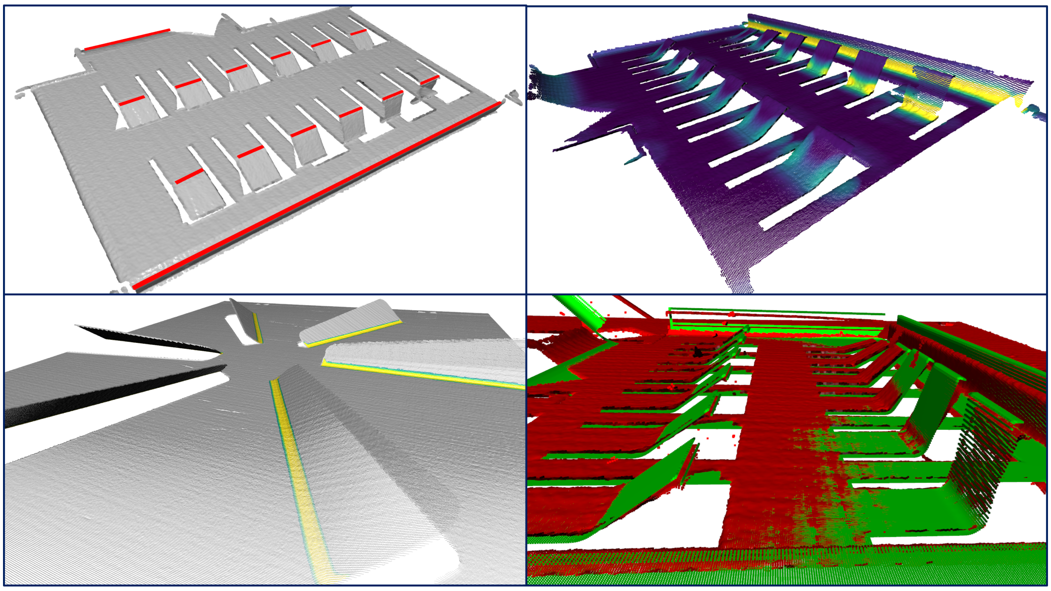
Computer-Vision Based Bend Quality Inspection for Sheet Metal Parts



Master Thesis
Christos Kantas
Computer Engineering, AI Vision and Sound
Aalborg University
June 4, 2025

AALBORG UNIVERSITY
STUDENT REPORT

Electronics and IT
Aalborg University
http://www.aau.dk

Title:
Computer-Vision Based Bend Quality Inspection for Sheet Metal Parts

Theme:
Scientific Theme

Project Period:
Fall 2024 - Spring 2025

Project Group:
-

Participant(s):
Christos Kantas

Supervisor(s):
Rikke Gade
Morten Kristiansen

Copies: 1

Page Numbers: 61

Date of Completion:
June 4, 2025

Abstract:

In this work, the research question of how to inspect the quality of bends in 3D point cloud representations of sheet metal parts is investigated. The manufacturing system responsible for fabricating the sheet metal parts is introduced, alongside the currently available scanned data. A novel pipeline is developed, which automatically preprocesses the scanned point clouds, triangulates the positions of the bends and measures their total inclination angle, as well as their arc length. The method is tested on ideal data, which proved that it is conceptually sound. Subsequent experiments are conducted to test the method's robustness to noise, as well as its ability to provide realistic results on the real scans.

Preface

All code developed in this work can be found in:

`https://github.com/ckantas/PointClouds.git`. Aalborg University, June 4, 2025

C.Kantas

Christos Kantas
<ckanta18@student.aau.dk>

Contents

Chapter 1

Introduction

From the hood of a car to the frame of a plane, or the inside of a microwave, sheet metal parts are used as core components. But how can manufacturers verify that sheet metal parts meet the required quality standards for their downstream application? In mass production settings, a sample is typically taken for every n manufactured parts, and trained operators use Coordinate Measuring Machines (CMMs) to inspect various features, such as cutouts and bends. While this can be a viable solution in these high-volume scenarios, what about a smaller but perhaps more flexible manufacturing system that can produce a variety of different sheet metal parts?

As it turns out, such a system has recently been developed in the Materials and Production department in Aalborg University. This system is built around a high-power laser, capable of performing tasks such laser cutting, forming, welding and engraving. While it has been demonstrated in Nikolov et al. [1] how the manufacturing of a sheet metal part can be fully automated using this system, the quality of these fabricated parts can not be automatically assessed by the system. However, a measurement scanner, capable of generating 3D point cloud representations of the manufactured parts, is already integrated into the manufacturing system. Can the quality of the manufactured sheet metal parts be automatically assessed using this scanner? This is effectively the research question that will be investigated in this project.

More specifically, since a pipeline has already been developed for detecting cutouts/holes in the 3D point clouds [2], this project will tackle the problem of detecting and measuring *bends* in the material, which are created through the process of laser forming. For a given sheet metal part, a bend that connects two surfaces can be described by two quantities; 1. the total inclination angle of the two connected surfaces, and 2. its arc length, meaning the distance along the bend's curve that is required for the material to obtain the previously mentioned total angle. Based on this information, the following initial problem statement can be formulated:

How can the total inclination angle, as well as the arc length of all bends be measured, when provided with a 3D point cloud that represents a sheet metal part?

In order to get an understanding of the problem, as well as inspiration about its possible solution, the following section provides an overview of related literature.

1.1 Related Works

The two most relevant research fields to the problem at hand are curvature analysis and sharp feature detection. Even though bends are not necessarily sharp features, in order to fabricate them, cutouts around the bent surface are made. Detecting and tracking these cutouts through sharp feature detection can allow for the measurement of the arc length of a bend.

The field of curvature analysis focuses on identifying how a surface bends locally by estimating properties like normals (the direction a surface is facing) and curvatures (how much it bends). A well-known method in this domain was proposed by Kalogerakis et al. in 2008 [3], where the goal is to extract what are called lines of curvature. These lines are smooth paths that are formed along directions where the surface bends the most or the least, describing identifiable structures in the point clouds. Their method begins by estimating surface normals and then calculating a mathematical object called the curvature tensor, contains information about the change of the local surface normals across different directions. To trace these curvature lines, their method takes small steps along the directions where the surface bends most sharply, as defined by the curvature tensor. A similar method is developed in this work to follow the direction of a bend in the point clouds and measure their arc length.

A more recent development in curvature analysis is presented in the work of Guerrero et al., who introduced PCPNet [4]. Inspired by PointNet [5], PCPNet is trained to predict per-point geometric descriptors, such as surface normals, principal curvatures, and a set of probabilities, for example that a point lies on an edge or a corner. PCPNet is trained on a large set of synthetic surfaces, and it can generalize fairly well to real-world point clouds that contain noise or irregular sampling. Perhaps PCPNet can be used to identify the regions in the scans that contain bends, as points in those regions will share distinct geometric properties, such as principal curvatures, which are the directions in which local surface normals change.

In the domain of sharp feature detection, one of the commonly cited work is Weber et al. (2010)[6]. The method in the paper begins by constructing a neighborhood graph for a given point cloud, where each point is connected to its k nearest neighbors. These local connections are then used to form triangles between the point and its neighbors. The surface normals of these are then plotted on a Gauss map, which is essentially a sphere that holds all the normal directions. The distribution of these normals in the Gauss map reveals the underlying surface structure. If the normals form several distinct clusters, this suggests that the point lies near a sharp feature, such as an edge or corner, as opposed to a smooth surface, where the normals would be spread more evenly.

In more recent work, the availability of large synthetic datasets like the ABC dataset [7] has made it possible to train data-driven models for sharp feature detection. One such method, proposed in DEF [8], first converts the point cloud into a mesh and then renders it from multiple virtual viewpoints to generate depth images. A ResNet-152 model [9] is then trained to predict, for each pixel in the image, the points distance to a sharp feature. The predictions of ResNet are then fused across multiple views and mapped back onto the original 3D point cloud. In the resulting point cloud, each point is assigned a distance-to-feature value, which can be used to reconstruct the sharp curve structures like parametric curves.

1.2 Outline

The rest of this thesis is organized as follows:

- Chapter 2 presents the scanning set-up and the scans used for development and testing;
- Chapter 3 gives a high-level overview of the proposed approach and explains the theoretical building blocks;
- Chapter 4 walks through each step in the proposed solution;
- Chapter 5 evaluates the approach on both synthetically generated and real scans;
- Chapter 6 reflects on the approach and results, and then concludes.

Chapter 2

Manufacturing System & Scan Data

In this chapter, a further intuition about the problem will be given by first introducing the laser manufacturing system that creates the sheet metal parts. Afterwards, the technique by which the parts are scanned will be explained, and the expected scan noise will be analyzed. The available dataset, consisting of three distinct sheet metal parts, each containing multiple bends will then be presented. Finally, a small discussion on the different origins of errors from pre-manufacturing to post-manufacturing will be made.

2.1 Laser Manufacturing System

In this section, the manufacturing system responsible for fabricating the sheet metal parts inspected in this work will be briefly analyzed. The setup can be seen in Figure 2.1 below, where most components are visible. The main component of the system is the YLS-3000 laser source [10], which is used for all manufacturing processes, namely laser cutting, laser forming, welding, as well as engraving. This laser is guided through an optical fiber, and its orientation as an end-effector is controlled by the ARGES Fibre Elephant 50 galvanometric laser scanner (GLS) [11], which uses mirrors to redirect the laser beam.

The GLS system is mounted on a KUKA KR 120 R2700 industrial manipulator [12], which allows for larger workspace than the GLS system can provide. A Wenglor MLWL 153 2D laser line scanner [13] is also mounted on the manipulator. This scanner is responsible for the acquisition of the 3D point clouds that will be analyzed in this project. All components are coordinated though a Beckhoff C6920-0060 industrial PC [14]. A list of these components, with a few additional details on each one can be found in Table 2.1 below.

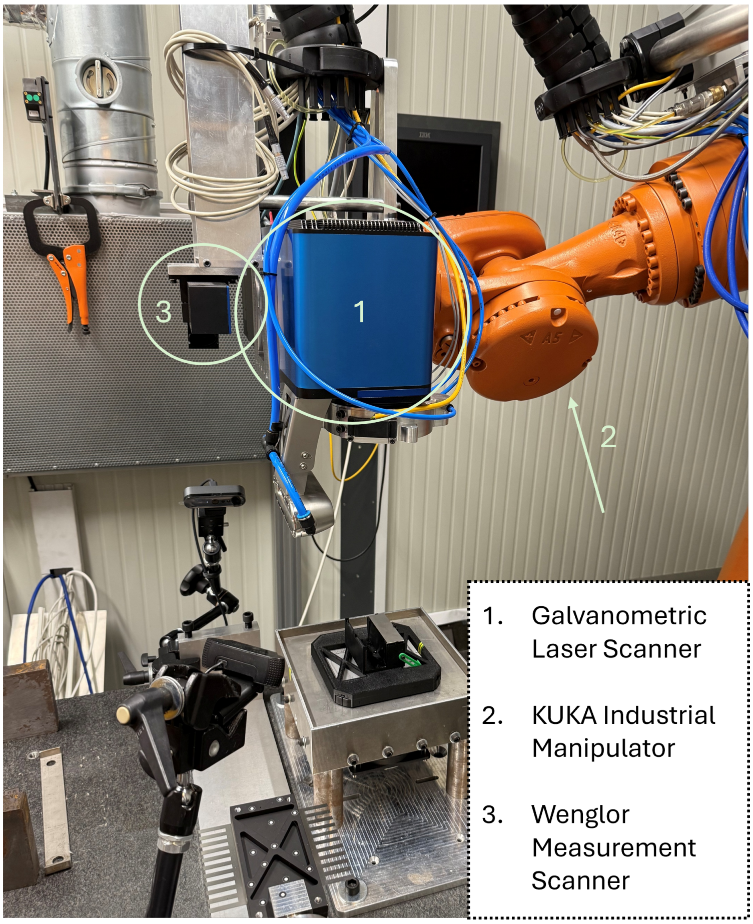

3
1
2
1. Galvanometric Laser Scanner
2. KUKA Industrial Manipulator
3. Wenglor Measurement Scanner

Figure 2.1: The Laser Manufacturing Setup. Three of its key components are highlighted.

