Real-Time 3D Position Reconstruction of the Hansen Connector

ING-INF/03, Advanced Digital Image Processing - 2023/24

Full Name	Student ID
Andrea Porri	109604

Professor: Alessandro Mecocci

UNIVERSITÀ DI SIENA 1240

Contents

1 | Abstract

This paper presents an automated system for reconstructing the three-dimensional position of the Hansen Connector using computer vision techniques and Iterative Closest Point (ICP) algorithms.The system is based on the acquisition of data from the Intel RealSense camera, which are processed through an advanced image segmentation algorithm. From this process, portions of the image emerge that are converted into 3D point clouds, to which global and local registrations based on ICP algorithms are applied along with the point cloud of a reference model. This allows for accurate alignment between the 'model' and 'target' point clouds, facilitating the three-dimensional identification of the object in the target point cloud.
This work focuses on the identification of the object in the target point cloud by exploiting the matching with the Hansen's Connector model, in order to derive its absolute position within the scene. The effectiveness of the approach was validated through a series of visual and quantitative evaluations, although conditioned by the multiple hardware issues encountered due to the specific problem under analysis and the available devices.
The automation of this process offers numerous advantages, including an increase in efficiency and a reduction in human involvement in risky tasks. In conclusion, this work aims to develop an automated system that exploits modern computer vision technologies and ICP algorithms to obtain a reconstruction of the absolute 3D position of the Hansen's connector, in real time, within the scene captured by the sensor, which is necessary for the robotic arm to perform its task correctly.

2 | Introduction

In the contemporary context, the importance of safety in the workplace is assuming an increasingly central role, emerging as an issue of primary importance and relevance especially in the most advanced companies. Unfortunately, every year, both in Italy and internationally, there are numerous accidents at work that are often attributable to safety systems that are inadequate to the specific characteristics of the work being carried out.
This project focuses on a particular work task, in which operators are called upon to handle tools for the introduction of high-pressure inflammable gas into special containers. This task entails inherent safety risks for the operators, who can be seriously injured if they do not perform the task properly.
The main objective is therefore to preserve the safety of the operators through the implementation of robotic arms to carry out this operation, allowing human personnel to play only a supervisory role in the operation and routine maintenance of the arms and devices.

3 | State of the art

This section discusses the technologies that made a fundamental contribution to the project's mission, most notably the YOLOv8-seg model[11] and the Iterative Closest Point (ICP) algorithm[14][9]. In particular, these methodologies played a key role in the segmentation of the connector and the alignment of a three-dimensional model of the Hansen connector to a target point cloud to accurately determine the absolute three-dimensional position of the object in the target reference system.

3.1 | Object Segmentation

Image segmentation is a crucial field of computer vision. Unlike simple object detection, where it is sufficient to return the co-ordinates of two points to identify a bounding box around the object; in segmentation, each pixel in the image is classified according to the region or object to which it belongs. In this context, YOLO (You Only Look Once) was a significant development for this field.

3.1.1 | YOLOv8-seg

YOLOv8-seg[11] uses a CNN-based architecture, and is a modified version of YOLOv8 that enables real-time detection and segmentation on images and videos. This model is a combination of YOLOv8[10] and YOLACT[5], where the first is used for object detection, while the second deals with semantic segmentation.
The model is available in five versions[11]: nano, small, medium, large and extra large. Each of these versions differs in the number of parameters and requirements needed for proper functioning. A 'lighter' model requires fewer parameters to be learnt, which leads to a reduction in accuracy in favour of higher processing speed and the possibility of using less expensive hardware. A proper analysis of the project objectives is always necessary in order to identify the best possible compromise between the model to be used and the hardware required.

3.1.2 | Comparison Architecture YOLOv8 and YOLOv8-Seg

Understanding the structure and the differences between the two architectures is essential in order to make efficient use of the YOLOv8-Seg model. To facilitate this understanding, an image containing the architecture of the YOLOv8-Seg model is shown below, followed by a brief explanation of the parts in which the two models differ.

