouds in order to compare their shapes. There are two main goals for this problem:

(1) Pixel-perfect accuracy: In this case, the goal is to find a transformation T that aligns point cloud A with point cloud B with high precision, without any constraints on the form of the transformation T.
(2) Constrained transformation: In this case, the goal is to find a transformation T that aligns point cloud A with point cloud B within a certain class of functions. This can be useful when we have prior knowledge about the form of the transformation or when we want to restrict the solution to a particular type of transformation.

The first goal, which is highlighted in the article with the lung example., is typically pursued by the Raw RobOT algorithm, while the second goal can be addressed using algorithms such as Rigid-RobOT, Spline-RobOT, and Deep-RobOT, which are designed to find a transformation T within a particular class of functions. These algorithms can be useful for aligning point clouds when we want to impose certain constraints on the transformation or when we have a detailed understanding of the physical processes underlying the problem. This can allow us to incorporate our prior knowledge of the transformation into the form of the function, which can help to improve the accuracy and robustness of the solution. Here are a few examples to illustrate this point:

(1) In the case where we have two LIDAR records of the same location at different times, and we want to find the displacement between the two times, we can restrict the transformation T to be a combination of translation and rotation. This is the approach taken by the Iterative Closest Point (ICP) algorithm.
(2) When the point clouds represent human or animal movements, a more complex transformation class may be appropriate, such as one that allows for rotations around the center of each joint.
(3) In the case of deformable bodies, such as an inflating balloon, the transformation T should be able to represent non-rigid deformations.

Similar to the ICP or CPD methods, the goal of optimal transport is to find the transformation T (called the transport plan) that minimizes the distance between the two point clouds. But the transformation that optimal transport gives us is similar to the transformation T of size $\mathbb{R}^{n^2}$: it is certainly the one that minimizes the underlying cost, but as is, it remains useless.

Therefore, the studied paper proposes several ways to transform the transport map T into a function of a class of interest. The rigid/affine ROBOT, which provides a transformation of the same type as the ICP and the Spline ROBOT.

The function class associated with the Spline ROBOT is a so-called spline functions.A spline function is a piecewise continuous polynomial function that is used to approximate a set of data points. It is defined by a set of control points, and the polynomial function is constructed in such a way that it is smooth and continuous over the entire domain.
In 1D, a spline function can be described this way:

$$f(x) = \begin{cases} p_1(x) & x \in [x_1, x_2] \\ p_2(x) & x \in]x_2, x_3] \\ \dots & \\ p_n(x) & x \in (x_{n-1}, x_n] \end{cases}$$

Here, f is the spline function, and p_i represents the polynomial function for the i-th interval. The intervals are defined by the control points $x_1, x_2, ..., x_n$.
In higher dimensions, the polynomial functions can be replaced by kernels that are much easier to deals with. Indeed, defining a meaningful partition of space in higher dimension become more and more difficult. Kernels allow us to only define landmarks defined by $x_1, x_2, ..., x_n$. The expression of f become:

$$f(x) = \frac{\sum_{i=1}^{n} K(x, x_i) y_i}{\sum_{i=1}^{n} K(x, x_i)}$$

2.2 Regularised Optimal Transport

The classic optimal transport problem can be written as follows:

$$\begin{aligned} OT(A,B) = \min_{\pi_{i,j}} \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{1}{2} \pi_{i,j} \|p_i - q_j\|^2 \\ \text{s.t.} \forall j, \sum_{i=1}^{N} \pi_{i,j} = q_j, \forall i, \sum_{j=1}^{M} \pi_{i,j} = p_i, \end{aligned} \tag{1}$$

where $A = \{p_1, p_2, ..., p_N\}$ and $B = \{q_1, q_2, ..., q_M\}$ are the two sets of points, and $\pi_{i,j}$ is the mass transported from point p_i in set A to point q_j in set B. The objective is to find the transportation plan that minimizes the total cost of moving the mass from set A to set B, subject to the constraints that the mass of each point in A is equal to the mass of the corresponding point in B, and the mass of each point in A is transported to exactly one point in B.