Component	**Description**
Laser Source	3 kW IPG YLS-3000 single-mode ytterbium fiber laser. Ytterbium is a rare-earth element inside the optical fiber that amplifies the intensity of the laser through stimulated emission [15].
Galvanometric Laser Scanner (GLS)	ARGES Fibre Elephant 50 Galvanometric Laser Scanner. The laser beam passes through this component. Here, the the laser can be redirected, as well as focused and de-focused. This works mainly through a set a galvanometric mirrors and a system called DFM (Dynamic Focus Module).
Measurement Scanner	Wenglor MLWL 153 2D laser line scanner used for calibration [16], as well as to generate the point clouds of the manufactured parts.
Manipulator	KUKA KR 120 R2700 industrial robot arm. The manipulator increases the degrees of freedom of the GLS system, but it also allows for the 2D Wenglor line scanner to acquire 3D data, by setting it in motion about the Y axis.
Controller	Beckhoff C6920-0060 industrial PC responsible for synchronizing and controlling all hardware.

Table 2.1: Components of the laser manufacturing system.

In Nikolov et al. [1], it is demonstrated how the different processes, such as laser cutting and laser forming, can be coordinated to automatically manufacture a sheet metal part, when provided with its CAD model.

2.2 Scanning and Expected Noise

The 3D point clouds that will be inspected in this work are generated by the Wenglor 2D line scanner. The scanner emits a laser along the X axis and records the amount of reflected light in each position. The distance to the scanned object is measured through time-of-flight. The sheet metal part is placed on a stationary fixture, so in order to capture the Y axis, the KUKA manipulator sets the line scanner in motion along that direction. Parts are only scanned from one side at a time. The following figure shows a close up look of a flat region in an acquired 3D point cloud.

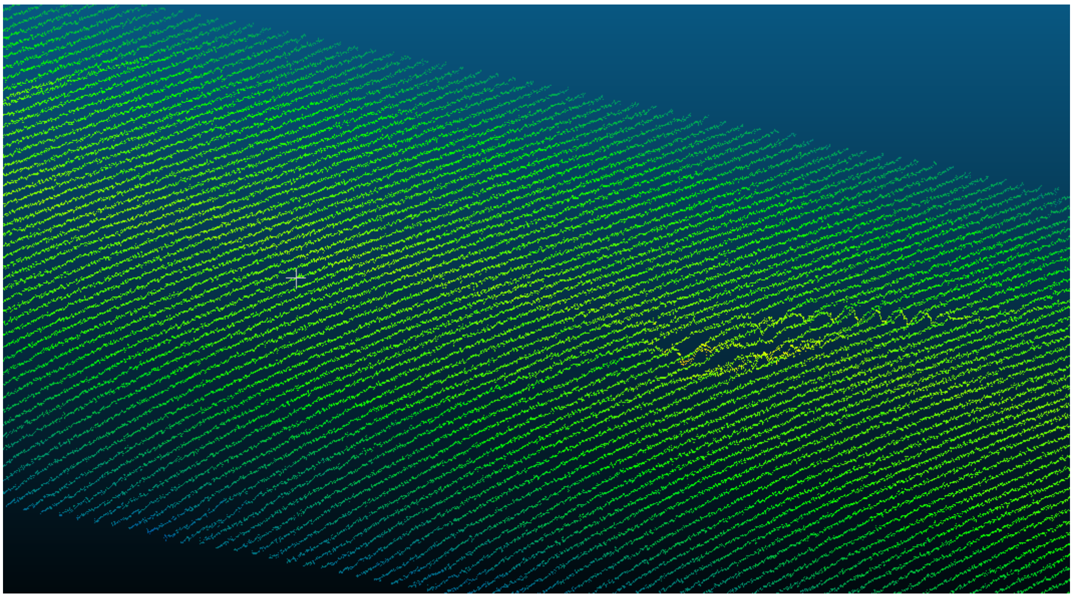


Figure 2.2: A close up of a flat region in the 3D scan of one of the manufactured parts.

Since the parts are only scanned from one side, the point clouds have a thickness of exactly one point. As it can also be seen in the above figure, noise will inevitably exist in the point clouds. This noise can be attributed to a variety of sources, including:

1. The material itself. Scratches or foreign entities in the surface of the material will create local inconsistencies in the scan. The material used for manufacturing is AISI 304 stainless steel, which is highly reflective. Reflective surfaces are notoriously difficult to scan, as even slightly incorrect specular reflections lead to almost no light returning to the sensor.

2. The manipulator. The manipulator moves the line scanner along the Y axis. Each joint of the manipulator has its own positional uncertainty. Additionally, depending on its joint configuration, the manipulator can slightly tremble along the Z axis, which will cause noise along the Z coordinates in the scan.

3. The Wenglor scanner comes with its own uncertainties. Based on the datasheet [13], the estimated positional uncertainty on the X axis is around $\pm 120\mu m =$ 0.12 mm, which is the scanner's upper-bound X sampling resolution. From the specified linearity deviation, the Z axis uncertainty is approximately $\pm 65\mu m =$ 0.065 mm. However, these uncertainties are upper-bounds and systematic deviations, and do not necessarily describe random point-wise noise, which still exists, but is considerably smaller.

This provides some insight to the type of noise that can be found in the 3D point clouds. Any measurements on point clouds made in this work will inevitably be affected by this noise.

2.3 Data Showcase

In this section, the three sheet metal parts that this work will focus on are shown. Additionally, the total inclination angle and the arc length of a bend will be visualized, in order to provide an intuition of what this work aims to detect and quantify.

Each sheet metal part is manufactured according to a specific design, which is described by its CAD model. After manufacturing, each part is scanned using the Wenglor 2D line scanner. In Figure 2.3 below, all three CAD models, as well as the scans of the manufactured parts are displayed. The visualizations of the CAD (Computer-Aided Design) models are taken in SolidWorks [17], whereas ones for the scan point clouds are taken in CloudCompare [18].

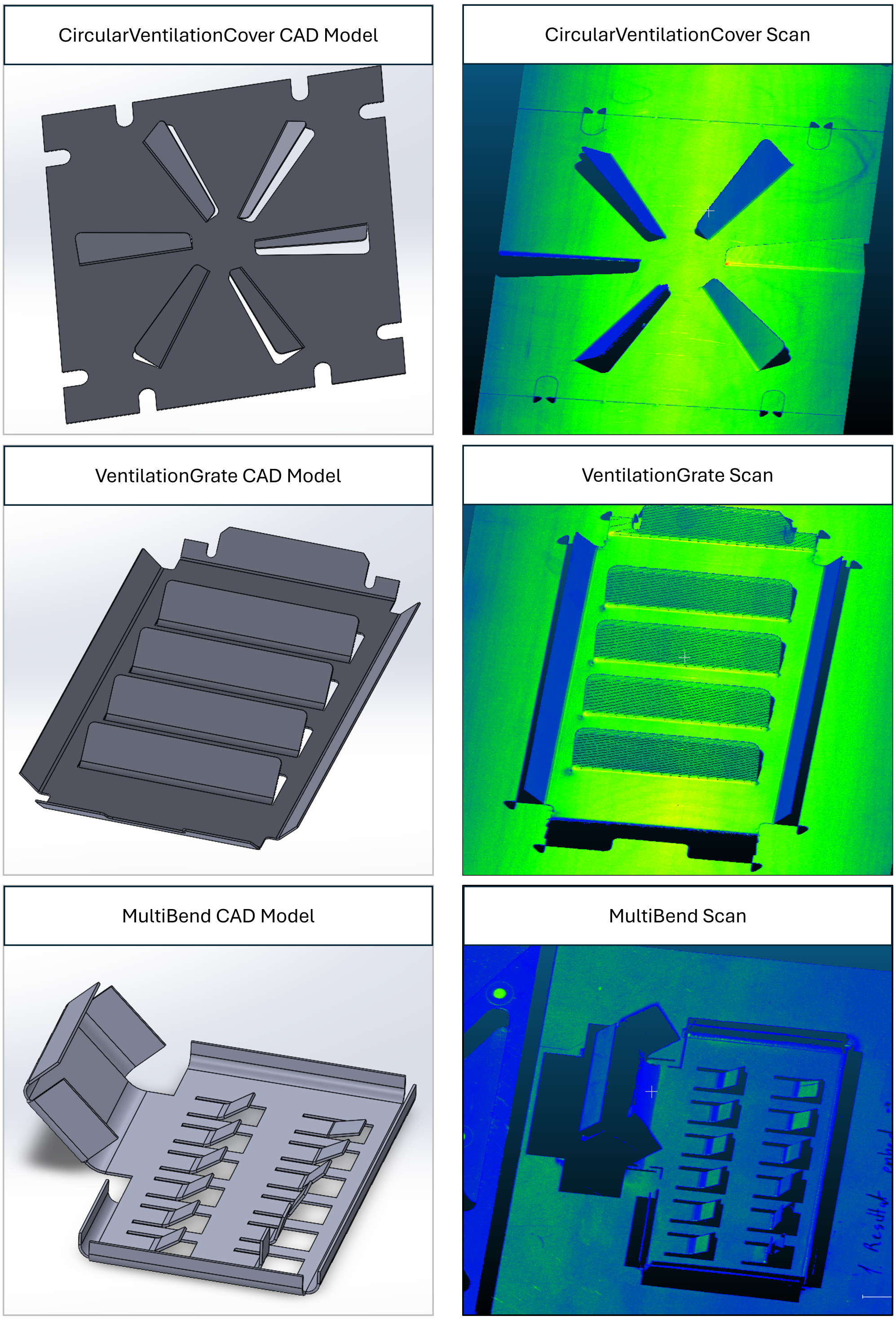
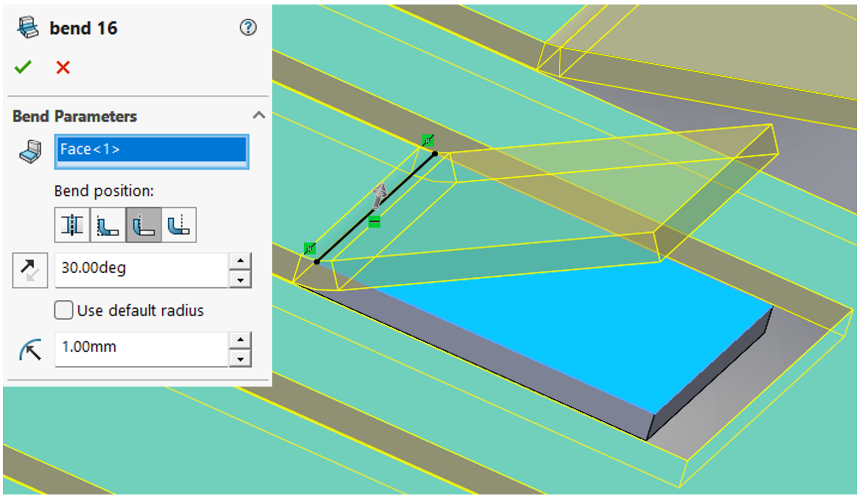

CircularVentilationCover CAD Model
CircularVentilationCover Scan
VentilationGrate CAD Model
VentilationGrate Scan
MultiBend CAD Model
MultiBend Scan

Figure 2.3: The three sheet metal parts in the dataset. In each row, the CAD model of a given part is shown on the left, and the corresponding scan is shown on the right.

A bend in SolidWorks is created by first sketching the bend axis as a line across a flat surface, and afterwards using the *Sketched Bend* feature of SolidWorks, with the input parameters being a bend angle and a bend radius. The following figure provides a visual insight to the process, with the sketched line shown in black.



bend 16

Bend Parameters

Face<1>

Bend position:

30.00deg

□ Use default radius

1.00mm

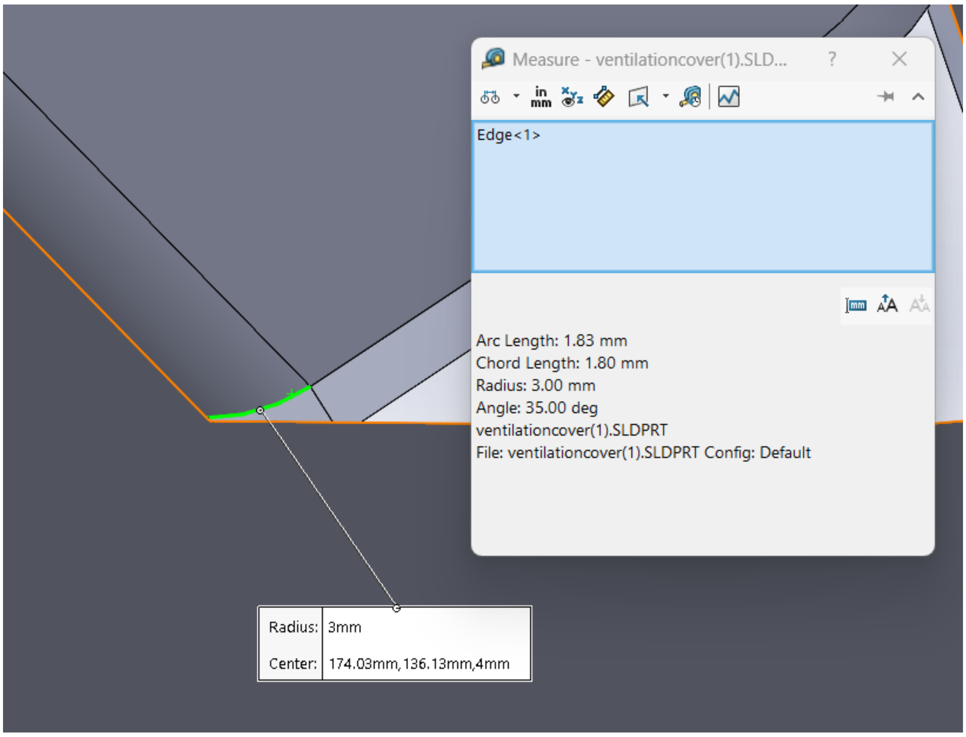
Figure 2.4: This figure shows how a bend is created and defined in SolidWorks.