3.1.2.1 | Model Architecture

The architecture of YOLOv8-seg, which is the same for each of the five variants, has a well-defined structure, which is shown below:

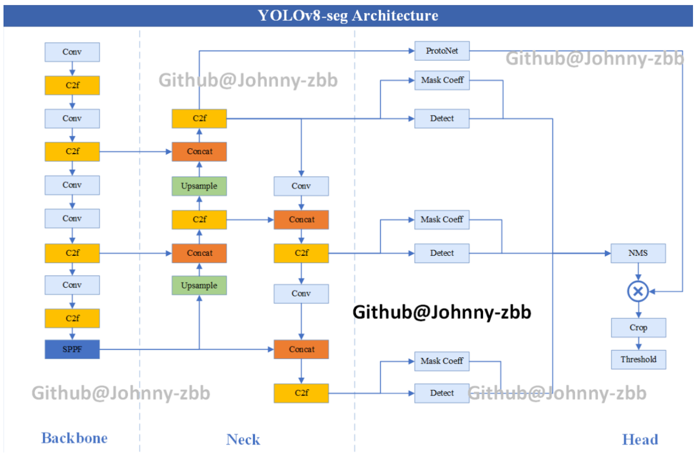


Figure 3.1: YOLOv8-seg Structure

3.1.2.2 | Component Analysis

As previously mentioned, YOLOv8-Seg extends the YOLOv8 framework by adding the ability to predict pixel-wise masks for object segmentation. The components of the segmenter framework are explained below, while highlighting the differences between the two architectures[13]:

- **Backbone**: The backbone is responsible for extracting features from the input image.
- **Neck**: The Neck of the model processes the characteristics extracted from the Backbone. It may include additional layers and mechanisms to refine the features.
- **Head**: The Head of the model is where the final predictions are made. For YOLOv8-Seg, there are usually two heads:
  - **Detection Head**: predicts bounding boxes and class probabilities. (YOLOv8 Head)
  - **Segmentation Head**: predicts pixel-wise classification masks to determine whether each pixel belongs to an object or not. (YOLACT Head)
- **Post-processing**: After the model has made its predictions, post-processing steps are applied to refine the results.

3.2 | Point Cloud Alignment

In computer vision, and robotics in general, the alignment of point clouds often plays a crucial role in finding objects (models) within three-dimensional environments. In the following, the specific algorithm used, the Iterative Closest Point (ICP)[14], is analysed.

3.2.1 | What is it?

The alignment of two point clouds is the process that determines the optimal geometric transformation. This transformation takes a point cloud from one coordinate system to another, so that the two point clouds are aligned with each other. This process is essential for applications such as object recognition in 3D environments, finding the absolute position of an object within a scene, building a 3D model using multiple point cloud alignments and more other applications.

3.2.2 | Specific applications

In this context, it was used for several purposes:

- 3D model construction of the Hansen Connector.
- Reconstruction of the absolute position of the object within a complex three-dimensional scene.

3.2.3 | Iterative Closest Point (ICP) Registration

The Iterative Closest Point (ICP) registration[14][9] is an iterative algorithm that attempts to approximate the best possible overlap between two point clouds through the iterative application of appropriate geometric transformations in such a way as to minimise the discrepancy between the two. The registration process begins with an initial estimate of a rigid transformation and proceeds iteratively to refine it until it converges to an optimal transformation. Then, in the context of geometric registration, the ICP is used as an algorithm for estimating the transformation required to align two point clouds: one source to one target. During each iteration, the ICP calculates the correspondence, and discrepancy, between pairs of points belonging to the two point clouds respectively, and uses this information to update the previous transformation. This process continues until one or more convergence criteria are met.
Several variants designed to suit specific situations are derived from the ICP; three of these variants were examined in the course of this project:

3.2.3.1 | ICP Point to Point Registration

The Point-to-Point ICP[14][9] method aims to find the best transformation that minimises the sum of the distances between the corresponding points of two distinct point clouds. In practice, it only exploits the positions of the points without using additional information.

3.2.3.2 | ICP Point to Plane Registration

In the Point-to-Plane ICP[14][9], the objective function minimises the distance between the points of the source point cloud and the planes of the target point cloud surfaces, exploiting the geometric information of the surfaces to improve alignment. This approach uses surface normals to better fit the source point cloud points to the surfaces of the target point cloud rather than to individual points; thus improving alignment accuracy in the presence of smooth or gradually changing surfaces.

3.2.3.3 | ICP Colored Point Cloud Registration

The Coloured ICP registration[12][7] extends the traditional ICP algorithm by integrating both geometric and photometric information to align coloured point clouds. The key idea is to optimise a joint objective that considers both colour and geometric data, improving the accuracy and robustness of the alignment. This approach is particularly useful for avoiding slips along planar surfaces and ensures that the alignment is accurate both along the normal direction and in the tangent plane of surfaces.