They use the entropic regularised Optimal Transport:

$$\begin{aligned} OT(A,B) = \min_{\pi_{i,j}} \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{1}{2} \pi_{i,j} \|p_i - q_j\|^2 + \sigma^2 D_{KL}(\pi_{i,j} | A \otimes B) \\ + \tau^2 D_{KL}(\sum_{j=1}^{M} \pi_{i,j} | A) \\ + \tau^2 D_{KL}(\sum_{i=1}^{N} \pi_{i,j}) | B) \end{aligned} \tag{2}$$

where $D_{KL}(Q|P)$ is the KL divergence between the two distributions Q and P, defined as:

$$D_{KL}(Q|P) = \sum_{i=1}^{N}\sum_{j=1}^{M} P_{i,j}\log\frac{P_{i,j}}{Q_{i,j}}$$

The first KL divergence can be interpreted as the term of General Entropic Regulation [Peyré and Cuturi 2018], the two second term can be interpreted as a barrier method:
We exchange the constraint $\forall i, \sum_{j=1}^{M}\pi_{i,j} = p_i$ for a softer one which penalises the fact that $\sum_{j=1}^{M}\pi_{i,j} \neq p_i$:

$$\tau^2\sum_{j=1}^{M}\pi_{i,j}\log\frac{\sum_{j=1}^{M}\pi_{i,j}}{p_i} \tag{3}$$

Given that the entropic regulation remove the case $\pi = 0$:
That is equal to 0 if and only if $\sum_{j=1}^{M}\pi_{i,j} = p_i$. This is a convex problem, and it's dual resolution ([Feydy 2020], equations 3.214 and 3.187) give us an expression of π.
This give us a way to compute the transport plan π as a $\mathbb{R}^{N*M}$ matrix. But as we saw in 2.1, we want to change the transport plan into a transformation of a chosen class of functions.
The gain is twofold: not only do we get a transformation that makes sense for the application at hand, but as soon as the number of points increases, the memory size of π is no longer manageable since its dimension is $\mathbb{R}^{n*m}$.

To work around those memory issues, the authors choose to extract just some of the information inside that matrix, indeed, we can expect that the points were p_i is mapped are close to each other so that the barycenter of the shape formed by the mapped points contained nearly all the information needed.

$$v_i = \frac{\sum_{j=1}^{M}\pi_{i,j}(q_j - p_i)}{\sum_{j=1}^{M}\pi_{i,j}} \tag{4}$$

v_i correspond to this barycenter. It represents "on average" where p_i is mapped. We also need to re-weight it so that the new point cloud is a distribution.

$$w_i = \sum_{j=1}^{M}\pi_{i,j} \tag{5}$$

$(v_i, w_i)_{i\in 1,..,N}$ represents the new point cloud, with a much lower memory footprint ($\mathbf{R}^{3N}$ instead of $\mathbf{R}^{N*M}$).

The aim is now to transform this representation of the point cloud into a transformation of a chosen class.

2.2.1 From weighted RobOT matching to rigid transformation. In this section we will illustrate how it is possible to induce a rigid transformation (translation and rotation) from the weighted RobOT matching.

We have seen how ICP works in section 1. The algorithm in the case of optimal transport stay the same. But of going from point cloud A to point cloud B, we go from A to Y with $Y = W(A + V)$ where $W = Diag(w_i)$.
Optimal Transport solve two issues of ICP:

(1) Even if A and B does not have the same number of points it's possible to perform this algorithm
(2) It replace the assignment of nearest neighbour method (that can be problematic) by optimal transport matching.

2.2.2 From weighted RobOT matching to spline transformation. The idea is nearly the same: we replace the point cloud B, that does not have necessarily the same number of points by another point cloud : $Y = W(A + V)$ with $W = Diag(w_i)$.