The arc length of the bend is not explicitly defined in the design of the part, but can easily be calculated using the following formula:

$$\alpha = r \cdot \theta \tag{2.1}$$

where α is the arc length, r is the bend radius, and θ is the bend angle. The arc lengths of bends that are present in relevant to this work sheet metal parts range from 0.1 mm all the way up to 5 mm.

In SolidWorks, the radius that is defined when creating a bend is the *inner radius*, meaning the radius of the curvature along the inner surface of the sheet metal part. Due to the thickness of the material, the bend radius is larger on the outer surface. This affects the arc length α as well, which is different depending on which side the part is inspected from. In the current manufacturing setup, parts are only scanned from one side at a time. Thus, as long as the correct arc length (inner or outer) is considered as the CAD model ground truth in every scan, this does not pose a problem. The inner arc of a bend, whose length is to be measured, is highlighted in green in Figure 2.5 below.



Arc Length: 1.83 mm
Chord Length: 1.80 mm
Radius: 3.00 mm
Angle: 35.00 deg
ventilationcover(1).SLDPRT
File: ventilationcover(1).SLDPRT Config: Default

Radius: 3mm
Center: 174.03mm,136.13mm,4mm

Figure 2.5: A visualization of the arc created by a bend in the surface of a sheet metal part.

2.4 Origin of Errors

In Section 2.2, some of the scan noise that can be found in the 3D point clouds was analyzed. However, there are at least two other sources of error can cause discrepancies between the point clouds and the corresponding CAD model.

One origin of error comes from the possibly incorrect manufacturing of the parts by the laser manufacturing system. The system is still in its development phase, and it will inevitably fabricate the sheet metal parts imperfectly.

Another potential source of error comes from the data processing/inspection pipeline that will be implemented in this work. The following figure is a diagram that shows most of the sources of error, and how the total error is an accumulation.

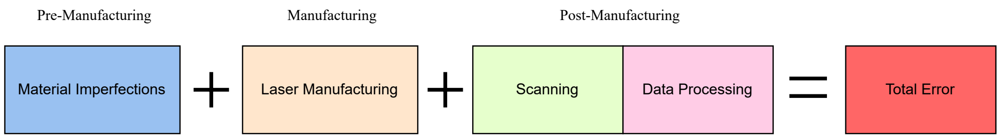


Figure 2.6: A diagram showing how errors accumulate from pre-manufacturing to post-manufacturing, leading to discrepancies between the CAD model and the 3D point cloud.

It is not trivial to determine why a manufacturing feature that is detected in the point cloud is different than the corresponding feature in the CAD model. For example, if the the arc length of a given bend is measured to be 1.58 mm in the point cloud, but the same arc length is 1.83 mm in the CAD model, is this discrepancy due to incorrect manufacturing, scanning noise or due to the measurement method?

The data processing/inspection pipeline must be able provide measurements for real scans, so its development will be focused around those. However, in order to isolate the data processing error from all other sources, and evaluate the inspection method, one idea is to sample the CAD models to create noise-free point clouds of the ideal parts. Noise (e.g. gaussian noise) can then be slowly introduced to the sampled CAD data.

One more important point to make is that a point cloud is a sparse representation of reality. The sparser the point cloud, the more inaccurate the measurements will be.

Chapter 3

Design and Methods

In this chapter, an initial design for the proposed solution will be given. Afterwards, some of the tools necessary to implement this pipeline will be examined.

3.1 Initial Design

Figure 3.1 below shows a simple version of the pipeline that will be used in this work.

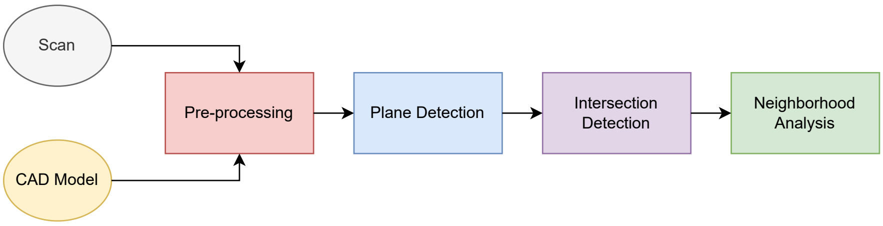


Figure 3.1: An initial design of the proposed solution.

First, the scan must be pre-processed, as it contains noise, as well as foreign regions, such as the metal fixture that the part is held by. To achieve this, the CAD model can be used as a guide. By aligning the CAD model inside the scan point cloud, it is possible to automatically identify the region of the scan in which the part is located in, and subsequently crop the rest of the scan out. One of the most common alignment techniques is to first use a global alignment method, like Fast Point Feature Histograms for feature extraction coupled with RAndom SAmple Consensus (RANSAC) for feature matching. Afterwards, Iterative Closest Point (ICP) is used for local registration refinement.

Once the part has been isolated from the rest of the scan, simple geometries such as planes can be fitted to the point cloud. The normals of these planes can provide the total inclination angle measurements. Additionally, the intersections of these planes can help narrow down the regions of interest, where bends are located.

From the intersections, local neighborhood analysis can be used to measure the arc lengths of all bends in the point cloud. The most common method of neighborhood analysis in 3D point clouds is Principal Component Analysis. Lastly, Density Based Clustering for Applications with Noise (DBSCAN) will be a useful tool throughout the whole pipeline for separating different geometries in the point clouds, as well as remove noise.

The tools named in the explanation of the initial design will be described in the rest of this chapter. For each method, its mathematical foundation will be provided. Additionally, the geometric interpretations, as well as visualizations will be supplied when appropriate. The mathematical notations defined in this chapter will be the ones used for the rest of this report.

3.2 FPFH Features as Geometric Descriptors

Fast Point Feature Histograms (FPFH), proposed by Rusu et al. [19] in 2009, are an optimization of Point Feature Histograms (PFH), which were introduced by the same authors one year earlier [20]. In this project, FPFH will be used in combination with RANSAC feature matching to align/register two point clouds together.

Point Feature Histograms are per-point local descriptors for 3D point clouds that encode the geometric relationships between a point $\vec{p}$ and its neighbors. At each point $\vec{p_i}$ in the 3D point cloud, a *Darboux frame* is defined for each unique pair of neighbors $\vec{p}_{j_1}$ and $\vec{p}_{j_2}$. The point of the pair that has a smaller angle between its own normal and the line connecting the two points is set as the source point $\vec{p}_s$, and the other point is denoted as the target, $\vec{p_t}$. This is the mathematical formulation of this condition:

$$\begin{aligned} \text{if} \quad & \vec{n}_{j_1} \cdot (\vec{p}_{j_2} - \vec{p}_{j_1}) \leq \vec{n}_{j_2} \cdot (\vec{p}_{j_1} - \vec{p}_{j_2}) \\ \text{then} \quad & \vec{p}_s = \vec{p}_{j_1} \quad \text{and} \quad \vec{p_t} = \vec{p}_{j_2} \\ \text{else} \quad & \vec{p}_s = \vec{p}_{j_2} \quad \text{and} \quad \vec{p_t} = \vec{p}_{j_1} \end{aligned} \tag{3.1}$$

Once the source and target points $\vec{p}_s$ and $\vec{p_t}$ are set, then a Darboux frame is created with its origin at $\vec{p}_s$ and its principal axes defined as:

$$\begin{aligned} \vec{u} &= \vec{n}_s \\ \vec{v} &= \frac{(\vec{p_t} - \vec{p}_s) \times \vec{u}}{|\vec{p_t} - \vec{p}_s|} \\ \vec{w} &= \vec{u} \times \vec{v} \end{aligned} \tag{3.2}$$

where $\vec{n}_s$ is the normal vector associated with the source point $\vec{p}_s$. Once this frame is defined, then a set of 4 features is calculated:

$$\begin{aligned} f_0 &= \vec{v} \cdot \vec{n}_t \\ f_1 &= |\vec{p}_t - \vec{p}_s| \\ f_2 &= \frac{\vec{u} \cdot (\vec{p}_t - \vec{p}_s)}{f_1} \\ f_3 &= \operatorname{atan2}(\vec{w} \cdot \vec{n}_t, \vec{u} \cdot \vec{n}_t) \end{aligned} \tag{3.3}$$

with $\vec{n}_t$ being the normal associated with the target point $\vec{p}_t$. Each of these four features provides unique geometric information about the pair of $\vec{p}_s$ and $\vec{p}_t$. Specifically, f_0 describes the alignment between the target normal $\vec{n}_t$ and the second axis of the Darboux frame $\vec{v}$, which is perpendicular to both the connection vector $\vec{p}_t - \vec{p}_s$ and the source point normal $\vec{n}_s$. The second feature, f_1, is the Euclidean distance between the two points in the pair, whereas f_2 is essentially the cosine of the angle between the connection vector and the source normal. Lastly, f_3 is a descriptor of how much the target normal $\vec{n}_t$ is rotated around $\vec{v}$.

These four features are then binned into a single histogram using a base-d number system. The index of the bin for a pair of four features $\{f_0, f_1, f_2, f_3\}$ is calculated as:

$$\text{index} = \sum_{i=0}^{3} \left\lfloor \frac{f_i - f_{i_{min}}}{f_{i_{max}} - f_{i_{min}}} \cdot d \right\rfloor \cdot d^i \tag{3.4}$$

where $\lfloor\rfloor$ denotes the integer floor operation.

In this way, a histogram is created for point $\vec{p}_i$ (the original point), and it is populated by m values, where m is the amount of unique pairs of points in $\vec{p}_i$'s neighborhood. This histogram acts as a geometric descriptor for point $\vec{p}_i$.

These were Point Feature Histograms (PFH). Fast Point Feature Histograms (FPFH) are an optimization for the calculation of PFH, where instead of all unique pairs in every neighborhood, only the pairs between the original point $\vec{p}_i$ and each of its neighbors are considered. This will result in what is referred to as a Simplified Point Feature Histogram (SPFH) for each point. A second pass through all points aggregates the SPFHs of the neighbors of each point $\vec{p}_i$ (weighted by their distance to the point) to get the final histogram for $\vec{p}_i$:

$$\text{FPFH}(\vec{p}_i) = \text{SPFH}(\vec{p}_i) + \frac{1}{k} \cdot \sum_{j=1}^{k} \frac{1}{|\vec{p}_i - \vec{p}_j|} \cdot \text{SPFH}(\vec{p}_j) \tag{3.5}$$

FPFH reduces the computational complexity of PFH from $\mathcal{O}(k^2)$ to $\mathcal{O}(k)$ and is the variant of the method that is most commonly used.

3.3 RANSAC for Feature Matching and Plane Detection

RANdom SAmple Consenus (RANSAC) is an iterative algorithm originally proposed by Fischler and Bolles in 1981 [21]. It is used to estimate the parameters of a mathematical model when provided with a set of data that contains outliers. The algorithm is composed of the following 4 steps:

1. Randomly select the minimum number n of data points required to define the mathematical model chosen.
2. Instantiate the model $f(x)$ using the selected points and evaluate all data points to determine which are inliers (points that fit the model within a predefined threshold t).
3. Repeat this process for a fixed number of iterations i, or until a sufficient number of inliers k is found.
4. Return the model which contributed to the most amount of inliers across all iterations.

In this project, RANSAC is used for two purposes; 1. FPFH feature matching and alignment estimation, and 2. Planar surface detection.