4 | Sensor Informations

The Intel® RealSense™ LiDAR Camera L515[6] represents a significant evolution in depth sensing technology. With compact dimensions, this sensor offers a capture depth ranging from 0.25 to 9 metres. Equipped with a 2MP RGB camera, an IMU and capable of capturing up to 30FPS at resolutions of 1024x768 (XGA) for depth and 1920x1080 (FHD) for RGB, this sensor offers high versatility. Furthermore, thanks to its LiDAR technology, which utilises an IR laser, the L515 offers accurate and detailed point depth. These features make it ideal for a wide range of applications requiring a deep understanding of three-dimensional space in large and not excessively bright environments.

5 | Proposed solution

In this project, an innovative method was developed to reconstruct the absolute position of the Hansen Connector within a three-dimensional space. The method detects and aligns a 3D model of the connector to a target point cloud, verifying the presence of the connector and determining its absolute three-dimensional position within the scene. This approach combines the YOLOv8-seg technologies and the ICP registrations technique, previously analysed in 'State of the Art'.
The steps of the method are listed and analysed below.

5.1 | Dataset Construction

In the initial phase, a segmented dataset of the Hansen's connector was created, specially formatted for fine-tuning the YOLOv8-seg pre-trained model. This process required the use of RoboFlow[3], a free platform that facilitates annotation and data preparation. Thanks to this tool, it was possible to efficiently label the images according to the format required by YOLOv8, providing precise information on the positions and dimensions of the Hansen connector within the images.
The original 250 images are in RGB format with a resolution of 1920x1080. To improve the diversity and robustness of the dataset, various augmentation techniques offered by RoboFlow were applied, allowing the number of images to be increased from 250 to 658.
Augmentation and preprocessing characteristics used for dataset creation:

1. Preprocessing: Resize to 640x640.
2. Augmentations: All the options offered by RoboFlow(free version).

Finally, the process of collecting images, creating and labelling the dataset took a total of about 4 days. Subsequently, thanks to the functionality provided by the platform, the dataset was divided as follows:

- Training set: 612 images
- Valid set: 38 images
- Test set: 8 images

5.2 | Train YOLOv8-seg

To conduct the training of YOLOv8-seg models, I relied on one of the machines in VisLab 205 where the Ultralytics library, and the necessary environment, was already present and installed. Then I had to download from the official Ultralytics website the multiple variants of YOLOv8-seg with their respective parameters and hyperparameters. At this point, by using the command line from the terminal and making small changes to the YAML configuration file present in the model folders, I easily conducted training phase on the already preparred training set.
During this phase I ran several trainings, differentiating them by variant type, number of epochs and mini-batch size.

5.2.1 | Configurations of YOLOv8-seg models

The selection of the YOLOv8-seg variant, mini-batch size and number of epochs, for this project was made taking into account several factors, including specific performance requirements, available resources and minimum required accuracy.

5.2.1.1 | YOLOv8-seg Variants

Three variants of the five available were trained and tested: Nano, Small and Medium.

5.2.1.2 | Hyper-parameters and Parameters

The characteristics of the training conducted are listed below, with an indication of the various specifications or where these can be consulted:

- **Parameters**: The parameters/weights with which the various models were initialised are those provided by Ultralytics[11].
- **Hyper-parameters**: Hyper-parameters can be found in the documentation of Ultralytics[11], or can be viewed (or edited before training) in the file *args.yaml* in the model download folder.
  The only changes you made in the YAML file were:
  - Epochs: 100, 250, 1000, 10000
  - Batch size: 8, 16, 32, 64

5.3 | Point clouds and 3D model of the Hansen Connector Construction

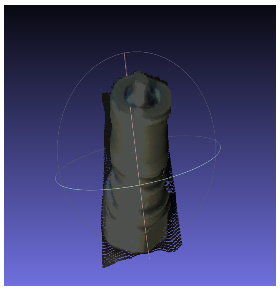
5.3.1 | Point cloud Acquisition

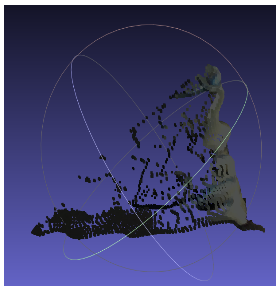
In the first script, a method was developed which, by means of video acquisition using a RealSense camera, performs the segmentation of the connector and subsequently extracts the colored point cloud. The code is structured schematically, with the main steps including:

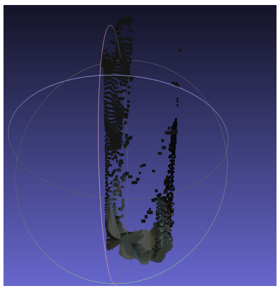
1. Loading the contents of the dedicated YAML configuration file;
2. Realsense sensor setting: minimum distance 25cm (default 49cm) and maximum depth filter at 70cm;
3. YOLO model initialisation;
4. Initialisation of RGB frame acquisition pipeline and depth;
5. ∀ frame:
   - [a] Acquisition and alignment of depth frame to RGB frame;
   - [b] Preprocess of frame;
   - [c] Object segmentation;
   - [d] ∀ detection:
     - i. Extraction of the detection box;
     - ii. Updating the segmentation mask (binary mask[4]);
     - iii. ∀ pixel ∈ detection box: calculates 3D coordinates using the position (x,y) of the pixel and the corresponding depth;
     - iv. Save the newly created point cloud in a specific file;
     - v. Plot Real-Time.

Real-time plots show the results of segmentation and other useful information for a correct understanding of the scene. These are displayed in real time using OpenCV, showing the segmented RGB and depth frames, the segmentation mask and the overlap between the RGB and depth frames.

An example of the point clouds obtained is shown below:


(a) Frontal view


(b) Side view


(c) Top view

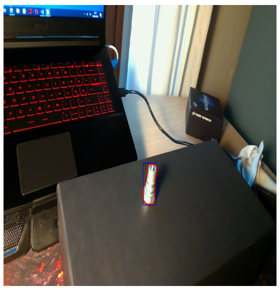

(d) RGB Detection Box

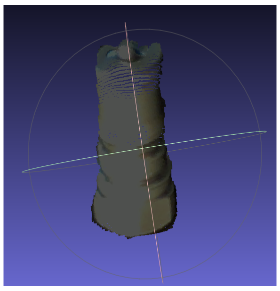
Figure 5.1: Good quality point cloud

Output: Folder containing the point clouds of the Hansen connector faces detected by the segmenter.

5.3.2 | Hand finishing a point cloud (Recommended)

After acquiring several point clouds, it was observed that the construction of the 3D coordinates of all the points within the box also included acquiring background points not relevant to the object. Therefore, I decided to visually check the best point clouds in my possession and, after selecting three of them, cleaning them and obtaining representative models of the face of my connector.
One of these was used as a 'reference model' for the construction of the entire Hansen's connector model. This refinement was obtained using MeshLab[1], following a tutorial on YouTube[2].

Below are pictures illustrating the result obtained:


(a) Frontal view

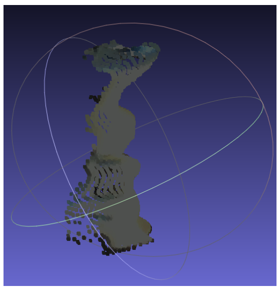

(b) Side view

Figure 5.2: Point cloud finished - Model 1

Output: 2-3 models of the Hansen connector face.

5.3.3 | Model construction: Solution via software alignment

To reconstruct the absolute position of the object, as mentioned above, I will still need a 3D model of the entire connector. It follows that having reconstructed a partial model of the connector is not sufficient for the ultimate purpose of the project. Consequently, in the second script I have developed a method that implements a 3D point cloud registration and alignment process using the Open3D library. In short, the basic model, obtained in the 5.3.2 section, and a series of target point clouds that have not been previously cleaned up are provided. To each of these, the model is iteratively aligned and integrated, which in turn will become the starting model for the next iteration, thus consuming target point clouds and creating my 3D object. The main functions of the code are described below:

1. **global_registration**:
   Performs a global registration[15][8] between two point clouds using the RANSAC (Random Sample Consensus)[16] convergence criterion. It uses this algorithm, based on FPFH(Fast Point Feature Histograms) matching, to find the optimal transformation that aligns the source point cloud (source) with the target point cloud (target).
   The function returns the alignment found.
2. **single_alignmentlabel**:
   Performs an alignment between a source (model) point cloud and the first point cloud (target) in an input folder. Use the global registration1 to find the optimal transformation, then apply this transformation and save the result if approved. So in practice:
   - Find the transformation that aligns a source point cloud with the first target point cloud found in an input folder;
   - Apply the obtained transformation to the new source point cloud;
   - If the user approves, by typing a specific approval input after visual check, it merges the two clouds (transformed source + target) creating the new source point cloud;
   - Save the new model (source) in an output folder and update the counter.

