# Improving curvature estimation as feature descriptor for registration of point clouds

Mattia Pannone[1]

[1]*Student of Master in Artificial Intelligence and Robotics at Sapienza University of Rome*

Abstract

This paper aim to improve the use of curvature feature to align two point clouds. In particular it is used a method called umbrella method to compute the curvature's value in each point and it is compared with different parameters and with another method called surface variance. Furthermore, the improvement is sought in a situation where there is a lot of noise in the point cloud, so the computation of curvature values can be compromised and can hinder the matching between points. Another step in this paper is to compute the local variance of a point cloud and use it in aggregation with the curvature to try to improve the registration between the point clouds. Just in reference to the registration step, it is used the Iterative closest point algorithm. The results show the effective validity of the umbrella method and of the variants implemented to increase the noise robustness. Some particular cases has to be observed.

## HIGHLIGHTS

- The **curvature** of a point is the main element for this work. In particular it is used because its invariance to translations and rotations of a same distribution of points, given that the goal is to find points with similar features
- The robustness of the **umbrella method** with respect to the surface variance method is underlined, also comparing with respect to different parameters
- Another feature is introduced to help the curvature, it is the **local variance** in a point, that is to say the variance in a neighborhood of a point
- Different computations are done to try to improve **robustness to noise** in computation of curvature values, in particular two roads are followed based on the median of the points
- The registration of two point clouds, that is the matching between the same points in different position, is done implementing the **iterative closest points algorithm** using the curvature values as main feature.

## 1. Introduction

In computer vision, which has made a lot of progress in a few years, there are many techniques used to work on images to achieve different purposes. One of these can be the matching between "pieces" of images with different parts and some in common that represent the same image whose goal is precisely to form a single image or in any case find information by exploiting the common parts. The images are generally in two dimensions formed by many pixels which are analyzed (there is a third dimension if one considers an RGB image), where to study the characteristics of these pixels there are different techniques that are based on different feature descriptors such as the Scale Invariant Feature transform (SIFT) which identifies and describes local features in an image, the Histogram of Oriented Gradients (HOG) which calculates the distribution of gradient orientations in an image and represents it as a feature vector, Local Binary Patterns (LBP): it describes the texture of an image by comparing the intensity of a central pixel with its surrounding neighbors, and then many others.

However, in this field we also analyze other types of data which, even if they have similarities to images, are represented in a different way. Just think for example of an image, or rather an object, acquired by a 3D scanner, it will be represented in space by different points with precisely 3D coordinates. Precisely with regard to scanning, an object to be well represented can be scanned from different angles thus obtaining different scans which then need to be combined to arrive at the final representation as similar as possible to reality. Also in this case, as stated before for the case of images, the points in a representation in space, called point clouds, can have feature descriptors useful for studying which points are more similar and therefore candidates for the match between different representations at in order to arrive at a unique representation through special algorithms.

In particular in this paper through a particular feature we want to find possible similar points between two single representations of the same object, in order to find a transformation matrix that leads the points of a source point cloud to overlap as much as possible to a point target cloud. The feature descriptor used is the value of the curvature at each point, above all because it is

pannone.1803328@studenti.uniroma1.it (M. Pannone)



CEUR Workshop Proceedings (CEUR-WS.org)

an invariant characteristic with respect to rotations and translations.

Curvature can be defined as a function which associates a real value to each point of the surface. In particular, in space the curvature is defined as the derivative of the tangent with respect to the surface at that point, while for a surface, depending on the direction, the direction of the maximum curvature and the minimum curvature are defined, to then define two main parameters which can be used: the mean curvature defined as the average between the maximum and minimum curvature and the Gaussian curvature defined as their product. However, in this work a new type of curvature called "Umbrella Curvature" [1] is used, which respects the fact that if a surface is "flat" or similar to a plane then the curvature is zero (like that of an open umbrella) while instead if the surface is very "curved" then the value will be high (like a closed umbrella). Furthermore, in this work another method called Surface Variance is also used which relates the variance of the data in a point and in its surroundings, this in particular is mainly used to make various comparisons and demonstrate the greater validity of the first compared to the second method. Looking into the detail of the curvature, in both methods used to calculate the value in a point the closest points are taken into account (of course the portion of the surface in consideration depends on the geometric characteristics in that portion and in particular on the position of a subset of points), but also the calculation of these neighbors may differ, it will be seen later that compared to the calculation of the closest points used in the second method which only takes into account the distance from the reference point, in the first method the closest points are calculated homogeneously , or at a certain distance and angle, just like in an umbrella with the outermost vertices and the central point of the handle.

Despite, as mentioned, the main feature studied in this work is the curvature, to try to improve the accuracy in establishing the similarity between two points a feature that works in cooperation with it is introduced: the local variance, i.e. the calculation of the variance of the data in a subset of points, which is a measure of the dispersion of the points with respect to their mean. It is added as a further filter to establish the similarity in the points, in fact if in the initial algorithm the similarity of two points was established through the difference of their curvatures, in the second one the similarity between the respective local variances is also added. This reasoning underlies the fact that multiple points can have similar curvatures even if they refer to two different parts of the same object, so we decided to add a new easily calculable feature to raise the probability of taking similar points.

Another topic covered by this work is the robustness to noise. It can often be present in point clouds, especially in 3D scanner acquisitions caused by defects and distortions of the device and/or abinental factors during the scan. Therefore, after comparing different matching techniques, random noise was added to measure the performance of the same instruments, to then finally add variations to the algorithms to try to increase the robustness to noise. In particular, the solutions concern the calculation of the median and the application of a small filter in the portion of the point cloud analysed.

In conclusion, in this work a registration algorithm for point clouds is implemented, which is precisely the process that consists in combining multiple point clouds into a single coherent and aligned representation, i.e. the final goal of the project. There are several techniques to perform this process, the algorithm chosen and implemented is the Iterative Closest Point (ICP) which has the task of finding the best geometric transformation that aligns the points of a reference point cloud (target) with the points of another point cloud (source) in an optimal way by following a series of steps (implemented in this case using the studied features) in an iterative way.

In the next sections a state of the art is presented with related works first, then the implementation of the techniques and algorithms used will be illustrated, to continue finally with the illustration of the results achieved with a final conclusion.

## 2. Related Works

In this section we will see some papers related to these topics that comprehensively present the state of the art.

The first relevant paper [1] is certainly where the authors present a new method called Umbrella Curvature where they use an 8 neighborhoods ring for each point, called homogeneous neighbours, and thanks to them they estimate the normal at the point of interest and with it calculate the value of curvature, demonstrating good performance and better accuracy at the expense of more computation time. In it, the type of method defined is compared with a method defined by many and called in different ways, including in [2], where it is defined as Surface Variation (or Surface Variance); in it the authors analyze and quantitatively compare a number of surface simplification methods for point-sampled geometry, including Surface Variance defined as the ratio between the minimum eigenvalue and the sum of the three eigenvalues of the covariance matrix defined with the point of interest and its closest points. Just about the nearby points, there are different methods for their calculation and this can be imported as precisely these points can define the geometry and give important information about the part of an object. For example in [3] the authors proposed a novel method for efficiently determining neighbors in a point set using the Elliptic Gabriel Graph (EGG) that captures the local geometric

relationships between points in a point cloud taking an elliptic influence region. In [4] the KD-Tree algorithm is introduced, which is a multidimensional binary tree used for the efficient search of nearby points in a higher dimensional space. The KD-Tree algorithm subdivides the space recursively and organizes the points into a tree where each node contains the points. Using this structure, it is possible to search for nearby points with logarithmic complexity with respect to the number of points (this is also the method used in the current work). However, the results may not be satisfactory if the point distribution is uneven, so it adds a definition of nearby points where for their calculation beyond distance the directional balance is taken into account which describes where the points are well. spread around the point of concern, this method is called homogeneous neighborhood.

The analysis of the points in a point cloud is therefore fundamental and there are many ways to analyze them, for example in [5] a multilayer perceptron network (MLP) is also used to which the unordered points in the ball are fed of neighborhoods that are reordered according to the space-filling curve (SFC), its goal was to extract geometric information in the point cloud and have a strong semantic recognition capability for objects with complex morphologies. However the latter work is more complex and applies well to more complex point clouds, such as entire maps of places. A work more similar to the current work is [6], where the authors proposed a novel method for estimating curvature based on the fitting of normal section curvatures. The key idea is to analyze the variations of the normal vectors along different directions within local neighborhoods of the point cloud. By fitting a quadratic surface to the normal vectors, the local curvature can be computed as the principal curvatures of the fitted surface. The proposed method provides a robust and accurate estimation of curvature by considering the variations in multiple directions. It overcomes some of the limitations of traditional methods that rely solely on local surface geometry or local neighborhood analysis. The fitting of normal section curvatures offers a more comprehensive and reliable approach to curvature estimation.