RANSAC for Feature Matching and Alignment Estimation

After calculating FPFH features for two point clouds, RANSAC samples a small set of points from one of the point clouds (called the source point cloud). Nearest-neighbor search in the *FPFH descriptor space* is performed in order to match those points with the best candidates from the other point cloud (the target point cloud). In this way, for a given point $\vec{p_i}$ in the source point cloud $\vec{p_i} \in P_s$, the matching point point in the target point cloud $\vec{q_i} \in P_t$ is found by:

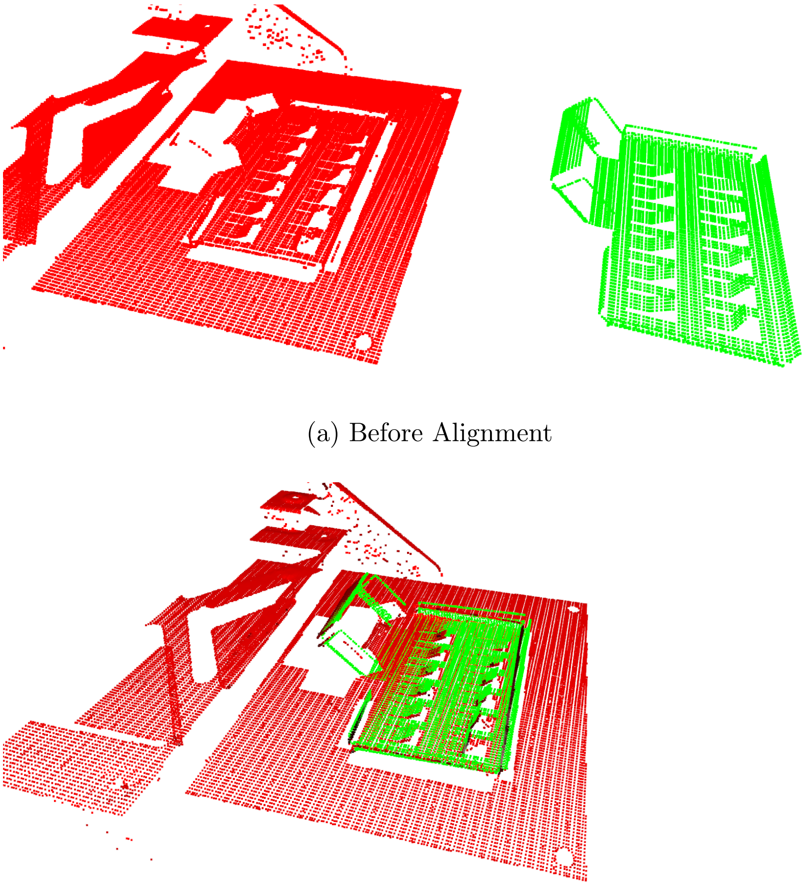
$$\vec{q_i} = \underset{\vec{q} \in P_t}{\arg\min} \, ||\text{FPFH}(\vec{p_i}) - \text{FPFH}(\vec{q})|| \tag{3.6}$$

By minimizing the distance between the transformed set of source point cloud points $R \cdot \vec{p_i} + \vec{t}$ to the matched target point cloud points $\vec{q_i}$ cloud under the constraint of rigid transformation, the optimal transformation matrix T is found. This minimization problem, expressed in equation 3.7 below, is typically solved using a least squares approach.

$$\min_{R,\vec{t}} \sum_{i=1}^{n} \left|\left|R \cdot \vec{p_i} + \vec{t} - \vec{q_i}\right|\right|^2 \tag{3.7}$$

The optimal transformation is applied to the source point cloud and each of its points is counted as an inlier if there exists a point in the target point cloud within a predefined distance threshold t. The process of sampling points from the source point cloud, matching them to ones in the target point cloud, estimating the optimal rigid transformation, and counting number of inliers is performed i times. The transformation T that contributed to the most amount of inliers is returned.

This process is visualized in Figure 3.2. In 3.2a, the scan point cloud of the MultiBend part and its associated CAD model (sampled) are shown prior to alignment. The scan point cloud is shown in red, whereas the CAD point cloud is shown in green. Both point clouds have been downsampled for visualization clarity. In 3.2b, the two point clouds have been aligned by first extracting both of their FPFH features and subsequently estimating the optimal transformation T using RANSAC.



(a) Before Alignment

(b) After FPFH+RANSAC Alignment

Figure 3.2: This figure is a visualization of the FPFH + RANSAC alignment method between two point clouds. The point cloud shown in red is the scan of the MultiBend part, whereas the point cloud shown in green is the associated CAD model, which was sampled using raycasting.

Pros and cons of RANSAC This method of using RANSAC to align two point clouds through matching their FPFH features is called a *global alignment* method. This is because the matches are not established in Euclidean space, but in descriptor space, which is irrespective to the XYZ positions of the two point clouds. One drawback of this method however is that as with all RANSAC implementations, it is a stochastic method, meaning that a different registration will be estimated each time it is executed. Additionally, it does not iteratively refine the optimal registration, resulting in a slightly inaccurate alignment. To visualize this, Figure 3.3 below provides a close up view of FPFH + RANSAC alignment.

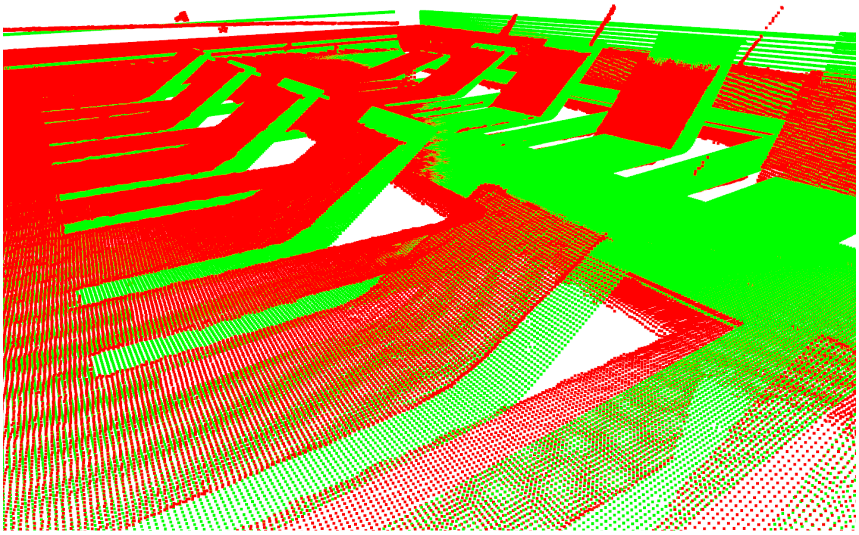


Figure 3.3: A close up visualization of the FPFH + RANSAC alignment. The sampled MultiBend CAD point cloud is shown in green, and the scan point cloud of the same part is shown in red.

This is why it is common to couple FPFH + RANSAC registration with a *local alignment* method, like Iterative Closest Point (ICP), which will be explained in Section 3.4.

RANSAC for Planar Surface Detection

RANSAC can also be used to detect distinct geometries in 3D point clouds, such as planar surfaces. In order to detect a planar surface, first, three random points $\vec{p}_\alpha$, $\vec{p}_\beta$ and $\vec{p}_\gamma$ are sampled from the point cloud. Given that the three points are not collinear, the normal vector of the plane π formed by these points is calculated:

$$\vec{n}_\pi = (\vec{p}_\beta - \vec{p}_\alpha) \times (\vec{p}_\gamma - \vec{p}_\alpha) \tag{3.8}$$

The normal is then substituted into the plane equation:

$$n_{\pi_x} \cdot x + n_{\pi_y} \cdot y + n_{\pi_z} \cdot z + \delta = 0 \tag{3.9}$$

By substituting the coordinates of each of the points into x, y, z, the offset δ can be found, thus a plane is formed. The rest of the points in the point cloud are then counted as inliers if their orthogonal distance to plane π is within a predefined threshold t. For a point $\vec{p}$, given that the normal vector $\vec{n}_\pi$ is normalized, this condition is evaluated as:

$$|(\vec{p} - \vec{p}_\alpha) \cdot \vec{n}_\pi| \leq t \tag{3.10}$$

Point $\vec{p}_\alpha$ can be any point on the plane. The process of sampling three points from the point cloud, forming the plane π and counting how many points are inliers is repeated i times, and the plane that contributed to the most amount of inliers is returned in the form shown in equation 3.9, with the vector $\vec{n}_\pi = [n_{\pi_x}, n_{\pi_y}, n_{\pi_z}]$ being the normal vector of the plane. The set of points $P = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_i\}$ that were counted as inliers is also returned.

Figure 3.4 below visualizes the plane that resulted in the most amount of inliers after $i = 1000$ iterations with a threshold of $t = 0.4$ mm for the MultiBend scan point cloud (after pre-processing). Inliers are visualized in blue color, where the rest of the point cloud is in gray.

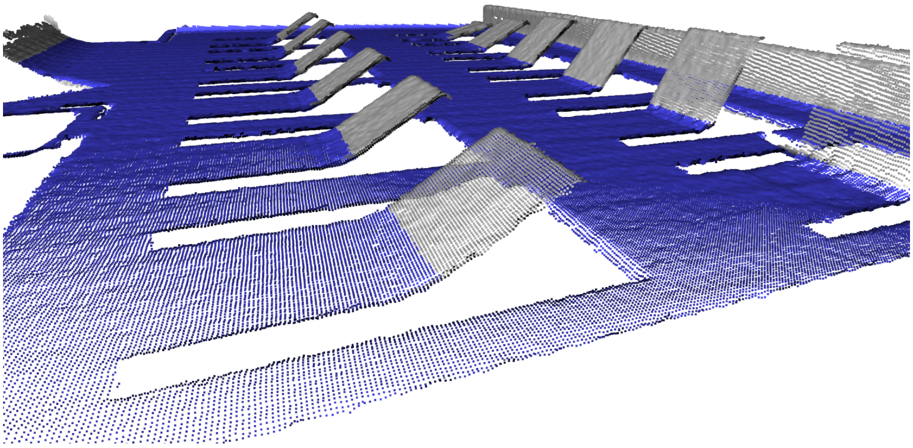


Figure 3.4: Result of RANSAC plane detection on the MultiBend scan. The inliers of the fitted plane are shown in blue, whereas the rest of the point cloud is shown in gray. The hyperparameters used where $i = 1000$ and $t = 0.4$ mm.

3.4 ICP for Point Cloud Alignment

Iterative Closest Point (ICP) is a method used to geometrically align two point clouds. It works by iteratively establishing a set of correspondences between the two and then estimating the optimal rigid transformation that minimizes the distance between the correspondences. It was proposed separately by Besl et al. [22] and Chen et al. [23], both in 1992.

As with the previous alignment method discussed, one of the point clouds is set as the source point cloud P_s and the other as the target point cloud P_t. Because ICP is a method prone to local minima, an initial transformation T_{init} is provided for P_s. The default initial transformation is the identity matrix $T_{init} = I_4$, but common choices include center-to-center translation:

$$T_{init} = \begin{bmatrix} I_3 & \vec{t}_{c-c} \\ \vec{O}^T & 1 \end{bmatrix}, \text{ with } \vec{t}_{c-c} = \frac{1}{|P_s|} \cdot \sum_{\vec{p}_i \in P_s} \vec{p}_i \;-\; \frac{1}{|P_t|} \cdot \sum_{\vec{p}_j \in P_t} \vec{p}_j \tag{3.11}$$

as well as FPFH + RANSAC global alignment, which is what is used in this project. The initial transformation is applied to the source point cloud:

$$P_s' = T_{init} \cdot P_s \tag{3.12}$$

Then, a set of n point correspondences $K = \{(\vec{p}_i, \vec{q}_i) \text{ for } i \text{ in } n\}$ is established between points in the transformed source point cloud $\vec{p} \in P_s'$ and the target point cloud $\vec{q} \in P_t$, with n equal to the number of points of the point cloud of smallest cardinality. This set of correspondences can be determined in various ways, with the most common being cross-cloud nearest neighbor search. From this set, a rigid transformation is calculated that minimizes the error between the correspondences, which takes the form of a least squares problem:

$$\min_{R,\vec{t}} \sum_{i=1}^{n} \left|\left|\vec{q}_i - (R \cdot \vec{p}_i + \vec{t})\right|\right|^2 \tag{3.13}$$

In practice, this least squares problem (and the one described in equation 3.7) is solved using Singular Value Decomposition (SVD), commonly referred to as the Kabsch-Umeyama algorithm. [24, 25]. After the optimal rigid transformation is computed, it is applied to the already transformed source point cloud P_s' and a new set of correspondences K' is established. This process repeats until the point clouds have *converged* significantly, or a maximum number of iterations i is reached. Convergence is usually evaluated through the RMSE of the point correspondences after the transformation is applied.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left\|\vec{q}_i - (R \cdot \vec{p}_i + \vec{t})\right\|^2} \tag{3.14}$$

In the following Figure, a close up view of the resulting transformation estimated by ICP is visualized, with the starting alignment being provided by FPFH + RANSAC.