After implementing the above-mentioned functions, they were integrated into the main code:

1. Loads configuration parameters from a YAML file;
2. Prepare input and output folders, checking and cleaning invalid files;
3. Cycle until there are no more target point clouds in the folder:
   [a] Upload the most up-to-date model (source);
   [b] Extracts the first target point cloud from the folder;
   [c] Performs the alignment process of source and target point clouds using single_alignment2;
   [d] Prompts the user to confirm the saving of the new partial model obtained;
      i. If accepted, the model is saved in the appropriate folder together with the other models, and will be the most up-to-date;
      ii. Otherwise, it is discarded and the one in use remains the most up-to-date model;
   [e] Delete the target point cloud from the input folder and update the counter.

Output: Folder containing the models created step by step. The last one is the model of the entire connector.

5.4 | Real time 3D Absolute Position Reconstruction

The third script provides a method that, in addition to performing what was done in 5.3.1, attempts to perform matching between the point cloud obtained and the connector model obtained in 5.3.3. The objective is to align the model to the portion of the connector in the target scene, so as to accurately reconstruct and extract the absolute position of the connector with respect to the entire scene. The code is structured schematically and includes several functions for plotting registration, one for point cloud preprocessing, and finally two functions that perform local and global registration.
The two most important functions, and which deserve further study, are global and local registration:

1. **global_registration**:
   The function is the same as that analysed in 5.3.3 at 1.
2. **local_registration**:
   Performs a local alignment between source(model)-target point clouds using one of several variants of the ICP registration algorithm at the user's choice. This function is used after the application of the global_registration to refine this initial alignment.
   To execute the code, first the variant to be used is chosen by the user and then, regardless of the choice made, the following steps are performed:
   - Set the distance threshold;
   - Find and apply the transformation for the alignment of the source to the target point cloud;
   - Print and display the result;
   - Returns the transformation found and applies it to the origin point cloud, which will be none other than the absolute position of the connector.

The code can be divided into two sections, where for each iteration (detection), I perform the following main steps:

- **Section 1**:
  Acquisition of the target point cloud using 5.3.1.
  Difference: point cloud acquisition is not continuous but user-selected by pressing 's'.
- **Section 2**:
  Continuing the code in the first section, for each point cloud, it performs the following steps:
  1. Loading the source(model) and target point cloud;
  2. Performs an initial check on the correct loading of point clouds, via a rigid registration (optional);
  3. Preprocessing of the two point clouds (normals and FPFHs calculation);
  4. Performs a global registration1 and plots;
  5. Executes a local registration2 initialised with the global transformation, obtained in the previous step, and plots;
  6. Prints the graphs and displays the point cloud containing the absolute position.

The global registration function was necessary because it does not require an initial input transformation, thus allowing a transformation to be obtained that aligns, even roughly, the two objects. This step is essential to provide an optimal initial transformation for later use in local registration, which, unlike global registration, requires an initial transformation to function properly.
In essence, registration techniques are used to determine a transformation to be applied to the source point cloud (model) in order to obtain the best possible alignment with the target point cloud. Subsequently, by extracting the transformed point cloud of the model, it is possible to obtain the absolute position of the object within the scene in which the inference was made.

6 | Results and Analysis

In this section, all issues and results that emerged in the various phases of the method just described are analysed in a schematic and sequential manner. In addition, further relevant empirical observations will be provided during this analysis in order to highlight the strengths and weaknesses of the method more clearly.

6.1 | Sensor

Drawbacks

The main problem with using this type of sensor is that depth measurements taken at distances of less than 25 cm are unreliable, producing elongated point clouds. Furthermore, for such small objects, the sensing distance greatly affects the accuracy of the measurements, generating unreliable and inaccurate point clouds. In addition, it is extremely sensitive to light, which makes it unsuitable for excessively bright environments. These characteristics make it unsuitable for the purpose of this project.

Advantages
The quality of the RGB camera, although not very high, is sufficient to obtain decent results during the inference phase with YOLOv8-seg.