We obtain the adapted formula:

$$f(x) = \frac{\sum_{i=1}^{n} K(x, x_i)w_i * v_i}{\sum_{i=1}^{n} w_i * K(x, x_i)}$$

2.3 RobOT final form: Deep-RobOT

To sum up a little what we have seen:
Today state of the art method for point cloud matching use deep-learning. The aim of the methods developed in the article is to do better than them.
To do that the authors choose to combine all the good ideas to mitigate the issues of the deep-learning methods.
The final method proposed by the authors is the following:

(1) We use the Rigid RobOT method to pre-aligned the point.
(2) We use a deep-learning method, based on PointNet-PWC which achieved state of the art performance on this task.
(3) We fine-tune the results using Spline RobOT to achieve pixel-perfect accuracy that is really expensive to compute with deep-learning framework.

Each of the steps is essential to achieve the performances described in the article. It corrects several flows of the deep-learning method when used by itself. The network is not invariant by rigid transformation : this impact a lot the robustness and the transparency of the model. When medical data are at stake such as those described in the article, this is a very important point. The first pre-alignment using Rigid RobOT method to pre-aligned the points solve this issue.
The second point is similar to state of the art method, this is the core and most expensive part of the computation.
The last part allows to limit the number of training samples (that can be quite expensive to annotate) and the size model.

3 LIMITATIONS

You can find in appendix the experiments done to compare the CPD methods with Rigid-ROBOT (the code is also available (https://github.com/BenJMaurel/MVA_Geometric_Data_Analysis.git). It should be noted that these tests have not been implemented with the code provided by the GitHub of the article but re-implemented. It is therefore quite possible that these results are not completely consistent with those provided by the authors' code. Yet, those results are quite interesting: we clearly see an under-performance of the optimal transport part in the case of rotations, especially for Rigid-RobOT. This is not surprising: Optimal transport is not adapted to that kind of deformation.

That is why it might not be that clear that adding pre-alignment Rigid-RobOT before the deep-learning assignment induces an equivariance to rigid transformations as mentioned in section 3.2.
Still, one can argue that it makes sense to prefer Rigid-RobOT other

a CPD method (implemented with pycpd): the vector field (v_i, w_i) has to be computed for Spline-RobOT post-processing, so it does not cost anything to compute that pre-alignment even if it less efficient than CPD.

Another limitation that we faced during our experiments was the tuning of the method. The results presented in appendix are toy-examples on Rigid-RobOT and Spline-RobOT. However, we can see that a well-calibrated OT algorithm and an under-calibrated one does not perform at all the same way. This calibration of the blur and the reach can become quite time consuming when applied to challenging task.

REFERENCES

K Somani Arun, Thomas S Huang, and Steven D Blostein. 1987. Least-squares fitting of two 3-D point sets. *IEEE Transactions on pattern analysis and machine intelligence* 5 (1987), 698–700.

Jean Feydy. 2020. LaTeX: Geometric data analysis, beyond convolutions. *https://www.math.ens.psl.eu/~feydy/geometric_data_analysis_draft.pdf* (2020).

Gabriel Peyré and Marco Cuturi. 2018. *Computational Optimal Transport.* arXiv. https://doi.org/10.48550/ARXIV.1803.00567

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. https://doi.org/10.48550/ARXIV.1706.02413

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. (12 2016).

Zhengyang Shen, Jean Feydy, Peirong Liu, Ariel Hernán Curiale, Rubén San José Estépar, Raúl San José Estépar, and Marc Niethammer. 2021. Accurate Point Cloud Registration with Robust Optimal Transport. *CoRR* abs/2111.00648 (2021). arXiv:2111.00648 https://arxiv.org/abs/2111.00648