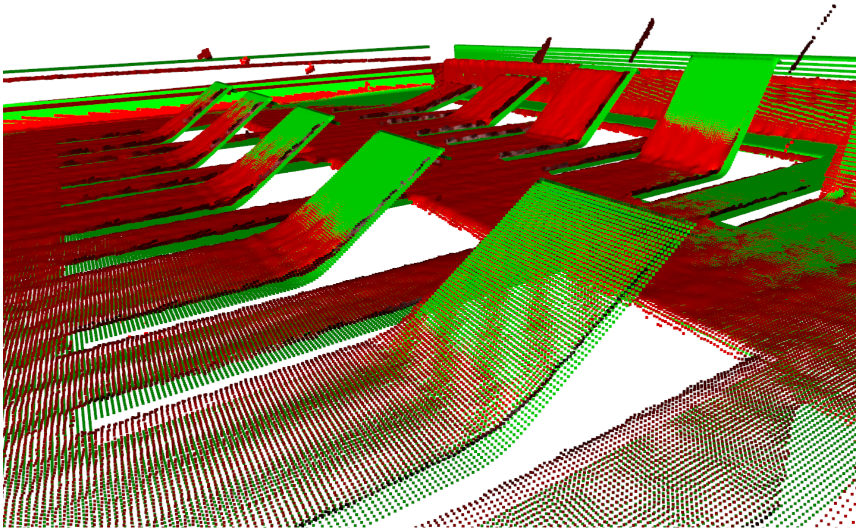


Figure 3.5: Result of ICP local refinement on the FPFH + RANSAC initial registration. The sampled cad point cloud is shown in green and the scan point cloud is shown in red. In some places, where only one of the two colors is visible, it is because one of the point clouds is right on top of or behind the other.

In this example, the RMSE metric is measured to be 0.8364 mm after FPFH + RANSAC alignment and 0.4792 mm after ICP refinement.

3.5 DBSCAN for Clustering and Noise Removal

DBSCAN, short for Density-Based Spatial Clustering of Applications with Noise is a data clustering method introduced by Ester et al. in 1996 [26]. For any given point $\vec{p}$ in the point cloud P, the ϵ-neighborhood of the point is defined as:

$$N_\epsilon(\vec{p}) = \{\vec{q} \in P \mid \| \vec{q} - \vec{p} \| \leq \epsilon\} \tag{3.15}$$

All points in the point cloud are classified into one of three categories: 1. core points, 2. border points, and 3. noise points. Below are the definitions of each category:

1. Core point: a point $\vec{p}$ whose ϵ-neighborhood contains at least *MinPts* number of points.
$$|N_\epsilon(\vec{p})| \geq \text{MinPts} \tag{3.16}$$
Core points will be denoted as $\vec{p_c}$.

2. Border point: a point $\vec{p}$ whose ϵ-neighborhood contains less than MinPts number of points, but there exists a core point $\vec{p_c}$ whose ϵ-neighborhood contains it.
$$|N_\epsilon(\vec{p})| < \text{MinPts} \quad \text{and} \quad \exists\, \vec{p_c} \in P : \vec{p} \in N_\epsilon(\vec{p_c}) \tag{3.17}$$

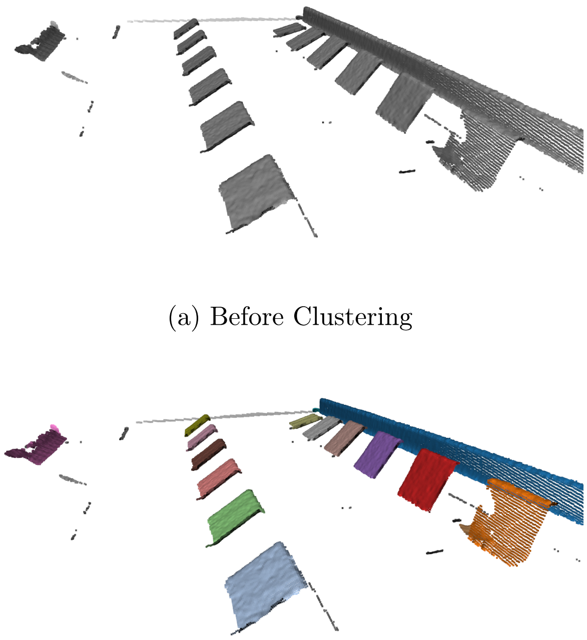
3. Noise point: a point $\vec{p}$ that has less than MinPts in its ϵ-neighborhood and there does not exists a core point $\vec{p}_c$ whose ϵ-neighborhood contains it.

$$|N_\epsilon(\vec{p})| < \text{MinPts} \quad \text{and} \quad \forall\, \vec{p}_c \in P : \vec{p} \notin N_\epsilon(\vec{p}_c) \tag{3.18}$$

Once all points have been classified as core, border, or noise points, clusters are formed through the following iterative process:

1. Select an unvisited core point $\vec{p}_c$ and assign it a new cluster ID. All points in its ϵ-neighborhood are assigned the same cluster ID.
2. For each neighbor that is also a core point, recursively assign that point's neighbors to the same cluster.
3. Once there are no reachable core points left, select a new unvisited core point and repeat the process with a new cluster ID.

In the following figure, the result of the DBSCAN method is visualized for the MultiBend part, after the main surface was detected by RANSAC and removed. Each cluster detected by DBSCAN is shown in a separate color.



(a) Before Clustering

(b) After Clustering

Figure 3.6: A visualization of the DBSCAN algorithm applied to the MultiBend scan, after the main surface was detected by RANSAC and removed. Each cluster is visualized with a unique color. The hyperparameters used were $\epsilon = 0.8$ mm and MinPts $= 50$

3.6 PCA for Normal Estimation and Neighborhood Analysis

Principal Component Analysis (PCA) is a statistical analysis method traditionally used for dimensionality reduction, first introduced by Karl Pearson in 1901 [27]. PCA transforms the original data into a new coordinate system whose axes correspond to the directions of maximum variance. In the context of point cloud processing, PCA is most commonly used to estimate surface normals and for local neighborhood analysis.

More specifically, for a given point cloud P, let $N_\epsilon(\vec{p}) \subset P$ be a local neighborhood around $\vec{p}$, where $N_\epsilon(\vec{p}) = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_k\}$. The local mean of the neighborhood is defined as:

$$\vec{\mu} = \frac{1}{k}\sum_{i=1}^{k} \vec{p}_i \tag{3.19}$$

The covariance matrix of the neighborhood $C_N \in \mathbb{R}^3$ is then given by:

$$C_N = \frac{1}{k}\sum_{i=1}^{k}(\vec{p}_i - \vec{\mu})(\vec{p}_i - \vec{\mu})^T \tag{3.20}$$

Eigen decomposition is then applied to the covariance matrix:

$$C_N = U \cdot \Lambda \cdot U^T \tag{3.21}$$

Here, $U \in \mathbb{R}^3$ is an orthogonal matrix whose columns $\vec{v}_1$, $\vec{v}_2$, and $\vec{v}_3$ are the eigenvectors of C_N and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$ is a diagonal matrix whose diagonal entries are the corresponding eigenvalues.

The eigenvectors describe the directions of variance in a given neighborhood, and the eigenvector that corresponds to the smallest eigenvalue is the *surface normal* of that neighborhood. A visualization of PCA in a local neighborhood $N_\epsilon(\vec{p})$ can be seen in Figure 3.7 below, where the points $\{\vec{p}_1, \vec{p}_2, ..., \vec{p}_k\}$ in the neighborhood are shown in blue, and the three eigenvectors $\vec{v}_1$, $\vec{v}_2$, and $\vec{v}_3$ are also visualized. The eigenvectors are scaled by the square root of their corresponding eigenvalue.

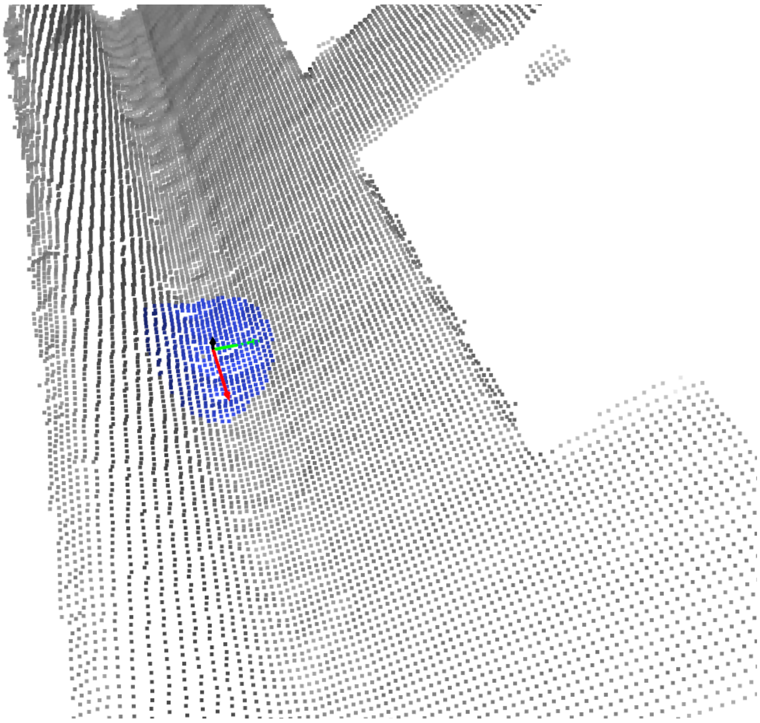


Figure 3.7: A visualization of PCA, where the eigenvectors of the local neighborhood in blue are shown each in a different color; green, red and black.

Chapter 4

Implementation

In this chapter, the entire pipeline for measuring the angles, as well as the arc lengths of all bends in a given point cloud will be established. In each section, there will be an overview paragraph, where the corresponding part of the pipeline will be briefly described, after which, a technical explanation will be provided. The flowchart of the whole pipeline can be seen in Figure 4.1 below.

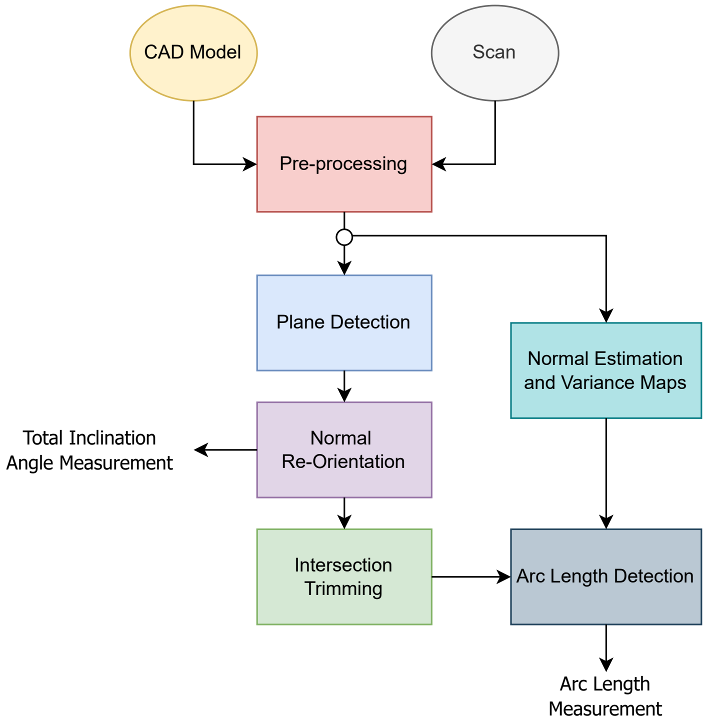


Figure 4.1: The implementation pipeline

4.1 Pre-Processing

Overview The goal of the pre-processing stage is to isolate the sheet metal part from the rest of the scan, as well as to remove noise. First, the average density of the scan point cloud is calculated. Then, a ray-casting process allows the CAD model to be sampled in a way that matches the structure created by the laser scanner. The sampled CAD model is aligned to the scan point cloud using FPFH+RANSAC for initial alignment, and ICP for refinement. The scan point cloud is then cropped, and noise is removed by using a combination of statistical outlier removal and DBSCAN. The result of the pre-processing pipeline can be seen in Figure 4.7, with the final point cloud shown in gray.