6.2 | Dataset

Problems and Solutions
The size of the training and validation dataset is very limited and not sufficient for the purpose for which they were created, the segmentation. It is therefore necessary to increase the dimensionality and robustness of the dataset by adding more images, trying to include many close-up photos of the object (0-10 cm) and others taken at distances greater than 50 cm; this is because the current dataset contains very few photos belonging to these ranges.

Advantages
The dataset is adequate for training a YOLOv8-seg model, allowing it to detect the connector with good accuracy (the model detect the box well, but segments the connector poorly). Furthermore, it is easily extended by taking advantage of the paid versions of RoboFlow.

6.3 | Training YOLOv8-seg

Results
The training results were encouraging, especially for the small and medium versions, with minibatch size 32 and at least 1000 epochs. In cases with a large number of epochs, training stopped after about 250 epochs, according to the stopping criterion established by Ultralytics. The medium model, consisting of 27 million parameters, produced promising results considering the following validation metrics: Accuracy (Box P and Mask P): 0.973, Recall (Box R and Mask R): 0.974.
Other results of the training phase evaluation metrics were: Box Loss: 0.5545, Class Loss: 0.4001, DFL Loss (Distribution Focal Loss): 0.9138, Segmentation Loss: 1.007.

Problems and Solutions
The nano version produced very poor results, as well as the small and medium versions with minibatch sizes 8 and 64. It was not possible to test the large and extra large versions due to their excessive weight, which could be solved by using more big and powerful GPUs. Furthermore, the segmenter does not work optimally, as it does not segment the object correctly; however, the detection box is always well positioned and accurate. The detector fails to detect the object correctly in the ranges 0-10 cm and over 50 cm. To solve the latter two problems, it will be necessary to increase the training and validation datasets to several thousand photos, introducing additional photos with the previously expressed characteristics.

Advantages
The model used is lightweight and easily implemented, being suitable for the required task, albeit with unsatisfactory results for the reasons listed. Thanks to Transfer Learning, it is possible to reuse the model already trained and made available by Ultralytics, requiring only fine-tuning, which is less onerous in terms of time and cost. Moreover, YOLO remains the state-of-the-art for object detection and segmentation. A well-done segmenter can make the removal of outliers in point clouds unnecessary.

6.4 | Point clouds and 3D model of the Hansen Connector Construction

6.4.1 | Point cloud Acquisition

Results
The results obtained from this capture are promising. Despite the problem related to the sensor6.1 used, the object is captured correctly with the associated RGB colours. The function that performs this operation is easily modified in case further information needs to be saved in the ply file.

Problems and Solutions
The point clouds created not only contain the connector, but also include everything within the detection box generated by YOLO; as a result, the point clouds produced will be 'dirty'. This can be solved by using a

YOLOv8-seg segmenter trained on a significantly larger dataset 6.2 and with optimised hyper-parameters. Ideally, the segmenter, at this point, will be able to segment the object optimally, allowing the point cloud of only the pixels within the segmentation mask to be extracted. This solution will make it possible to keep only the object of interest and eliminate the rest.
The point clouds currently created are very inaccurate due to the sensor used. The solution is to adopt a more precise technology suitable for the purpose.

6.4.2 | Hand finishing a point cloud

Results
The process of manually finishing a point cloud has produced excellent results, although it is laborious and takes a considerable amount of time. However, it is an indispensable step to remove the background from the connector face models. This process will become superfluous with the implementation of a more precise segmenter.

Advantages
Manual finishing allows point clouds to be accurately analysed and verified using advanced software such as, for example, MeshLab. The advantage of this is to obtain models cleaned of points not belonging to the connector.

6.4.3 | Model construction: Solution via software alignment

Results
The results obtained are encouraging, although probably not very suitable for the specific case of the Hansen connector. This software aligns multiple point clouds of connector portions in order to reconstruct the entire object. However, the object in question presents significant symmetry and the point clouds are rather imprecise, with the result that the alignment produced is of poor quality both qualitatively (through, for example, number of vertices in the correspondence_set) and visually. Furthermore, it was observed that only half of the object is actually reconstructed.
Below is an example of an alignment produced from a face model and two 'dirty' point clouds:

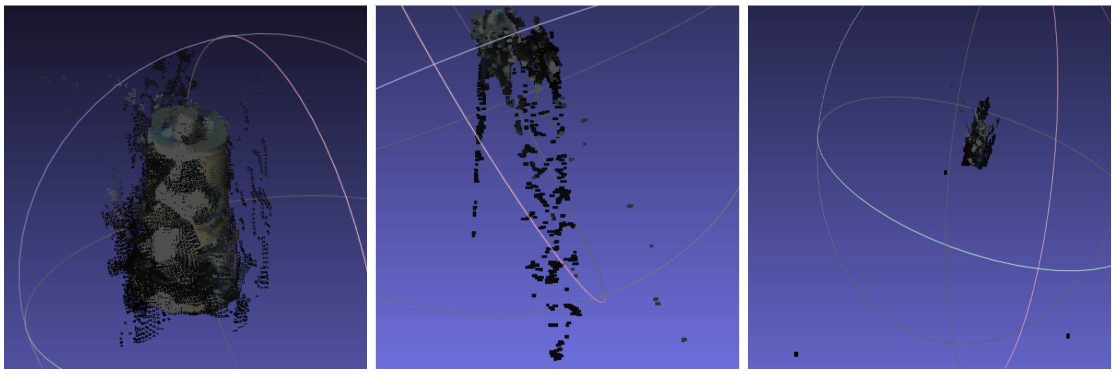

(a) Frontal vision (b) Top view (c) View points of the point clouds

Figure 6.1: Partial model of the Hansen connector

An attempt at manual alignment was also made by following a YouTube tutorial for MeshLab[17], but was unsuccessful (returns error) due to the lack of similarity and imprecision of the point clouds.

Problems and Solutions
The new point cloud produced will be very heavy due to the merging of the points of the various point clouds. Therefore, one solution to be able to 'lighten' it will be to carry out post-processing by downsampling, so that it can be used effectively.
In addition, in order to test the robustness of this method, I tried to introduce a kind of 'asymmetry' (by adding a shirt button in front of the connector) and see if the method could create the whole object. However, I found no significant improvement, again due to the inaccuracy of the point cloud. It would be good to perform this test using other sensors and especially to test it on objects that already have asymmetry by nature. A further problem could be when the global registration fails, but since the results are viewed before any merging, the problem is solved by the nature of the code.

Advantages
Being a software construction, it is very fast and efficient and can handle a considerable number of point cloud inputs. Furthermore, thanks to the choice functionality, it is possible to discard new partial models that are not considered valid.

6.5 | Point CLoud Alignment - Real Time 3D Position Reconstruction

Results

The results obtained are extremely positive, considering the poor quality of the point clouds (6.4.1 and 5.3.1) acquired during the execution of the process and the connector models created (6.4.3 and 5.3.3). A quantitative analysis, using specific metrics such as correspondence_set, fitness, inlier_rmse, etc., would be possible, but given the nature of the issues previously mentioned, would not be indicative of the robustness of the method. However, from an intuitive and visual point of view, the results obtained are, in most cases, acceptable and very promising already in the current situation.

Problems and Solutions

In addition to the problems inherent in point clouds 6.4.1, one of the main problems encountered at this stage occurs when the global alignment fails. This leads to a situation where the two objects are significantly misaligned or lack any intersection. This issue affects the local alignment, which fails to refine the initial global alignment because it requires a preliminary transformation that intersects and aligns the two objects at least partially. Consequently, if global alignment fails, local registration will also fail. When objects are very misaligned, local registration often fails, whereas if the two objects have no intersection, local registration will inevitably fail. Failure means that the transformation produced fails to align, even roughly, the objects.
Two possible solutions to this problem are:

1. Cascade, prior to local registration, another global alignment using a different algorithm from RANSAC. The idea is to activate this only when correspondence_set, produced by RANSAC, has few veritics in it and consequently the two objects are misaligned or detached. It must be considered that this will in any case burden and slow down the method, so careful evaluations must be made.
2. Set an adaptive minimum threshold, beyond which to discard that alignment and move on to the next frame, based on the correspondence_set and the number of vertices present in the target point cloud. For example, if the number of vertices in the correspondence_set is less than 70% of the vertices present in the target point cloud, discard the frame. This will allow the pipeline to be maintained and not burdened further.

Another issue concerns the coloured ICP, which also uses colour information to perform the alignment. This means that if two points are close to each other but have even slight colour differences (due to brightness, shadows, reflections, etc.), matching will not take place. During the tests, it was not possible to run the coloured ICP because the colours of the reference model differed slightly from those of the target point cloud, preventing alignment. It can be concluded that the model must be created with these variables in mind in order to function correctly. In my experience, and for the objective of this project, the Point-to-Point and Point-to-Plane (the best) methods are sufficient and much more suitable. In theory, the ICP is more precise, but also very problematic and sensitive to external factors.