As for the match between the same point clouds scanned in different positions or in any case with missing parts in relation to each other, there are different algorithms for matching them. In [7] he introduces the ICP algorithm in detail, describing the registration approach based on the minimization of the mean squared error between the corresponding points, in particular in each iteration, the corresponding points between the two models are identified and it is calculated the best transformation that aligns the points. [8] uses the geometrical features of the point cloud to be registered, such as curvature, surface normal and point cloud density, to search for the correspondence relationships between two point clouds and introduces the geometric features into the error function to realize the accurate registration of two point clouds. In [9], in a similar way to the current work, the registration algorithm is based on curvature feature similarity, in particular the key contribution of the paper lies in the novel approach to measuring the similarity between curvature features. The authors introduce a robust similarity metric that takes into account the local curvature variations and aligns the corresponding features in the point clouds. This allows for a more accurate and reliable registration compared to traditional methods that rely solely on geometric or spatial information. A similar goal is for [10], which concern about development of an efficient and effective algorithm for automated registration. The method incorporates techniques such as feature extraction, matching, and optimization to achieve accurate and robust registration results. It addresses challenges such as noise, outliers, and missing data commonly present in unorganized point clouds.

A last work that I want to highlight is [11], which focuses on the classification and segmentation of point clouds. The proposed method leverages the concept of local variance, which captures the variability of point densities and surface characteristics within a neighborhood of points. By analyzing the local variance information, the algorithm can differentiate between different types of surfaces and classify them accordingly. This enables the segmentation of the point cloud into distinct regions based on their characteristics.

## 3. Implementation

In this section I will discuss all the implemented part by dividend the topics in different sections. In particular, the topics will be treated in different subsections: the data used, the calculation of the closest points, the calculation of the curvatures and the different facets, the calculation of the local variance, the description of the different algorithms and techniques used with in-depth analysis and illustration of the iterative closest point algorithm, the techniques used to try to increase the noise robustness, the main parameters used. Everything was implemented with the help of the google colab tool, where every single function was implemented and where it is possible to see all the experiments performed being divided into sections, it is also possible to run the same notebook by importing your data with the right path. The same thing can be done locally with a python file launched from the terminal, attached to this work.

After the discussion of the topics mentioned above, the results achieved with the various comparisons performed will be illustrated, thanks to which it is possible to make observations and have graphic illustrations. The conclusions will follow at the end.

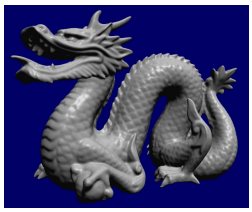
## 3.1. Data Information

The data used in this work are obviously point clouds. They are taken from the repository "The Stanford 3D Scanning Repository" (http://graphics.stanford.edu/data/3Dscanrep/), it is a widely recognized and comprehensive collection of 3D scanning data, maintained by Stanford University and contains a diverse range of 3D scanning datasets , captured using various techniques and devices, providing a rich and diverse collection for experimentation and evaluation purposes, moreover it provide many information as geometry, surface normals, and other annotations or measurements specific to each dataset. It is very useful as a valuable resource for researchers and practitioners in the field of computer vision, computer graphics, and 3D data analysis. In fact it has become a standard benchmark in these fields.

Going back to the data used in this work, 3 different point clouds representing 3 different subjects were chosen Fig. [1 (a,b,c)] (the images represent the objects in reality which, after being scanned, are represented with a series of points, the point clouds in fact) and for each of them 4 representations (scans) were taken, in which one used as target and the other 3 used as source, so for each object it was possible to do more alignment tests.

In general point clouds are three-dimensional data structures commonly used to represent the geometric properties of objects or scenes. They consist of a set of points in a three-dimensional coordinate system, typically defined by its Cartesian coordinates (x, y, z), where each point corresponds to a specific position in space. So it provides a rich representation of the object or scene's geometry, capturing detailed spatial information at discrete locations. They can be represented with various file formats, such as the ".ply" format, which is used in this project.

(a) *Bunny* (b) *Armadillo*



(c) *Dragon*

**Figure 1:** Used scanned items

## 3.2. Neighbors of a point

As mentioned, for most of the calculations carried out, it was necessary to calculate the points closest to a reference point, as it is precisely the position of these points that defines the characteristics and geometry in a certain portion of the object. In fact, more generally from a set of points it is possible to define the normal, the variance with the relative directions of minimum and maximum variance and precisely the curvature.

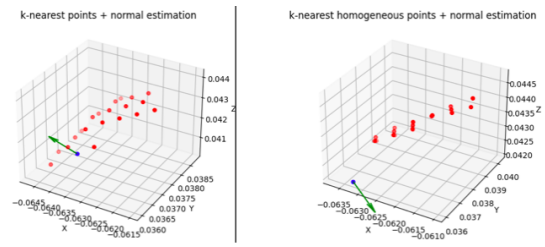
As seen in the related works illustration, there are several ways to find the closest points of a given point, however the one used in this work uses the KD-Tree algorithm, which is a tree data structure and to find the nearest points a search algorithm is used in which one starts at the root of the tree and proceeds recursively downward following the correct path based on the division values. During the search, the closest points found so far are kept and the closest point is continuously updated as you go down the tree. It is very efficient as with its balanced tree structure it allows to considerably reduce the search space while searching for the closest points, in fact the search algorithm can avoid exploring entire sub-trees when the distance between the search point and the division value exceeds the distance of the closest point so far found. So its cost is logarithmic. But as it was highlighted in the introduction, in this project not only the closest ones are used, but also the closest points that have a similar directional balance, called homogeneous neighbors, in particular used for the computations of the umbrella curvature, which are distributed homogeneously around the point of interest. This is done by calculating the cosine similarity between the direction vectors of neighboring points and the mean direction:

$$\text{cosine\_similarities} = \frac{\langle \text{directions}, \text{mean\_direction} \rangle}{\|\text{directions}\| \cdot \|\text{mean\_direction}\|}$$

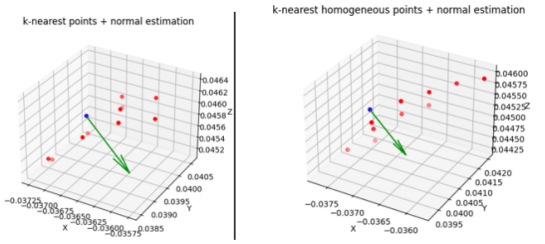
So when the nearby points are calculated, first more than the necessary ones are calculated, then the first k necessary are filtered and taken with the previously calculated value that is higher than a certain threshold (set to 0.8 in this work).

From Fig. [2 (a,b,c)] it is possible to observe the difference between the nearby points (in red) and the one of interest (in blue) calculated using the two different methods mentioned. Therefore it can be observed very well how the distribution of the points is more homogeneous with the calculation of the neighbors homogeneous with

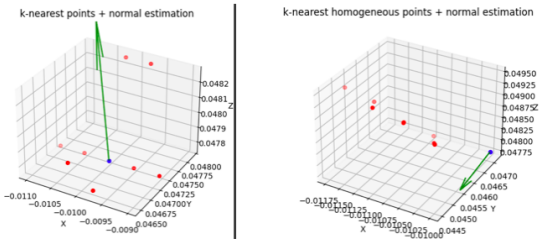
respect to the closest points based only on the distance.




(a) *First example*




(b) *Second example*




(c) *Third example*

**Figure 2:** Each pair of graphs represents the closest points (in red) of a given point (in blue) based on the distance (graphs on the left) and those distributed evenly (graph on the right). The green arrow is the normal.

Besides being used for curvature calculation, the calculated subset of points can be used to study other properties. In fact, in this work the normal vector and the local variance have also been calculated (however used in the various methods of calculating the curvature). In particular, this was done using the Principal Component Analysis (PCA) which is a statistical method that identifies the principal components that explain most of the variance and more generally can reduce the dimensionality of the data. The step followed by the algorithm to compute it are:

- center the points found (neighbors) by subtracting the mean vector from them (in the specific case of this project the points were centered, and therefore subtracted, from the point of interest)
- calculate the covariance matrix of the centered points. This matrix represents the statistical relationship between the variables (x,y,z coordinates) of the point cloud.
- compute the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the principal directions of variation, while the eigenvalues represent the amount of variance explained by each direction

Sorting the pairs of eigenvalues and eigenvectors in ascending order, it can be obtained that the eigenvector corresponding to the largest eigenvalue represents the direction of the maximum variance, while the smallest one represents the normal of the point of interest.

As regards instead the variance used as a further feature to help the curvature, as mentioned in the introduction, it is intended as a measure of dispersion which indicates how much the values in the array are distributed around their mean.

## 3.3. Curvatures Computation

As previously anticipated, curvature is the key feature of this work. In particular, a new type of curvature introduced in [1], called Umbrella Curvature, is exploited and is compared with a simpler method based on the variance of the points called Surface Variance. As already underlined in the previous subsection, for both methods it is necessary to find a group of points close to the one of interest, as the curvature is a characteristic of the object which depends on its geometry and in particular depends on the neighborhood of the point in question. question.