Technical Explanation

As it was shown in Figure 3.2, other than the part itself, the scanner picks up the surrounding fixtures, and noise is also present. In order to automatically isolate the part from the rest of the scan and remove noise, the following pipeline is implemented:

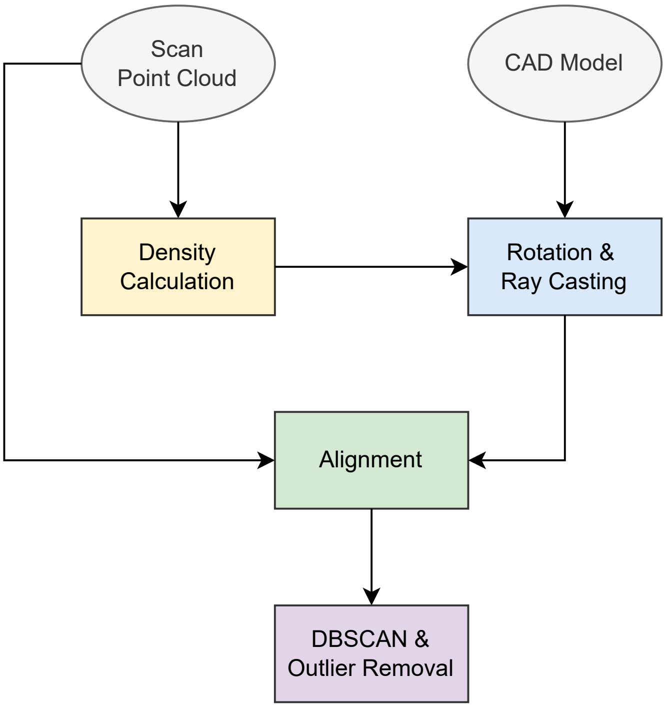


Figure 4.2: This figure shows the flowchart for automatically preprocessing the scan data.

Density Calculation

The average point density of the scan $\bar{\rho}$, represents the mean number of points present in a sphere of radius $r = 1\,\text{mm}$. It is calculated using the following formula:

$$\bar{\rho} = \frac{1}{|P|} \sum_{\vec{p} \in P} |N_\epsilon(\vec{p})| \quad [\text{points}/\text{mm}^3] \tag{4.1}$$

where P is the set of all points in the point cloud, $|P|$ is cardinality of the set, and $|N_\epsilon(\vec{p})|$ is the number of points in the ϵ-neighborhood of $\vec{p}$, meaning points that have a distance equal to or less than $\epsilon = 1\,\text{mm}$ to point $\vec{p}$. Points with 10 or less neighbors ($|N_\epsilon(\vec{p})| < 10$) are excluded from the calculation, in order to prevent noise from affecting the average.

Rotation and Ray Casting

The CAD model of the part can be used to isolate it from the rest of the scan. As previously mentioned, the parts are currently only scanned from one side[1]. Simply sampling the mesh representation of the CAD model will result in a point cloud that looks different than the scan, due to occlusions that can occur during scanning from only one side, as well as the inherent thickness of the metal. In order to give the point cloud representation of the CAD model the same structure as the scanned part, it can be loaded as a mesh, rotated in the same way as the real part was scanned and rays can be cast top down to provide the final point cloud.

First, the CAD file is loaded and converted into a mesh. The mesh is then rotated in order to match the orientation of the part when it was scanned. A rotation of the mesh for ϕ degrees around the Z axis is needed to match the orientation of the part as it was placed in the scanning fixture, and a rotation of θ degrees around the X axis will match the rotation of the fixture. The rotations are applied using the standard rotation formula:

$$M_R = R_z(\phi) \cdot R_x(\theta) \cdot M \tag{4.2}$$

$$\textit{with}$$

$$R_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

where M is the matrix that contains the positions of the mesh vertices.

In order to cast the correct rays and 'scan' the rotated mesh, a linear space in the XY plane with point spacing s is created above the mesh. Spacing s is calculated

[1] The moving scanner always samples the XY plane top down, but the part can be rotated inside the scanning fixture

using the following equation:

$$s = \sqrt{\frac{\pi \cdot r^2}{\bar{\rho}}} \tag{4.3}$$

The equation can be expressed as: what must the size of the squares be, if $\bar{\rho}$ amount of squares need to fit inside a circle of radius r? This is an approximation, as the assumption that most points in the scan lie on a flat surface is made.

From each point $\vec{p_i}$ in this linear space, a ray is projected downwards with direction $\vec{d} = [0, 0, -1]$. The parametrized equation of the ray is:

$$\vec{r}(t) = \vec{p_i} + t \cdot \vec{d} \tag{4.4}$$

By solving the system of equations between $\vec{r_t}$ and the plane formed by a triangle in the mesh π_j, their point of intersection $\vec{q_i}$ can be found. The set Q of all intersection points $\vec{q_i}$ is a point cloud which approximates a top down scan of the CAD model. Figure 4.3 below shows the result of this process for the Circular Ventilation Grate CAD model.

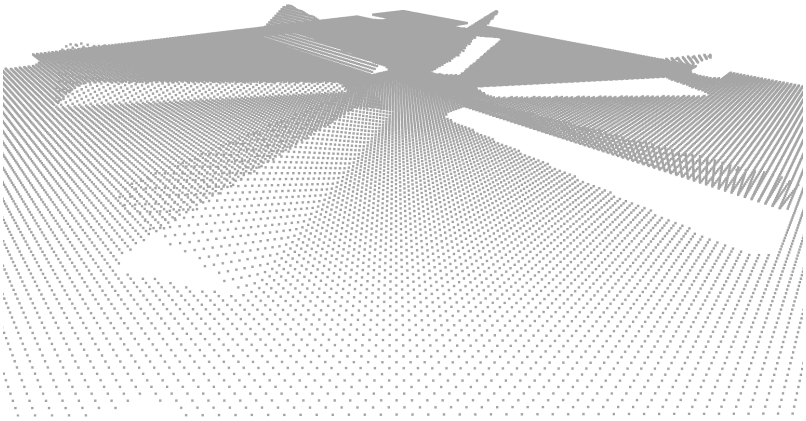
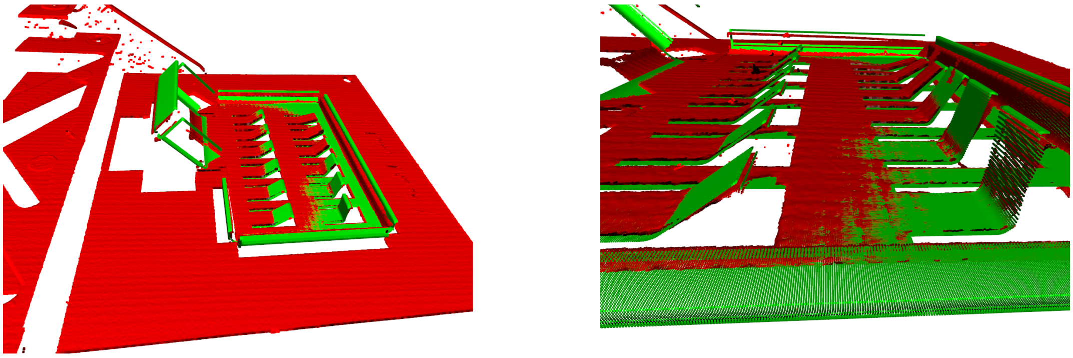


Figure 4.3: This figure shows a visualization of the ray casting process on the Circular Ventilation Grate CAD model. The spacing of the rays is increased for clarity.

Alignment

The acquired CAD point cloud (shown in Figure 4.3 above) can now be aligned with the scan point cloud. FPFH descriptors are extracted for both point clouds and a global transformation matrix is estimated using RANSAC, as described in 3.3. The two point clouds are then further aligned using ICP, with the final result shown in Figure 4.4 below.



(a) Zoomed out

(b) Zoomed in

Figure 4.4: This figure visualizes the alignment of the CAD point cloud (in green) with the scan point cloud (in red) when combining FPFH+RANSAC and ICP.

Hyperparameter sweeps are performed on ICP-only alignment, as well as FPFH + RANSAC-only alignment. These tests can be found in Appendix A, alongside visualizations and discussion on their results.

The same transformation applied to the CAD point cloud is also applied to the original CAD mesh. For each point $\vec{p}$ in the scan point cloud P_s, its signed distance $d(\vec{p}, M')$ to the transformed mesh M' is calculated. Points that are inside the mesh will have a negative signed distance, whereas points outside the mesh will have a positive one. The filtered set of points $P_{s,f}$ consists of all points that have a negative signed distance to the mesh, as well as points that have a positive signed distance less that t mm:

$$P_{s,f} = \left\{ \vec{p} \in P_s \mid d(\vec{p}, M') < 0 \text{ or } 0 \leq d(\vec{p}, M') < t \right\} \tag{4.5}$$

Threshold t is empirically set to 2.5 mm for all scans. Figure 4.5 below provides a visualization of the cropped MultiBend scan point cloud. The shown cropped point cloud is slightly downsampled for visualization clarity.

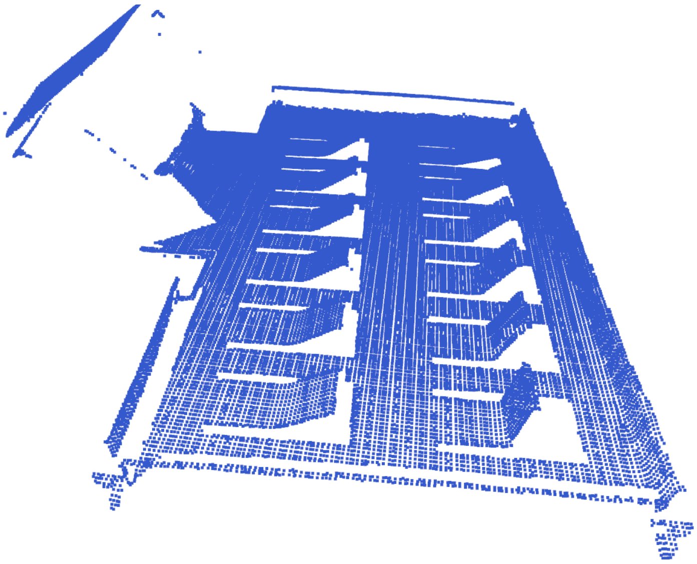


Figure 4.5: A visualization of the cropped MultiBend scan point cloud by filtering based on signed distances to the aligned mesh M'.

Noise removal

The cropped scan still contains under-sampled regions and noise. To alleviate that issue, statistical outlier removal, as well as DBSCAN can be employed.

Statistical outlier removal works by first calculating the global mean $\mu_{\bar{d}}$ and standard deviation $\sigma_{\bar{d}}$ of the average distance $\bar{d}_i$ between each point $\vec{p}_i$ in the point cloud and its k nearest neighbors:

$$\bar{d}_i = \frac{1}{k}\sum_{j=1}^{k} \left\|\vec{p}_i - \vec{p}_{i_j}\right\| \tag{4.6}$$

$$\mu_{\bar{d}} = \frac{1}{|P|}\sum_{i=1}^{|P|} \bar{d}_i \tag{4.7}$$

$$\sigma_{\bar{d}} = \sqrt{\frac{1}{|P|}\sum_{i=1}^{|P|}(\bar{d}_i - \mu_{\bar{d}})^2} \tag{4.8}$$

The set of filtered points (kept) is then defined as:

$$P_F = \left\{\vec{p}_i \in P \,\middle|\, \bar{d}_i < \mu_{\bar{d}} + t \cdot \sigma_{\bar{d}}\right\} \tag{4.9}$$

which means that points $\vec{p}_i$ whose average distance $\bar{d}_i$ from their k closest neighbors is more than the global mean of the average distances plus t times the standard deviation $\sigma_{\bar{d}}$, are regarded as noise and get filtered out. Figure 4.6 visualizes the outlier removal process, with outliers shown in red.