A APPENDIX A : SPLINE AND RIGID ROBOT ON TOY EXAMPLE

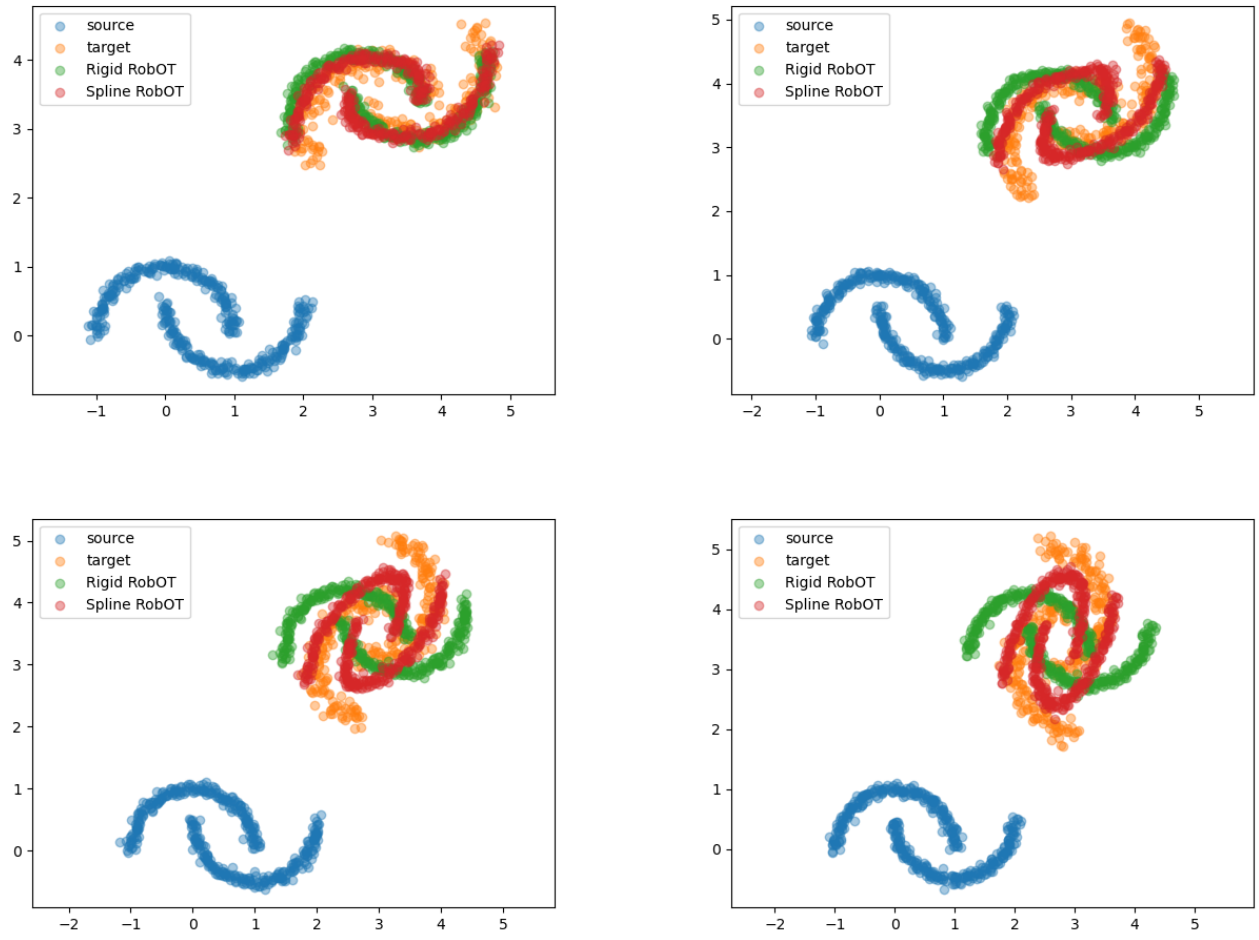

Figure 1: From left to right from top to bottom we have Rotation of 30°, 50°, 70°, 90°.

We wanted to check if Spline and Rigid RobOT performed well on toy dataset. To do that we create two points clouds of a double moon (using scikit-learn). The first one (in blue) is used as the source and has 400 points (noise level of 0.05). The second one has 300 points (noise level of 0.1) and will be used as target (in orange). We apply to the target point cloud a rotation of different angles (30°, 50°, 70°, 90°). We also apply to it a translation for visual purpose.

The transport plan is computed with a blur value $\sigma^2 = .005$ and the reach parameter $\tau = 1000$. The Spline RobOT use a Gaussian Kernel with variance of 0.5.