In the case of the "Surface Variance" method the curvature at a point is calculated as:

$$k_{SV} = \frac{\lambda_1}{\sum_{i=1}^{3} \lambda_i} \tag{1}$$

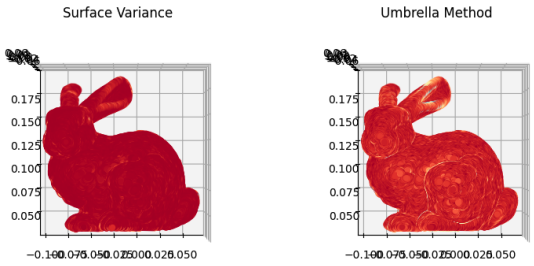
where $\lambda_i$ are the eigenvalues of the covariance matrix of the nearest points of a point of interest and $\lambda_1$ is its minimum eigenvalue. Instead for the method of the "Umbrella" curvature we have:

$$k_{UM} = \sum_{i=1}^{N} |\frac{N_i - p}{|N_i - p|} \cdot n| \tag{2}$$

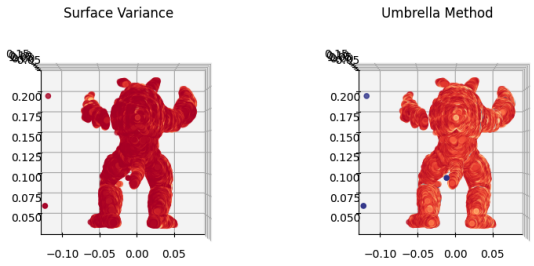
where p is the point considered and $N_i$ are all the N neighbors found, n is instead the normal found at p. So the curvature in this case is defined as the sum (in absolute value) of the projections on the normal of each vector from the point of interest to each of its neighbours. If in the previous case the value was between 0 and 1/3, (having three eigenvalues which have values between zero and one), in this case the value is between 0 and N, in fact considering a certain number of nearby points , if

the proportion of all were one, as a value we would have N which is precisely the number of neighboring points and also indicates the maximum curvature.

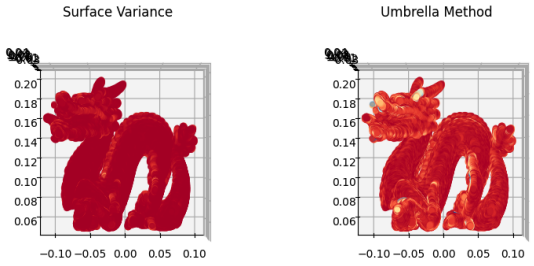
As demonstrated in the paper in which it was introduced, the Umbrella method can better capture curvature information being more accurate. This can also be seen from Fig. [3 (a,b,c)] where it is possible to notice from the colored map, where the intensity of the curvature is printed for each point (normalized for both methods between zero and one), how the intensity is more accentuated where there are more marked curves.




(a) *First example*




(b) *Second example*




(c) *Third example*

**Figure 3:** Each pair of graphs print the intensity of the curvature's value in each point from the lower (from dark red) to the higher (more and more clear until blue). It is possible to see the difference between the surface variance method (on the left) and the umbrella method (on the right). n particular, in the second it is more evident how in the presence of the curved surface of the objects the points are always lighter, meaning a higher value.

## 3.4. Robustness to noise

Another aim of this work was to improve the noise robustness. In fact, for example when scanning an object, it may happen that for various types of reasons, such as environmental conditions, instrument defects, etc. the final result is affected by noise, i.e. the points are perturbed and do not reproduce the object well enough. This can be a problem for different tasks, but especially for the one covered by this paper, i.e. the alignment of point clouds based on the similarity between the values of the curvatures in the points. In fact, as mentioned, the properties, including the curvature of a point, depend in particular on the distribution of its neighboring points, therefore if the data are clearly perturbed, this can affect the quality of the final result. To carry out various tests, both to demonstrate the differences between alignment with and without noise, and to test the techniques put in place to try to mitigate the presence of any noise, Gaussian noise was added in one of the two point clouds used for alignment (for each test).

In particular, two different techniques have been adopted to try to calculate the umbrella curvature in the presence of noise. It should be specified that an obvious solution would have been to use a Gaussian filter, having added Gaussian noise, however this route was not taken into consideration as, as mentioned, the obvious choice that would have completely eliminated the noise before calculating the curvature , instead the goal was precisely to calculate the curvature in the presence of noise. In both methods the median of the points found (the closest points of the point of interest) is used. It is a measure of central position of a data set, it is the value that is located in the center of an ordered data set, such that it has an equal number of data points above and below it. It can be used to mitigate Gaussian noise because it is less sensitive to outliers, i.e. these outliers, since it only considers the central value of the data set.
In a first method, it was used the same base formula (1), however the points are centred with respect to the median, so from these point is subtracted the median:

$$\text{new\_points} = \text{neighbors} - \textbf{median}(\text{neighbors})$$

In the other method, before applying the formula (1), the neighbors are filtered substituting for each of them the median of a window containing k points:

$points \leftarrow$neighbors of point of interest
$window \leftarrow [[0,0,0], ...[0,0,0]]$
$points_{filtered} \leftarrow [[0,0,0], ...[0,0,0]]$
**for** $i$ **in range**$(number\_of\_neighbors)$ **do**
    $window[:-1] \leftarrow window[1:]$
    $window[-1] \leftarrow points[i]$
    $points_{filtered}[i] \leftarrow$ **median**$(window)$
**end for**

In the section Results will be showed the different results obtained.

## 3.5. Registration algorithm

The last step after calculating the closest points and the curvatures for each point is the registration between point clouds. As mentioned, with this term we refer to different algorithms to allow the match between points which, based on certain characteristics, are considered to be the same points (but rotated and translated). So the goal is to find a transformation matrix that allows the rototranslation of a point cloud to better align it with another one.

The algorithm used is the Iterative Closest Point (ICP) [8], which is one of the most used in this type of task because it can be adapted to handle various variants. The main steps are:

1. identify correspondences between the points of the source point cloud and the destination point cloud
2. calculate a transformation that minimizes the distance between the points of the source point cloud and the destination point cloud
3. apply the transformation to the source point cloud, aligning the points with the destination point cloud.
4. iterate the previous steps until convergence is reached, a predefined termination criterion is satisfied, or a maximum number of iterations is reached

To compare the "quality of the recording", surely the plot of both point clouds on the same plane can be judged "visually" how much they overlap after the completion of the recording algorithm, however another metric is also used, i.e. the registration error, which indicates the mean between the Euclidean distance between the points of the two point clouds, or rather, between the points for which a similarity has been found and therefore which are considered matched. Therefore this distance $d$, considering two points $p1_A[x, y, z]$ and $p1_B[x, y, z]$ of two point clouds $A$ and $B$ is defined as:

$$d_i = \sqrt{\sum_{i=1}^{3}(p1[i] - p2[i])^2}$$

and the registration error, considering to have N couples of points is:

$$registration\_error = \frac{\sum_{i=1}^{N} d_i}{N}$$

Now let's see the specific implementation of the ICP algorithm used, i.e. a version of this algorithm that exclusively uses the similarity between the values of the curvatures to understand if it can be the same point, in fact points with similar curvature values are much more likely to be candidate points for the match, even if this is not obvious, just think of points in areas with similar curvature because the surrounding geometry is similar even if perhaps they are different areas. So to compare the curvatures, the difference in absolute value is made, and if it is low enough, below a certain threshold, then

**Algorithm 1** ICP based on curvature feature

$new\_source\_pc \leftarrow []$
$new\_target\_pc \leftarrow []$
$curv \leftarrow []$
$diff \leftarrow []$
**for** $(p_{source}, c_{source})$ **in** $(pc_{source}, curvatures_{source})$ **do**
    $near\_points\_idx \leftarrow$ **neighborhood**$(k, p_{source}, pc_{target})$
    **for** $i$ **in** $near\_points\_idx$ **do**
        $curv \leftarrow$ **append**$(target\_curvatures[i])$
    **end for**
    **for** $c_{target}$ **in** $curv$ **do**
        $diff \leftarrow$ **append**$(|c_{source} - c_{target}|)$
    **end for**
    **if** $\min(diff) \leq \tau_1$ **then**
        $i_{target} \leftarrow near\_points\_idx[$**argmin**$(diff)]$
        $p_{target} \leftarrow target\_pc[i_{target}]$
        $new\_source\_pc \leftarrow$ **append**$(p_{source})$
        $new\_target\_pc \leftarrow$ **append**$(p_{target})$
    **end if**
**end for**
**return** $new\_source\_pc, new\_target\_pc$