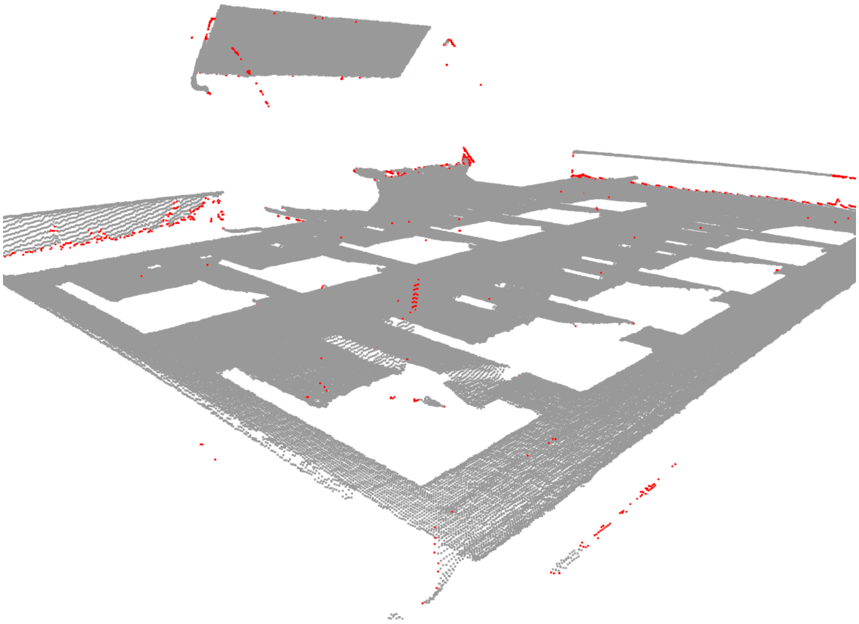


Figure 4.6: A visualization of the outlier removal process. Points outside the filtered set P_F are visualized in red, whereas the rest of the point cloud is visualized in gray. The hyperparameters used are $k = 20$ and $t = 2$

Hyperparameters $k = 20$ and $t = 2$ are standard for statistical outlier removal and those are the ones used across all four scans. Increasing the number of neighbors k or decreasing the standard deviation threshold t will result in more points being considered as outliers, and increases the risk of critical edge points being filtered out. The type of noise that persists after statistical outlier removal is usually small or occasionally large clusters of points that lie away from the main surface of the scan. In order to identify and remove those, DBSCAN can be employed.

DBSCAN is applied with hyperparameters $\epsilon = 0.6$ and MinPts $= 15$. The cluster with the most points is kept, whereas all other clusters are filtered out. A visualization of the clusters formed is shown in Figure 4.7 below, where the largest cluster is gray, and all filtered-out clusters are shown in red.

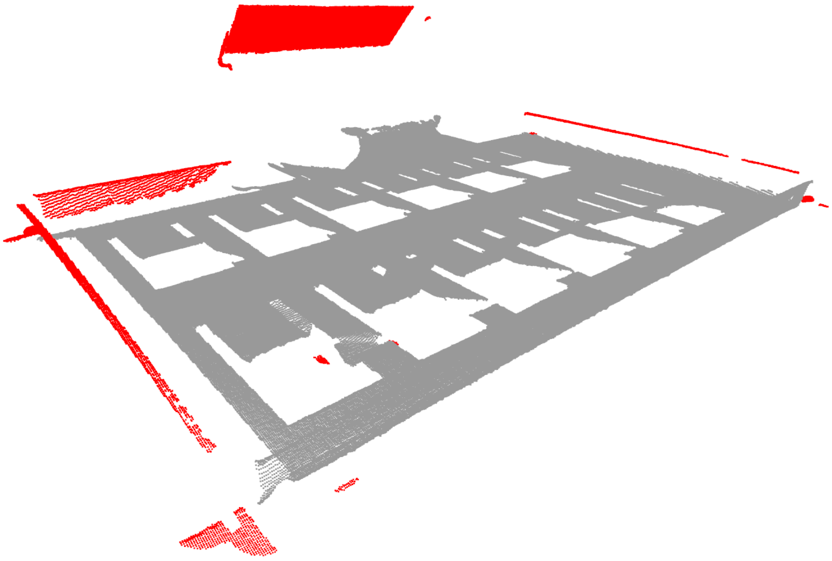


Figure 4.7: A visualization of all the clusters filtered out through DBSCAN. The main cluster, which is the one kept, is shown in gray, whereas the rest of the clusters are shown in red.

4.2 Plane Detection

Overview In this stage of the implementation, all planar surfaces in the point clouds are detected. First, the main surface is identified using RANSAC and temporarily removed. The remaining points are then clustered using DBSCAN, and a plane is fitted individually to each of the clusters. A visualization of all detected planar surfaces, as well as their associated normals can be seen in Figure 4.11.

Technical Explanation

Given a point cloud $P = \{\vec{p_1}, \vec{p_2}, ..., \vec{p_N}\}$, a **bend** is defined as a contiguous[2] set of n non-coplanar points:

$$B = \{\vec{p_i}, \vec{p}_{i+1}, ..., \vec{p}_{i+n}\} \tag{4.10}$$

that connects two disjoint sets that each contain coplanar points:

$$P_1 = \{\vec{p_j}, \vec{p}_{j+1}, ..., \vec{p}_{j+m}\} \quad \textit{and} \quad P_2 = \{\vec{k_i}, \vec{p}_{k+1}, ..., \vec{p}_{k+l}\} \tag{4.11}$$

such that all points in P_1 lie approximately on plane π_1, all points in P_2 lie approximately on plane π_2, and the points in B do not lie on either plane π_1 and π_2, but instead they form a transition region between the two.

In practice, due to the imperfect nature of the data, it is almost impossible to isolate

[2] sharing a common border; touching.

these regions completely, but defining all planes π_i in the point cloud will provide a good basis for further analysis. As explained in Section 3.3, RANSAC can be used to identify planes in a given point cloud.

Once a plane π_i is fitted with P_1 being the set of points that lie approximately on π_i, the set P_1 can be then removed from the point cloud and RANSAC can be applied again. This approach is called multi-order RANSAC or multiRANSAC [28]. There are two issues with using multi-order RANSAC to detect all planes in the preprocessed scan point clouds. First, the number of times n that RANSAC is executed (a.k.a the number of planes that are fitted) is a difficult hyperparameter to determine automatically, without any intrinsic knowledge about how many planes should be fitted to the point cloud. A different drawback to multi-order RANSAC is that it is possible for points across different surfaces to be fitted into the same plane, and thus removed from the next RANSAC iteration. This is visualized in Figure 4.8 below, where the plain whose inliers are visualized in beige is spread across disjoint surfaces.

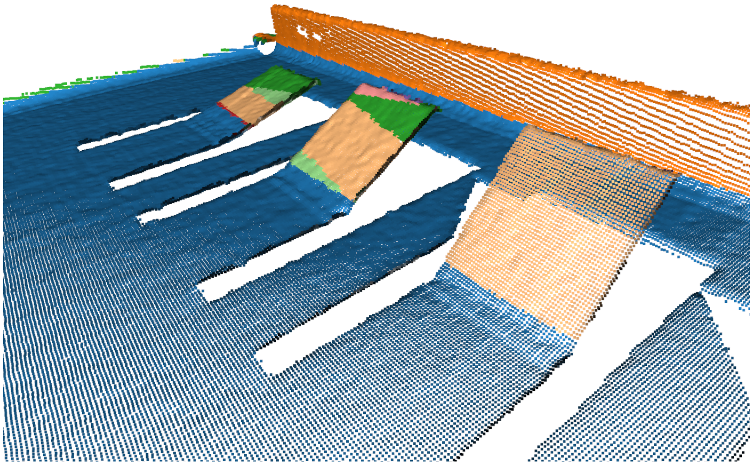
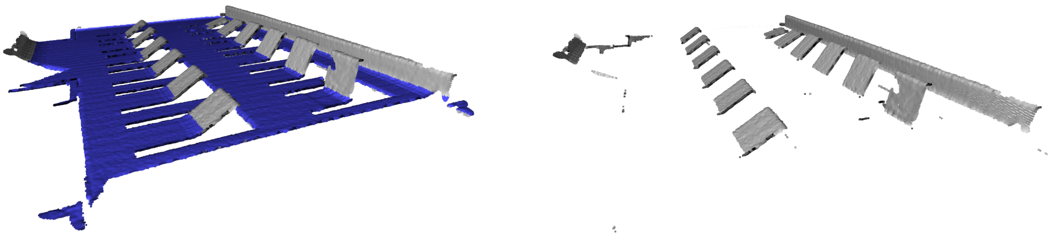


Figure 4.8: A visualization of one of the drawbacks of multi-order RANSAC, where points from disjoint surfaces get fitted into the same plane.

For these reasons, a more modular approach is used in this implementation. First, the *main planar surface* π_m of the point cloud is identified and segmented by executing RANSAC only once. The inliers of the main plane P_m are temporarily removed from the point cloud. This results in the following point cloud.



(a) Main planar surface as identified by RANSAC

(b) Main planar surface removed

Figure 4.9: The left figure shows the main/largest planar surface in the MultiBend point cloud as identified by RANSAC with the inlier points colored blue. In the right figure, the points of that surface are removed.

DBSCAN is then used to cluster the remaining point cloud and remove specs of noise created by RANSAC in the previous step. Other than the noise points that were identified by DBSCAN, an additional filter condition is applied, which requires a cluster to contain at least c number of points, with the hyperparameter c being set to 500 across all scans. Figure 4.10 below is a visualization of the identified and subsequently filtered clusters.

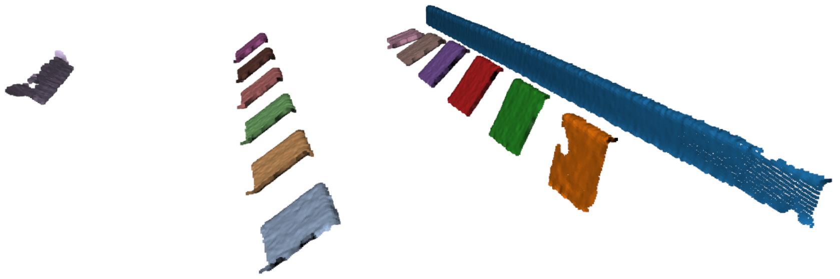


Figure 4.10: A visualization of the identified and filtered clusters for the MultiBend point cloud, once the main surface P_m has been removed.

Once these clusters have been identified, a plane can be fitted to each cluster individually by executing RANSAC only once for each cluster. As mentioned in Section 3.3 When a plane π is fitted, the equation of the plane $n_{\pi_x} \cdot x + n_{\pi_y} \cdot y + n_{\pi_z} \cdot z + \delta = 0$, as well as the inlier set of points P are both returned. In the following figure, the inliers of all planes $\pi_1, \pi_2, ..., \pi_n$ fitted to the clusters, as well as the inliers of the main plane

π_m are visualized, with each set given a unique color. The associated normal vectors of each plane $\vec{n}_m, \vec{n}_1, \vec{n}_2, ..., \vec{n}_n$ are also drawn.

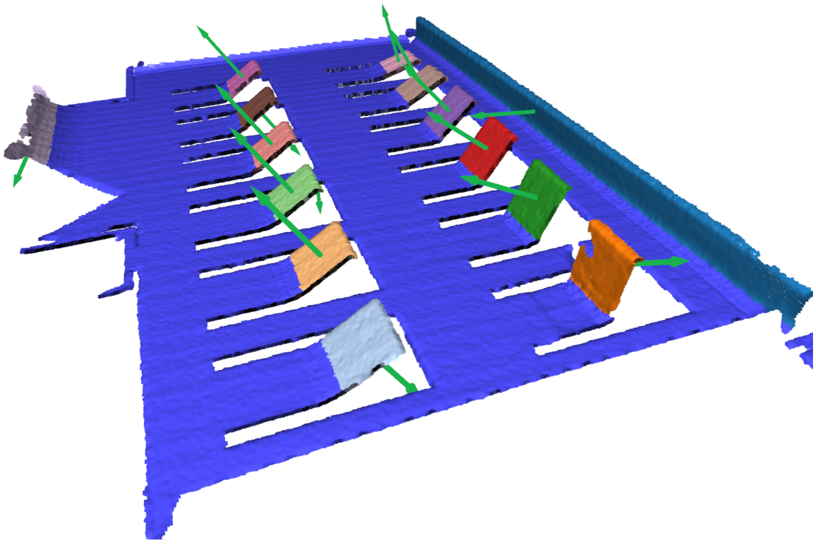


Figure 4.11: A visualization of all identified planes and their associated normals in the MultiBend scan.