Two things can be noticed:

(1) Without surprised, Spline-RobOT clearly outperformed Rigid-RobOT.
(2) Both Rigid and Spline RobOT failed to retrieve the rotation induced to the target.

B APPENDIX B : CPD ON TOY EXAMPLE

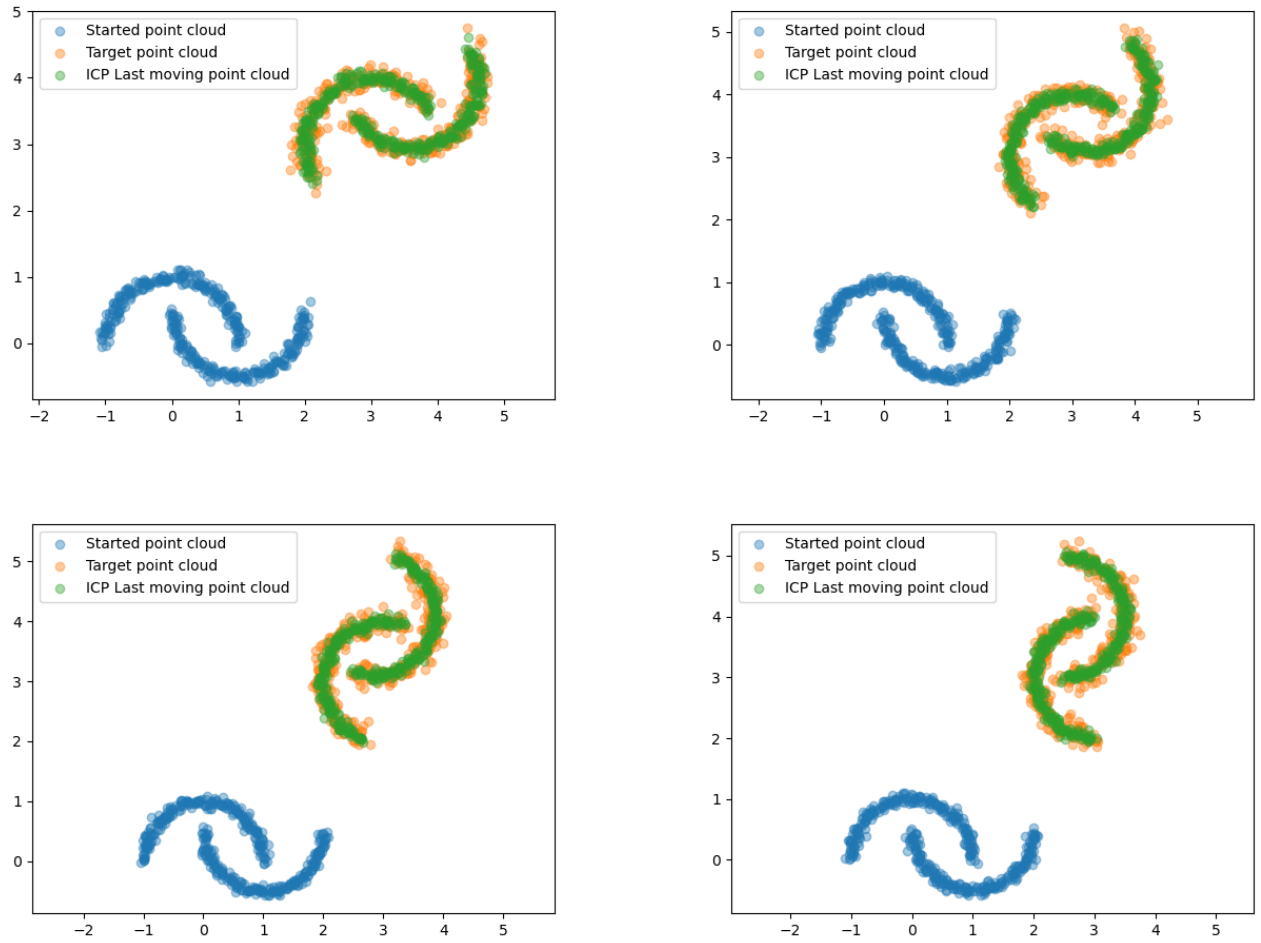

Figure 2: From left to right from top to bottom we have rotation of 30°, 50°, 70°, 90°. We used pycpd RigidTransformation function without any modification.