**Algorithm 2** ICP based on curvature feature and local variance

1: $new\_source\_pc \leftarrow []$
2: $new\_target\_pc \leftarrow []$
3: $curv \leftarrow []$
4: $diff \leftarrow []$
5: **for** $(p_{source}, c_{source})$ **in** $(pc_{source}, curvatures_{source})$ **do**
6: $\quad near\_points\_idx \leftarrow$ **neighborhood**$(k, p_{source}, pc_{target})$
7: $\quad$ **for** $i$ **in** $near\_points\_idx$ **do**
8: $\quad\quad curv \leftarrow$ **append**$(target\_curvatures[i])$
9: $\quad$ **end for**
10: $\quad$ **for** $c_{target}$ **in** $curv$ **do**
11: $\quad\quad diff \leftarrow$ **append**$(|c_{source} - c_{target}|)$
12: $\quad$ **end for**
13: $\quad$ **if** $\min(diff) \leq \tau_1$ **then**
14: $\quad\quad i_{target} \leftarrow near\_points\_idx[$**argmin**$(diff)]$
15: $\quad\quad p_{target} \leftarrow target\_pc[i_{target}]$
16: $\quad\quad nearest_{source} \leftarrow$ **neighborhood**$(k, p_{source}, pc_{source})$
17: $\quad\quad nearest_{target} \leftarrow$ **neighborhood**$(k, p_{target}, pc_{target})$
18: $\quad\quad variance_{source} \leftarrow$ **var**$(p_{source}, nearest_{source})$
19: $\quad\quad variance_{target} \leftarrow$ **var**$(p_{target}, nearest_{target})$
20: $\quad\quad variance \leftarrow |variance_{target} - variance_{source}|$
21: $\quad\quad$ **if** $variance \leq \tau_2$ **then**
22: $\quad\quad\quad new\_source\_pc \leftarrow$ **append**$(p_{source})$
23: $\quad\quad\quad new\_target\_pc \leftarrow$ **append**$(p_{target})$
24: $\quad\quad$ **end if**
25: $\quad$ **end if**
26: **end for**
27: **return** $new\_source\_pc, new\_target\_pc$

the points are taken as a pair of points, from which total of pairs the transformation matrix will be calculated. Specifically, all the curvatures of the first point cloud are not compared with all the curvatures of the second point cloud, but for each point in the source point cloud the closest $k$ points belonging to the destination point cloud are calculated and, among them , the differences are calculated and the smallest difference is taken. The algorithm is executed for a fixed number of iterations, so that the source point cloud is "closer" to the destination one, calculating a transformation matrix each time. There are two cases in which the algorithm does not reach the set number of iterations: if it is not possible to find a transformation matrix (no candidate points for the match have been found) or if the registration error no longer decreases for a fixed number of iteration (an early stopping has been implemented).

The Algorithm [1] has the implementation just described, while the Algorithm [2]] has a small variation, i.e. adding local variance control (previously defined) right after the curvature control, in a way almost the same as this last (control on the module of the difference, with a fixed threshold). In particular the difference can be seen in lines 16 - 24 in which the calculation and control of the variance of the points considered is added. The functions in bold used are "neighborhood" to find k points close to a point in a given point cloud, "append" to insert an item into a list, "var" to compute the variance from a given set of points (point of interest chained to its neighbors found). The two algorithms described are actually in both cases the central heart of the whole ICP algorithm, after them, the two returned matrices represent the pairs of points for which a match has been found, therefore from them is calculated but matrix of transformation, after which this transformation is applied and finally the registration error is calculated. As regards the transformation matrix, it is calculated as following:

1: $new\_source\_pc \leftarrow$ returned by Algorithm
2: $new\_target\_pc \leftarrow$ returned by Algorithm
3: $centroid_{source} \leftarrow$ **mean**$(new\_source\_pc))$
4: $centroid_{source} \leftarrow$ **mean**$(new\_target\_pc))$
5: $source \leftarrow new\_source\_pc - centroid_{source}$
6: $target \leftarrow new\_target\_pc - centroid_{target}$
7: $covaraiance \leftarrow source \cdot target$
8: $U, S, V^T \leftarrow$ **SVD**$(covariance)$
9: $R \leftarrow (V^T)^T \cdot U^T$
10: $t \leftarrow centrid_{target} - (R \cdot source_{centroid})$
11: $T \leftarrow$ concatenate $R$ and $t$

So, the final transformation matrix to apply to the source point cloud is given by:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

### 3.6. Parameters used

In this project there are few parameters used, i.e. those variables that can change the results based on how they are set. In particular, several experiments, which will be illustrated in the next section, were made by looking at two variables in particular: the number of nearby points to find each time they were searched and the thresholds used for selecting the points to match in the control phases. As for the former, different approaches were followed based on what the closest points were used for: to calculate the Umbrella Curvature, as in the original paper, the closest 8 points were used, this number was also used with the Surface Variance method to allow for a fair comparison. More in detail, when calculating the Umbrella Curvature, the searched neighbors are more and then only 8 are selected, this is because using a cosine similarity threshold to take homogeneous neighbors, then more are considered in the case some are discarded. Additionally, some tests were also done with 100 points closer. The closest points used during the execution of the ICP algorithm, were 50, this to allow to have enough to compare the curvature of the starting point, therefore in the execution of the algorithm the minimum curvature between the 50 is taken found (if within threshold). A further number of neighbors used is that in Algorithm [2] when calculating the variance in the origin point and in the destination point used, in this case 1000 neighbors were used, as the variance is a strongly dependent characteristic from the distribution of data, therefore it was decided to take it more into consideration.
As regards the thresholds used in both algorithms presented, after several statistical studies on the minimum, average and maximum values detected in some of the point clouds taken into consideration, the threshold was set at 0.0001, both in the case of the minimum difference between curvatures , than the one between variances.

## 4. Results

There are several tests carried out to compare the different ways to calculate the curvatures and to play with the noise; in particular, as mentioned at the beginning, the scans of 3 different objects were used, for each of which four were taken, i.e. 4 point clouds, one of which was taken as the destination and the others as the source (to be aligned with the destination). In the last pages of this paper, in Appendices A and B, some tables (A) are illustrated where the different registration errors are compared and a series of figures (B) show the alignment phases of some tests (Material inserted in the Appendices because of the high number of tables and images, to maintain the document ordered).

The first comparison made is the registration using the Umbrella method and the Surface Variance method and then compare them, in which from the tables [1, 2, 3] it can be seen that for almost all tests the registration error of the Umbrella method it is almost always lower (or at most slightly higher) than that of the Surface Variance, which is also achieved in fewer and fewer iterations in the first method. This last data regarding the number of iterations may seem to go against the trend of what was stated in [1] where it is said that the method introduced is more precise at the expense of greater execution time, however this paper does not study the time employed for the execution of the algorithm, but, as mentioned, the number of iterations. Another important thing to note is that a low value of the registration error does not necessarily imply a correct alignment: it can in fact be observed in Figs. [5, 6, 8, 9] that the point clouds are not aligned in a particularly well. Observing the figures and the initial position with which the algorithm starts, it can be seen that if the point clouds start from particularly different positions and furthermore some parts have excessively similar curvatures (note in Fig. [5] the head with the back of Bunny), then the algorithm can terminate as it cannot find other transformations to decrease the registration error. Figs [4, 5, 6, 7, 8, 9, 10, 11, 12] represent different iterations of Algorithm [1] on different point clouds using the Umbrella Curvature.

A test was also done to try to reduce the execution time in the case of very large point clouds (in those used the points were approximately between 10,000 and 40,000); so what has been done is to reduce the number of points by eliminating a certain percentage randomly to check how they affected the geometry of the object. In particular, 50% of the points were removed from all the point clouds and the results are shown in the table [4]. It can be noted here that the difference between the registration errors is less relevant, furthermore the values achieved are always greater than those achieved with the complete point clouds (see for each object the "Test 1" line of the tables [1, 2, 3]).

The subsequent comparison was made between the two algorithms presented before, Algorithms [1] and 2], i.e. between those that exploit only the curvature as features and the one that also controls the local variance. From the tables [5, 6] we can see some improvements on the registration error (not perceptible graphically compared to the Figs. [4-12], which is why the figures are not shown), however this does not happen for the last object considered exam (table [7]). It can therefore be deduced that the quality of the point cloud or in any case the geometry of the object can determine whether or not

these features depend on the distribution of the points. As regards the number of iterations, a fixed pattern is not observed between the comparisons which allows to conclude that in one case they are less/greater than the other.

From the tables [8, 9, 10] comparisons emerge using a different number of points close to the point of interest to calculate the umbrella curvature, in particular first using 8 points and then 100, therefore slightly changing the information taken in the area of the point of interest. These tests were carried out in particular because around a point, if too little information is taken (in this case the number of neighbors) it may not be sufficient, if too much is taken it could lead to errors due precisely to the influence of regions too far from the point of interest. From the comparisons it can be seen that, even if the number of iterations is almost always lower if closer ones are used, however there are no particular differences in the registration error, i.e. sometimes it is lower in one method, sometimes in another, in each case with little difference.

The latest tests are aimed at studying the calculation of the curvature in the presence of noise, in particular for the tests performed with Gaussian noise added randomly to the destination point cloud. In this regard, the ICP was performed using the Umbrella method as used up to now, subsequently 2 variants were implemented, i.e., as illustrated in the previous section, umbrella 2 which centers the points with respect to the median and umbrella 3 which applies a small point filter. With the latter, the algorithm [2] with the local variance was also tested. It is possible to visualize the results both in the tables [11, 12, 13] and in the Figs. [13, 14, 15]. Both from the registration errors and from the figures it is possible to notice that using the small expedients in the calculation of the curvature, in the umbrella2 and umbrella3 methods there are improvements in the registration. However the use of the local variance seems not to be suitable in the presence of noise because despite the low registration error, graphically we note that the alignment fails, moreover in the Dragon object in this last case the algorithm cannot find points to be matched to calculate the transformation matrix.