Advantages

Using this type of alignment allows the connector within the target point cloud to be identified automatically and quickly. Furthermore, as the model is aligned to the target point cloud, it is possible to extract the absolute coordinates of the entire 3D object using the coordinates of the model after the newly found transformation has been applied to it.

7 | Conclusion and Future Development

In conclusion, the method is very interesting and suitable for the task of reconstructing the absolute position in three-dimensions, according to my findings. As far as the creation of the model is involved, the technique developed is intriguing but not perfectly suited to the purpose of the project due to the symmetry of the object, but further analysis will be necessary to be certain. Furthermore, in the future, it would be interesting to explore other methods for constructing the model or to rely on 3D modelling programmes.
For future developments, it would be useful to implement and test the solutions proposed in 6, to increase the robustness of the method. In addition, it would be interesting to better evaluate the coloured ICP model, using a model created with point clouds acquired in the context and illumination to which the robotic arm station will be subjected. Other possible tests include analysing the other two ICP variants, testing different convergence criteria and other algorithms to be used in the global registration. In order to speed up the system, it could be useful to perform the analysis every 8-10 frames, thus reducing the workload while still maintaining a high degree of accuracy given the not-so-high speed of movement the connector will be subjected to by the robotic arm. Furthermore, it may be necessary to run these processes on the GPU to further increase execution speed. A preprocess with outliers removal (by statistical method or by distance) and a downsampling to reduce the number of vertices present in a target point cloud could further improve the efficiency of the system.

8 | References

[1] Meshlab. https://www.meshlab.net/.

[2] Meshlab tutorial youtube. https://www.youtube.com/@MrPMeshLabTutorials.

[3] Roboflow. https://app.roboflow.com/.

[4] How to create a binary mask from a yolo8 segmentation result. https://stackoverflow.com/questions/76168470/how-to-create-a-binary-mask-from-a-yolo8-segmentation-result, 2023.

[5] Fanyi Xiao Yong Jae Lee Daniel Bolya, Chong Zhou. Yolact: Real-time instance segmentation. https://openaccess.thecvf.com/content_ICCV_2019/html/Bolya_YOLACT_Real-Time_Instance_Segmentation_ICCV_2019_paper.html, 2019.

[6] Intel Realsense documentation. Lidar camera l515 datasheet. https://dev.intelrealsense.com/docs/lidar-camera-l515-datasheet.

[7] Open3D 0.18.0 documentation. Colored point cloud registration. https://www.open3d.org/docs/release/tutorial/pipelines/colored_pointcloud_registration.html.

[8] Open3D 0.18.0 documentation. Global registration. https://www.open3d.org/docs/0.7.0/tutorial/Advanced/global_registration.html.

[9] Open3D 0.18.0 documentation. Icp registration. https://www.open3d.org/docs/release/tutorial/t_pipelines/t_icp_registration.html#ICP-registration.

[10] Ultralytics documentation. Ultralytics yolov8. https://docs.ultralytics.com/#where-to-start.

[11] Ultralytics documentation. Ultralytics yolov8 docs - tasks: Segment. https://docs.ultralytics.com/tasks/segment/.

[12] Vladlen Koltun Jaesik Park, Qian-Yi Zhou. Colored point cloud registration revisited. https://paperswithcode.com/paper/colored-point-cloud-registration-revisited, 2017.

[13] YOLOv8-Seg model structure Contributorss. Architecture differences between yolov8 and yolov8-seg. https://github.com/ultralytics/ultralytics/issues/1710, 2023.

[14] Yanxia Wang Wuyong Tao Peng Li, mRuisheng Wang. Evaluation of the icp algorithm in 3d point cloud registration. https://ieeexplore.ieee.org/abstract/document/9060927, 2020.

[15] Vladlen Koltun Qian-Yi Zhou, Jaesik Park. Fast global registration. https://link.springer.com/chapter/10.1007/978-3-319-46475-6_47, 2016.

[16] Marc Pollefeys Rahul Raguram, Jan-Michael Frahm. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. https://link.springer.com/chapter/10.1007/978-3-540-88688-4_37, 2008.

[17] Mister P. MeshLab Tutorials. 3d scanning: Alignment advanced 1. https://www.youtube.com/watch?v=jAXAxQvX8Cc.