C APPENDIX C : FIND THE RIGHT TUNING

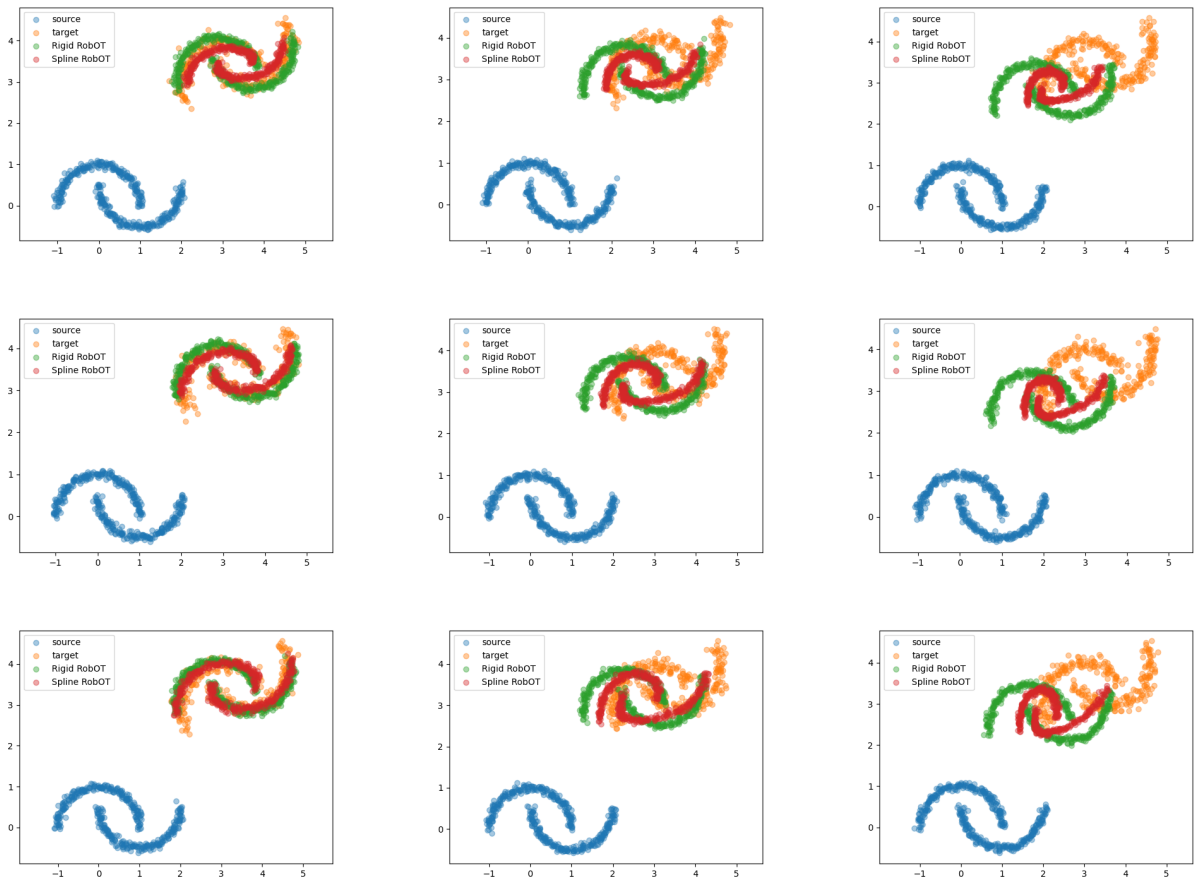

Figure 3: τ constant on a column, σ constant on a raw
$\tau = 1000, 10, 3$, $\sigma = 1, 0.5, 0.1$