## 5. Conclusions

In this paper curvature has been studied as a feature descriptor to allow the alignment of different point clouds depicting the same object but scanned in a different way (therefore with different orientation and/or missing parts). We started from the study of the new calculation method introduced in a recent paper, called Umbrella Method, to then carry out various tests to validate its effectiveness compared to other methods using this feature in an alignment algorithm. Subsequently, noise was introduced into the point clouds to find a computational method that would increase the robustness to this noise.

In the tests carried out it was possible to verify the effective validity of the Umbrella method as a method of calculating the curvature compared to other methods, looking at the images aligned by the algorithm but above all looking at the measurement of the registration error, which indicated in a nutshell the average distance between the points of the two point clouds considered. However, looking at these two measures, it should be noted that in some cases a low registration error does not necessarily imply a good alignment, as the algorithm may no longer be able to find points to match. In addition to the curvature, a variant of the algorithm was tested that made use of the local variance as an additional feature to look for points similar enough to match, which in some cases worked well, in others better and in others still did not work . It can therefore be concluded that the results can change according to the point clouds used in the algorithms introduced, which, if with an initial position that is too different and/or with parts of the surfaces that are too similar, can block the algorithms with unsatisfactory results. The last thing studied is the robustness to noise, for which two variants have been introduced to the calculation of the curvature with the umbrella method, which have proven to be more effective than the first method in the presence of noise. Again the results may depend on the shape of the point clouds and the noise distribution in them.

In conclusion, despite having demonstrated good alignment capabilities, the algorithms used could be improved to find a way that allows them to go on with the iteration if the point clouds are not aligned but the registration error no longer improves; but other ways could also be found to increase the robustness to noise.

## References


[1] A. Foorginejad, K. Khalili, Umbrella curvature: A new curvature estimation method for point clouds, Procedia Technology 12 (2014) 347–352.

[2] M. Pauly, M. Gross, L. P. Kobbelt, Efficient simplification of point-sampled surfaces, ACM Transactions on Graphics (TOG) 22 (2003) pp. 669–677.

[3] J. C. Park, H. Shin, B. K. Choi, Elliptic gabriel graph for finding neighbors in a point set and its application to normal vector estimation, Computer-Aided Design 38 (2006) pp. 975–986.

[4] J. H. Friedman, J. L. Bentley, R. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Transactions on Mathematical Software (TOMS) 3 (1977) pp. 209–226.

[5] X. Xiang, L. Wang, W. Zong, G. Li, Extraction of

local structure information of point clouds through space-filling curve for semantic segmentation, ISPRS Journal of Photogrammetry and Remote Sensing 122 (2016) pp. 108–123.

[6] X. Zhang, H. Li, Z. Cheng, Curvature estimation of 3d point cloud surfaces through the fitting of normal section curvatures, Pattern Recognition 45 (2012) pp. 3250–3263.

[7] P. J. Besl, N. D. McKay, A method for registration of 3-d shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992).

[8] Y. He, B. Liang, J. Yang, S. Li, J. He, An iterative closest points algorithm for registration of 3d laser scanner point clouds with geometric features, Sensors 17 (2017). doi:10.3390/s17081862.

[9] Z. Yao, Q. Zhao, X. Li, Q. Bi, Point cloud registration algorithm based on curvature feature similarity, Measurement 177 (2021) 109274.

[10] K.-H. Bae, D. D. Lichti, A method for automated registration of unorganised point clouds, ISPRS Journal of Photogrammetry & Remote Sensing 63 (2008) 36–54.

[11] D. Belton, D. D. Lichti, Classification and segmentation of terrestrial laser scanner point clouds using local variance information, in: ISPRS Commission V Symposium 'Image Engineering and Vision Metrology', 2006.

# A. Tables

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella Method | 19 | 0.0025711 |
| | Surface Variance | 47 | 0.0027847 |
| **Test 2** | Umbrella Method | 13 | 0.0082389 |
| | Surface Variance | 15 | 0.0093368 |
| **Test 3** | Umbrella Method | 21 | 0.0079683 |
| | Surface Variance | 47 | 0.0093608 |

**Table 1**
This table shows the execution of the ICP algorithm on 3 different models of the **Bunny** object, you can see the comparison between the use of the Surface Variance method and that of the Umbrella Curvature.

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella Method | 35 | 0.0039208 |
| | Surface Variance | 40 | 0.0037737 |
| **Test 2** | Umbrella Method | 17 | 0.009969 |
| | Surface Variance | 56 | 0.0122429 |
| **Test 3** | Umbrella Method | 21 | 0.0093331 |
| | Surface Variance | 29 | 0.0134528 |

**Table 2**
This table shows the execution of the ICP algorithm on 3 different models of the **Armadillo** object, you can see the comparison between the use of the Surface Variance method and that of the Umbrella Curvature.

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella Method | 38 | 0.0022854 |
| | Surface Variance | 43 | 0.0022744 |
| **Test 2** | Umbrella Method | 33 | 0.0035608 |
| | Surface Variance | 36 | 0.0042893 |
| **Test 3** | Umbrella Method | 18 | 0.0068711 |
| | Surface Variance | 43 | 0.0061069 |

**Table 3**
This table shows the execution of the ICP algorithm on 3 different models of the **Dragon** object, you can see the comparison between the use of the Surface Variance method and that of the Umbrella Curvature.

| | | Iterations | Registration Error |
|---|---|---|---|
| **Bunny** | Umbrella Method | 19 | 0.003647 |
| | Surface Variance | 39 | 0.0036281 |
| **Armadillo** | Umbrella Method | 13 | 0.0096267 |
| | Surface Variance | 41 | 0.0049265 |
| **Dragon** | Umbrella Method | 48 | 0.0029148 |
| | Surface Variance | 36 | 0.0031384 |

**Table 4**
This table shows the execution of the ICP algorithm on all 3 objects (with only 2 point clouds for each object) using reduced size point clouds, i.e. randomly deleting some points (in order to reduce the execution time).

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella Method | 19 | 0.0025711 |
| | Umbrella + Local Variance | 32 | 0.0018554 |
| **Test 2** | Umbrella Method | 13 | 0.0082389 |
| | Umbrella + Local Variance | 14 | 0.0044293 |
| **Test 3** | Umbrella Method | 21 | 0.0079683 |
| | Umbrella + Local Variance | 18 | 0.0043945 |

**Table 5**
This table shows the execution of the ICP algorithm on 3 different tests of the **Bunny** object and compares the results achieved with only the use of the curve as a feature Algorithm [1]] and that with the addition of the local variance Algorithm [2].

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella Method | 35 | 0.0039208 |
| | Umbrella + Local Variance | 10 | 0.0024387 |
| **Test 2** | Umbrella Method | 17 | 0.009969 |
| | Umbrella + Local Variance | 23 | 0.0032814 |
| **Test 3** | Umbrella Method | 21 | 0.0093331 |
| | Umbrella + Local Variance | 14 | 0.0024006 |

**Table 6**
This table shows the execution of the ICP algorithm on 3 different tests of the **Armadillo** object and compares the results achieved with only the use of the curve as a feature Algorithm [1] and that with the addition of the local variance Algorithm [2]].

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella Method | 38 | 0.0022854 |
| | Umbrella + Local Variance | 12 | 0.0059879 |
| **Test 2** | Umbrella Method | 33 | 0.0035608 |
| | Umbrella + Local Variance | 16 | 0.0060449 |
| **Test 3** | Umbrella Method | 18 | 0.0068711 |
| | Umbrella + Local Variance | 20 | 0.0061974 |

**Table 7**
This table shows the execution of the ICP algorithm on 3 different tests of the **Dragon** object and compares the results achieved with only the use of the curve as a feature Algorithm [1] and that with the addition of the local variance Algorithm [2]].

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella with 8 neighbors | 19 | 0.0025711 |
| | Umbrella with 100 neighbors | 18 | 0.0025364 |
| **Test 2** | Umbrella with 8 neighbors | 13 | 0.0082389 |
| | Umbrella with 100 neighbors | 15 | 0.0090874 |
| **Test 3** | Umbrella with 8 neighbors | 21 | 0.0079683 |
| | Umbrella with 100 neighbors | 10 | 0.0051346 |

**Table 8**
This table shows the execution of the ICP algorithm on 3 different tests of the **Bunny** object and compares the results achieved with the calculation of the Umbrella curvature with 8 neighbors and the calculation of the same with 100 neighbors.

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella with 8 neighbors | 35 | 0.0039208 |
| | Umbrella with 100 neighbors | 15 | 0.0076229 |
| **Test 2** | Umbrella with 8 neighbors | 17 | 0.009969 |
| | Umbrella with 100 neighbors | 11 | 0.0056058 |
| **Test 3** | Umbrella with 8 neighbors | 21 | 0.0093331 |
| | Umbrella with 100 neighbors | 8 | 0.0130134 |

**Table 9**
This table shows the execution of the ICP algorithm on 3 different tests of the **Armadillo** object and compares the results achieved with the calculation of the Umbrella curvature with 8 neighbors and the calculation of the same with 100 neighbors.

| | | Iterations | Registration Error |
|---|---|---|---|
| **Test 1** | Umbrella with 8 neighbors | 38 | 0.0022854 |
| | Umbrella with 100 neighbors | 20 | 0.0033579 |
| **Test 2** | Umbrella with 8 neighbors | 33 | 0.0035608 |
| | Umbrella with 100 neighbors | 12 | 0.0048088 |
| **Test 3** | Umbrella with 8 neighbors | 18 | 0.0068711 |
| | Umbrella with 100 neighbors | 12 | 0.0047976 |

**Table 10**
This table shows the execution of the ICP algorithm on 3 different tests of the **Dragon** object and compares the results achieved with the calculation of the Umbrella curvature with 8 neighbors and the calculation of the same with 100 neighbors.

| | Iteration | Registration error |
|---|---|---|
| **Noise + Umbrella** | 15 | 0.0072602 |
| **Noise + Umbrella2** | 14 | 0.0048573 |
| **Noise + Umbrella3** | 13 | 0.0050148 |
| **Noise + Umbrella3 + local variance** | 9 | 0.0012773 |

**Table 11**
This table shows the execution of the ICP algorithm between two point clouds of the **Bunny** object where Guassian noise has been added to one of them, the source point cloud. The results show the execution after calculating: in the first row the curvature with the classic umbrella method; in the second by subtracting the median from the points (umbrella2); in the third row by applying a small filter to the points (umbrella3); in the last one as in the previous one but adding the control on the local variance (algorithm 2).

| | Iteration | Registration error |
|---|---|---|
| **Noise + Umbrella** | 25 | 0.0086797 |
| **Noise + Umbrella2** | 9 | 0.0098761 |
| **Noise + Umbrella3** | 21 | 0.0079397 |
| **Noise + Umbrella3 + local variance** | 2 | 0.0021998 |

**Table 12**
This table shows the execution of the ICP algorithm between two point clouds of the **Armadillo** object where Guassian noise has been added to one of them, the source point cloud. The results show the execution after calculating: in the first row the curvature with the classic umbrella method; in the second by subtracting the median from the points (umbrella2); in the third row by applying a small filter to the points (umbrella3); in the last one as in the previous one but adding the control on the local variance (algorithm 2).

| | Iteration | Registration error |
|---|---|---|
| **Noise + Umbrella** | 11 | 0.0100419 |
| **Noise + Umbrella2** | 27 | 0.0052735 |
| **Noise + Umbrella3** | 24 | 0.004230 |
| **Noise + Umbrella3 + local variance** | - | - |

**Table 13**
This table shows the execution of the ICP algorithm between two point clouds of the **Dragon** object where Guassian noise has been added to one of them, the source point cloud. The results show the execution after calculating: in the first row the curvature with the classic umbrella method; in the second by subtracting the median from the points (umbrella2); in the third row by applying a small filter to the points (umbrella3); in the last one as in the previous one but adding the control on the local variance (Algorithm 2).

## B. Figures

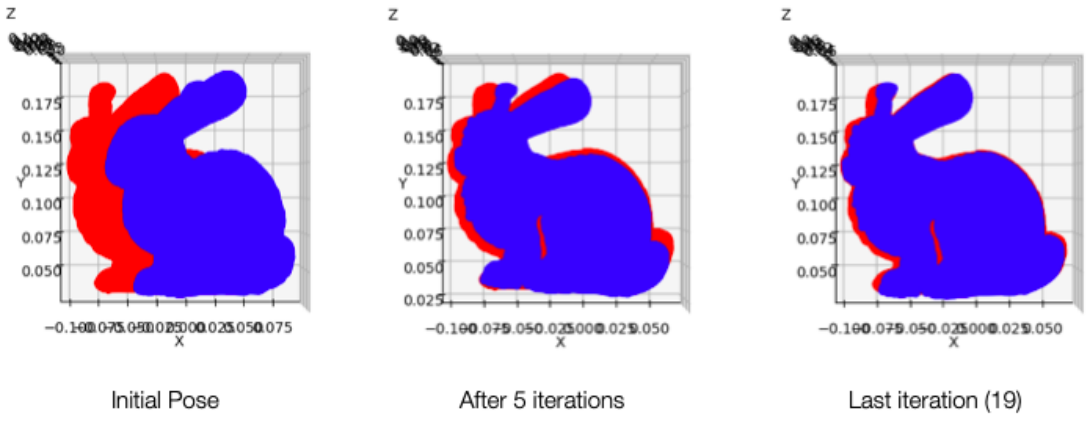



**Figure 4:** Test 1 of Bunny registration.

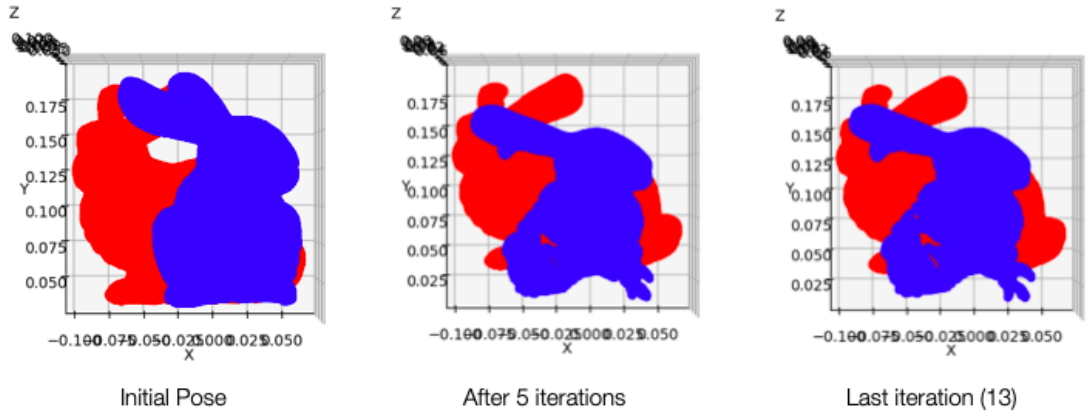



**Figure 5:** Test 2 of Bunny registration.

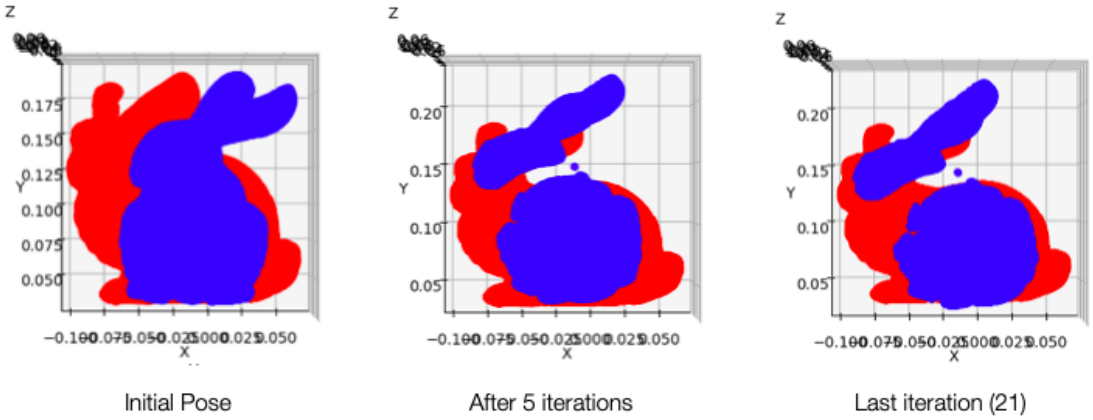



**Figure 6:** Test 3 of Bunny registration.

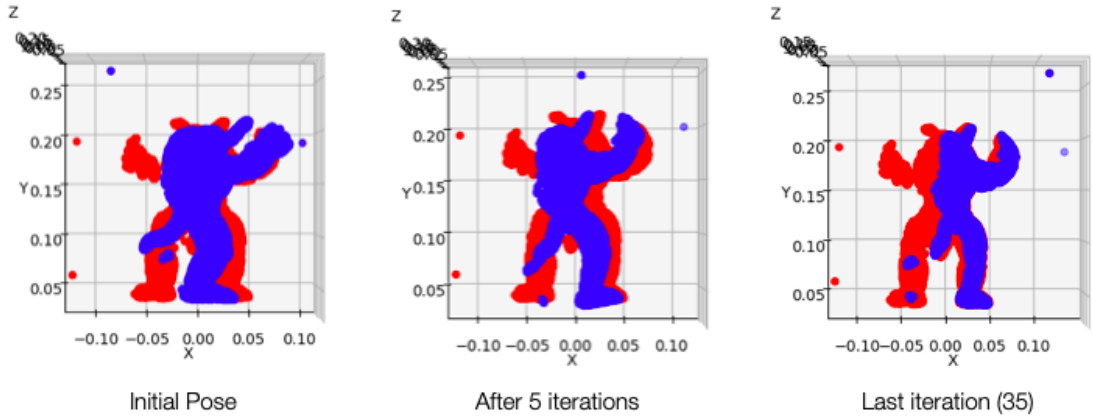



**Figure 7:** Test 1 of Armadillo registration.

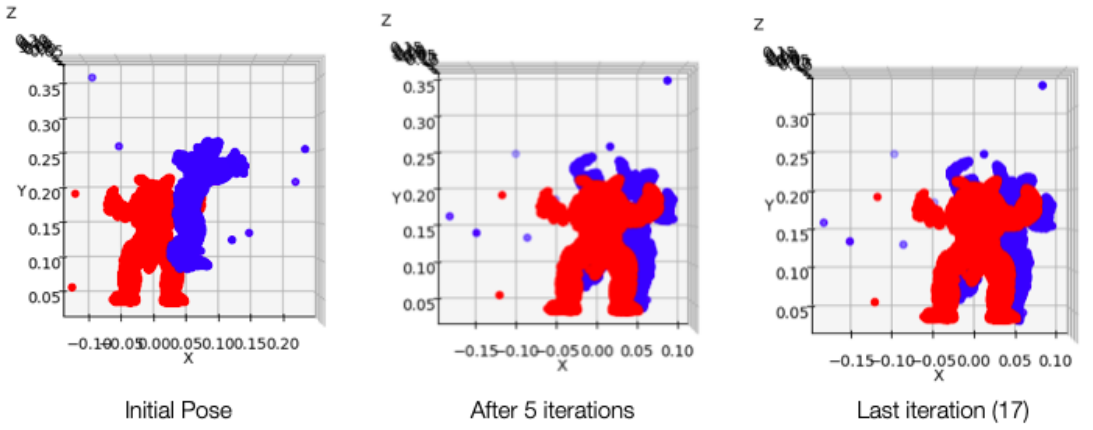



**Figure 8:** Test 2 of Armadillo registration.

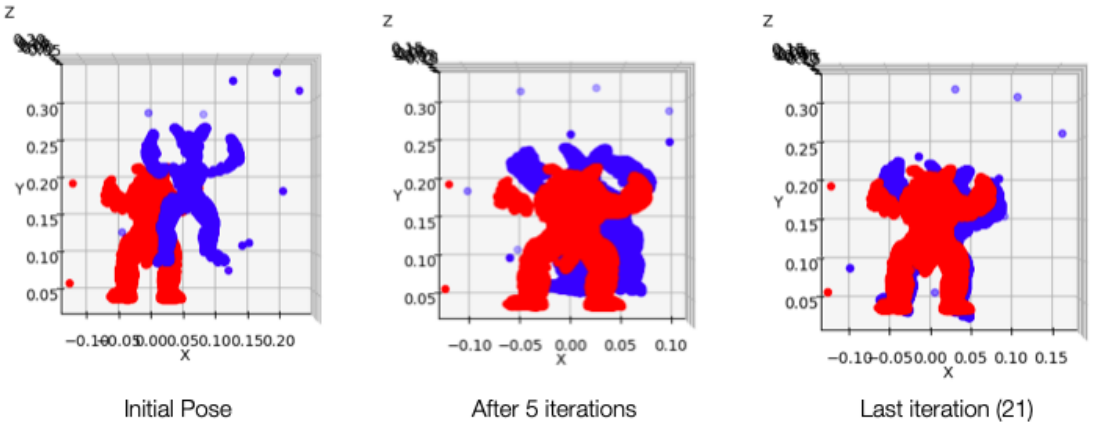



**Figure 9:** Test 3 of Armadillo registration.

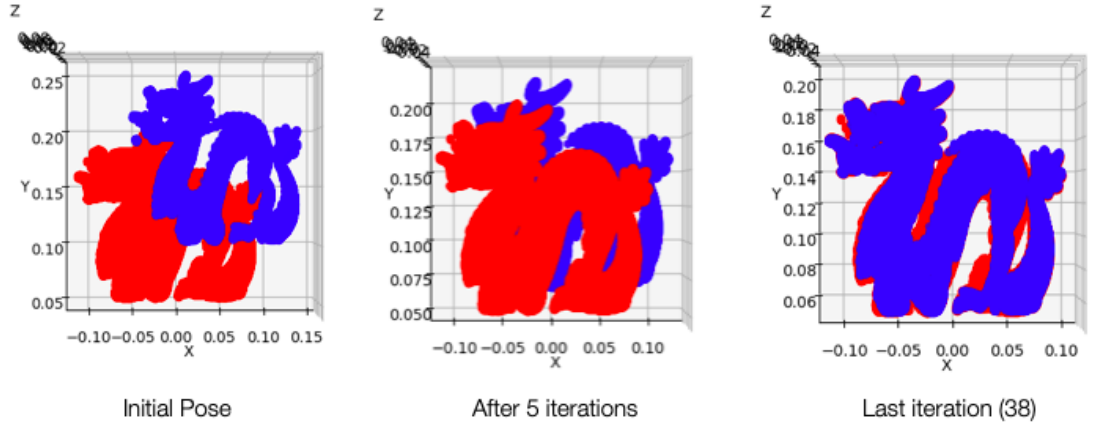



**Figure 10:** Test 1 of Dragon registration.

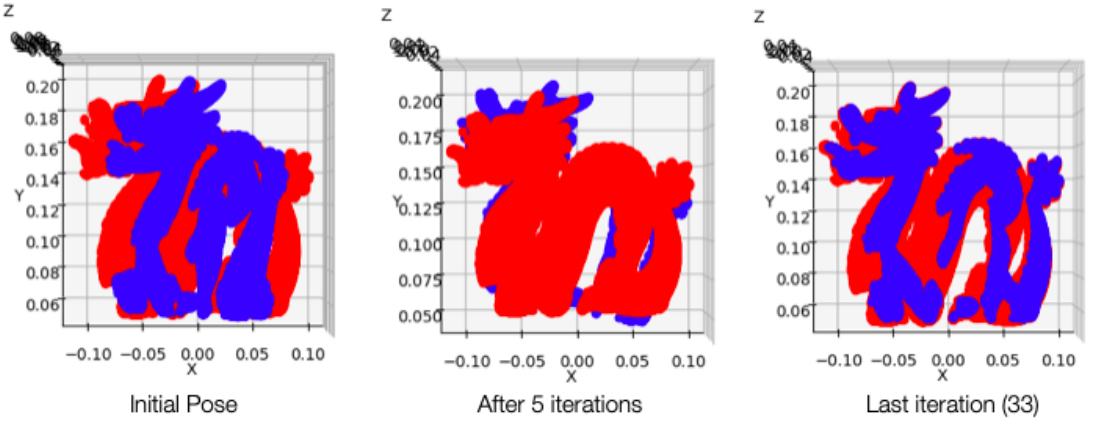



**Figure 11:** Test 2 of Dragon registration.

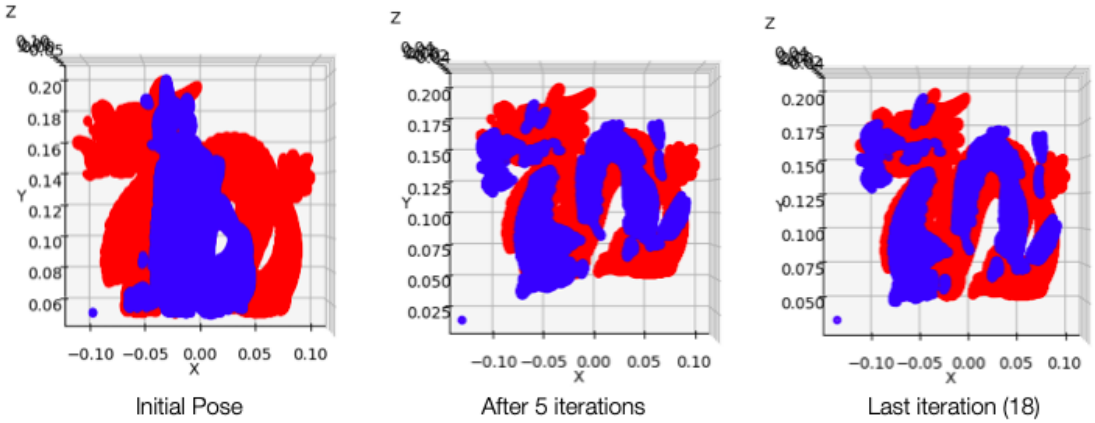



**Figure 12:** Test 3 of Dragon registration.

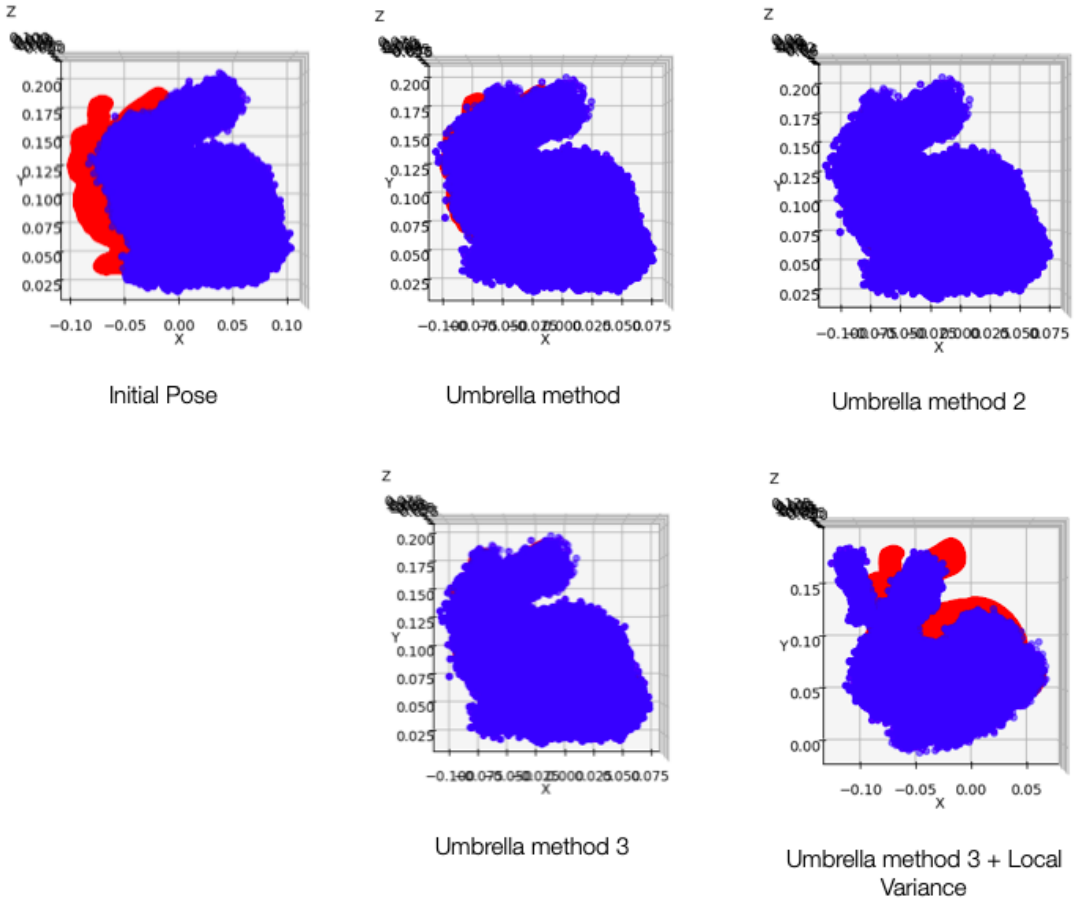



**Figure 13:** Test of Bunny registration on the point cloud with noise and the different methods to deal with it.

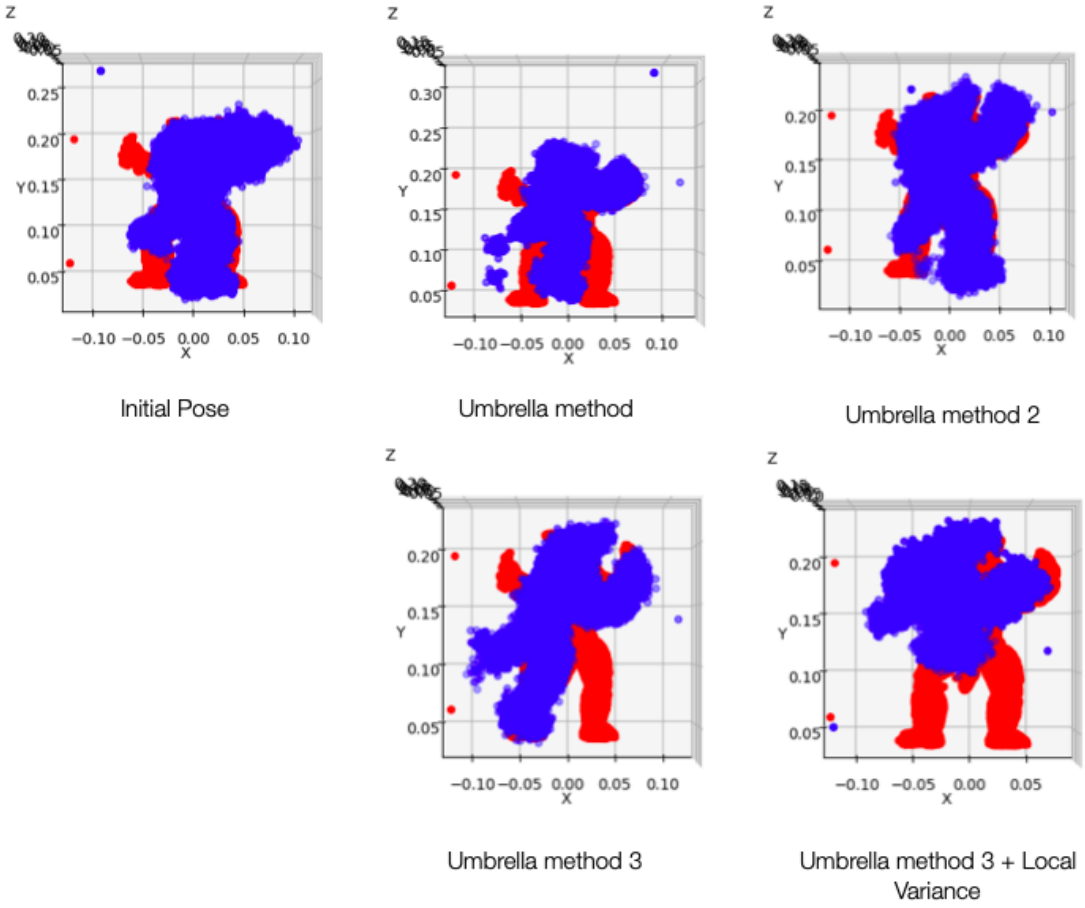



**Figure 14:** Test of Armadillo registration on the point cloud with noise and the different methods to deal with it.

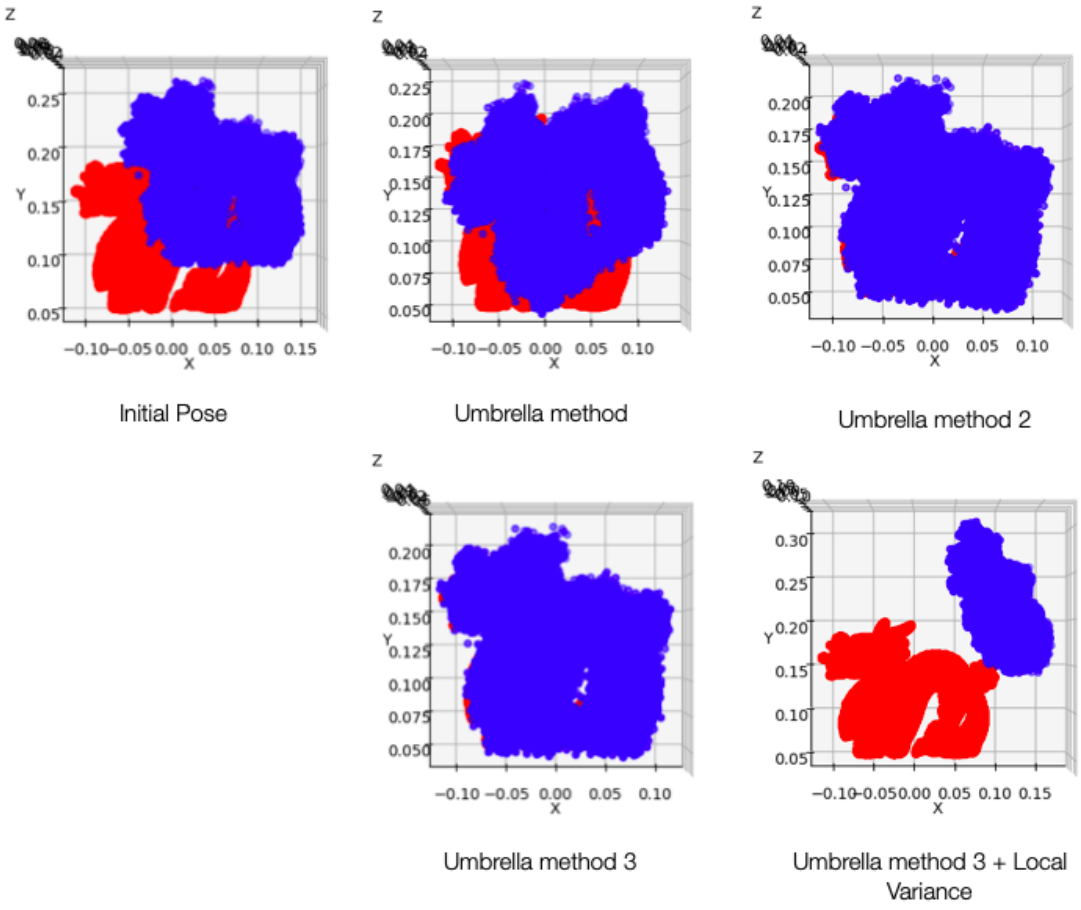



**Figure 15:** Test of Dragon registration on the point cloud with noise and the different methods to deal with it.