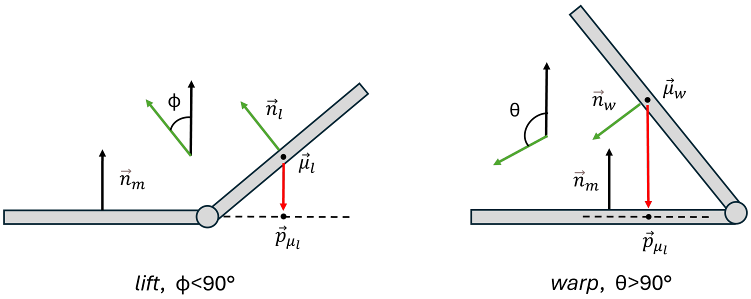
When fitting planes through RANSAC, or when estimating surface normals of local neighborhoods using PCA, the associated normal vector or surface normal $\vec{n}$ will have one of two equally probable orientations; $\vec{n}$ or $-\vec{n}$. This issue is called *sign ambiguity*, and it can be seen in Figure 4.11 above, where some normals are oriented upwards, while others are oriented downwards (including the normal of the main surface $\vec{n}_m$). If the normals of these planes were oriented correctly, then the angle between the normal of the main plane $\vec{n}_m$ and each of the other planes (e.g. $\vec{n}_1$) would provide the *total inclination angle* measurement for the associated bend. However, in their their current orientation, this angle will be measured as the complimentary of the true angle with equal probability. In the next section, the method for re-orienting the normals is analyzed.

4.3 Normal Re-orientation

Overview In this stage of the implementation, the sign ambiguity of all plane normals is resolved by introducing manufacturing knowledge about how bends are formed. By investigating the presence or absence of points underneath a planar surface, the correct orientation of its associated normal can be inferred. The resulting re-oriented normals can be seen in Figure 4.14.

Technical Explanation

If the only information given is the normals of all planes $\vec{n}_m, \vec{n}_1, ..., \vec{n}_n$, then the problem of normal re-orientation is not solvable. However, introducing information about the manufacturing of the parts, imposes a unique constraint. In order to understand this constraint, a bend will first be classified into one of two types; a *lift* and a *warp*. A *lift* is a deformation where the connecting surface rises from the main plane without exceeding a total inclination of 90°(measured as the angle between the surface normals), whereas a *warp* bend effectively wraps over the main surface and has a total inclination of more than 90°. The following figure is a simple illustration of the two types.



lift, ϕ<90°

warp, θ>90°

Figure 4.12: An illustration of the two types of bends, lifts and warps.

By projecting the mean $\vec{\mu}_{P_i}$ of the inliers P_i of a detected plane π_i onto the main plane and counting the number of points inside the ϵ-neighborhood N_ϵ of the projected point $\vec{p}_\mu$, it is fairly simple to determine whether the plane is connected to the main surface through a *lift* or a *warp*. More specifically, for a detected plane π_i and its associated set of points P_i, the mean/central point is calculated:

$$\vec{\mu}_{P_i} = \frac{1}{|P_i|} \sum_{\vec{p}_j \in P_i} \vec{p}_j \tag{4.12}$$

That mean is then projected onto the main plane $\pi_m = n_{m_x} \cdot x + n_{m_y} \cdot y + n_{m_z} \cdot z + \delta_m = 0$:

$$\vec{p}_\mu = \vec{\mu}_{P_i} - (\vec{\mu}_{P_i} \cdot \vec{n}_m + \delta_m) \cdot \vec{n}_m \tag{4.13}$$

If there are no points k in the ϵ-neighborhood N_ϵ of $\vec{p}_\mu$, then the bend which connects the two planes π_i and π_m is a *lift*.

The first step is to re-orient the normal of the main surface $\vec{n}_m$, depending on which way the part was scanned (upwards for top down scans and downwards for scans where the part is flipped). Then, all detected planes $\pi_1, \pi_2, ..., \pi_n$ are classified as coming from a lift or a warp, as described above. The angle θ_i between the normal vector $\vec{n}_i$ of plane π_i and the normal vector of the main plane $\vec{n}_m$ is then calculated:

$$\theta_i = \arccos\left(\frac{\vec{n}_i \cdot \vec{n}_m}{|\vec{n}_i| \cdot |\vec{n}_m|}\right) \tag{4.14}$$

The normal re-orientation logic is defined as:

Algorithm 1 Normal Re-orientation

```
if $\pi_i$ is a lift and $\theta_i > 90°$ then
    flip normal direction $\vec{n}_i \leftarrow -\vec{n}_i$
else if $\pi_i$ is a warp and $\theta_i < 90°$ then
    flip normal direction $\vec{n}_i \leftarrow -\vec{n}_i$
else
    Leave $\vec{n}_i$ unchanged
end if
```

This method for re-orienting plane normals might not work for planes for which $\theta_i \approx 90°$, as the projections $\vec{p}_\mu$ of planes that are almost perpendicular to the main plane will lie close to the intersection of the two planes, and consequently their ϵ-neighborhood will contain at least a few points k.

To resolve this issue, the intersections between all planes $\pi_1, \pi_2, ..., \pi_n$ to the main plane π_m are found first. To find the intersection between two planes π_i and π_m, a common point $\vec{p}_c$ which lies in both planes must be found first. This is done by setting $z = 0$ in both plane equations and solving the resulting system for x and y:

$$\begin{aligned} n_{i_x} \cdot x + n_{i_y} \cdot y + \delta_i = 0 \\ n_{m_x} \cdot x + n_{m_y} \cdot y + \delta_m = 0 \end{aligned} \tag{4.15}$$

The external product $\vec{v}_{int}$ of the two plane normals is then calculated:

$$\vec{v}_{int} = \vec{n}_i \times \vec{n}_m \tag{4.16}$$

The intersection line between π_i and π_m is then written parametrically as:

$$\vec{p}_{int}(t) = \vec{p}_c + t \cdot \vec{v}_{int} \tag{4.17}$$

With the intersection lines between the main plane π_m and all other planes established, an additional step can be introduced in the classification of warps and bends. To be specific, if $90-t \leq \theta_i \leq 90+t$, with t currently set to 15°, meaning if the initially detected angle between the normal of the main plane n_m and the normal of another detected plane n_i is within 15° of 90°, then the projected point $\vec{p}_\mu$ is moved *away* from the intersection line $\vec{p}_{int}(t)$ a certain distance d_{away}. This distance is currently set to 1 mm. This works by a re-projecting $\vec{p}_\mu$ onto $\vec{p}_{int}(t)$, to determine the right direction for translation, with the equations being the same as the once established earlier in this section. Figure 4.13 below is a visualization of the new point $\vec{p}_{away}$ determined in this way, whose ϵ-neighborhood N_ϵ will reveal whether the associated plane π comes from a lift or a warp. This new point $\vec{p}_{away}$ is colored red, whereas the initial projection $\vec{p}_\mu$ is colored blue.

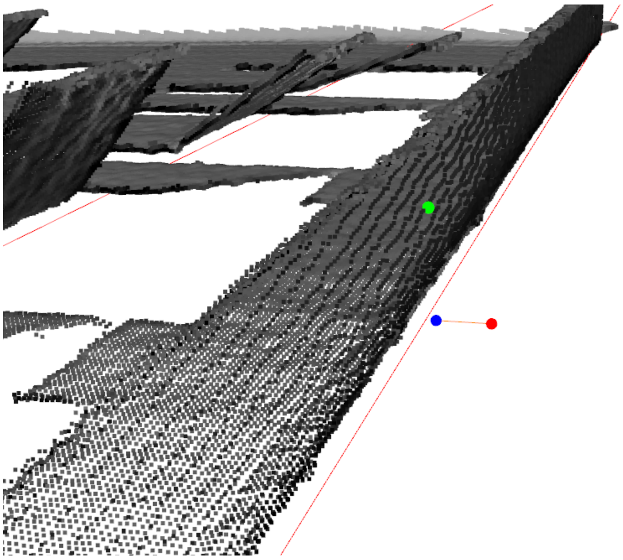


Figure 4.13: In this visualization, the mean $\vec{\mu}_{P_i}$ of one of the detected planes with inliers P_i is shown as a green sphere. The blue sphere is the projection $\vec{p}_\mu$ of $\vec{\mu}_{P_i}$ onto the main plane π_m. The red dot is the new point $\vec{p}_{away}$, found by applying a translation $\vec{p}_\mu$ such that it moves away from the intersection $\vec{p}_{int}(t)$ by 1 mm.

With this pipeline established, all plane normals $\vec{n}_m, \vec{n}_1, ..., \vec{n}_n$ can be correctly re-oriented, and the angles between the normal of the main plane $\vec{n}_m$ and all other normals can be re-calculated, using the formula in Equation 4.14. The intersections between the plane plane and each of the other planes are also found in the process, which is crucial for the downstream pipeline. The following visualization shows the re-oriented normals.

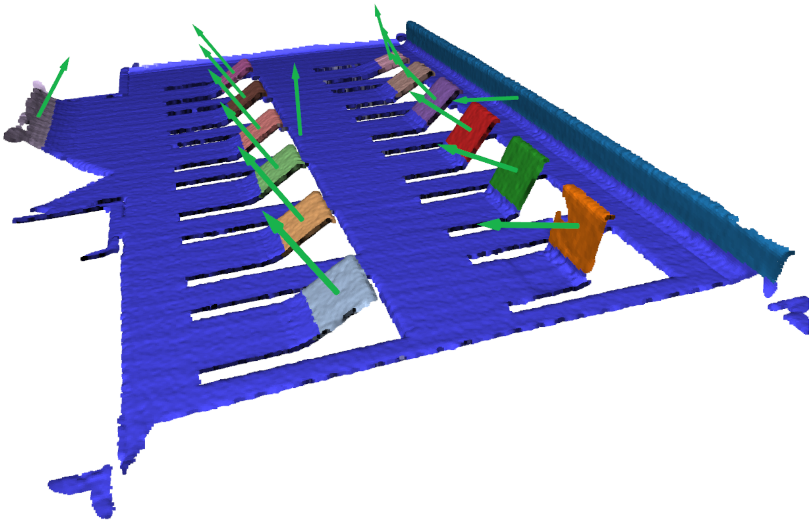


Figure 4.14: A visualization of all identified planes and their associated re-oriented normals in the MultiBend scan.

4.4 Intersection Trimming

Overview In this part of the pipeline, points are sampled along each intersection line, and the local neighborhood structure of the point cloud is analyzed at every step via PCA. Changes in the lengths of the eigenvalues or the orientation of the smallest eigenvector (surface normal) are tracked through two aggregate variables. If the local neighborhood structure changes significantly, the process stops, effectively determining where the bends start and where they end along the intersection direction. This trims down the otherwise infinite intersection lines. A visualization of the output can be seen in Figure 4.15.

Technical explanation

The intersection line $\vec{p}_{int}(t)$ between the main plane π_m and another detected plane π_i exhibits a useful geometric property; there exists a line segment $\overline{\vec{p}_a\vec{p}_b}$ along the line such that, for any point $\vec{p} \in \overline{\vec{p}_a\vec{p}_b}$, the closest point in the point cloud P will belong to the set of bend points B that connect π_m and π_i.

This subset of B that is directly accessible from $\overline{\vec{p}_a\vec{p}_b}$ will provide a good basis for local neighborhood analysis (and consequently arc length detection) across the full length of the bend. In this section, the method of estimating the line segment $\overline{\vec{p}_a\vec{p}_b}$ for each intersection is described.

For a given intersection between the main plane π_m and another detected plane π_i, the approximate center point of the line segment $\overline{\vec{p}_a\vec{p}_b}$ can be determined by projecting the mean $\vec{\mu}_{P_i}$ of the set P_i[3] onto the intersection line $\vec{p}_{int}(t)$ of the two planes. The resulting point $\vec{a}_i$ is the *anchor point* for that intersection. Inspired by the work in Matveev et al. [8], the algorithm for estimating $\overline{\vec{p}_a\vec{p}_b}$ proceeds as follows:

1. Pick a positive or negative direction $\vec{u}$ along the intersection line. Starting from the anchor point $\vec{a}_i$, take a small step with step size s in direction $\vec{u}$. Current position is $\vec{p}_c$.

2. From the current position $\vec{p}_c$, find the closest point in the point cloud $\vec{p}_b$ (b subscript for bend). Apply PCA to the ϵ-neighborhood $N_\epsilon(\vec{p}_b)$ of $\vec{p}_b$. Store the three eigenvalues $\lambda_1^j, \lambda_2^j, \lambda_3^j$, as well as the eigenvector $\vec{v}_j$ that corresponds to the smallest eigenvalue, where j indicates the current step/iteration. Repeat this process n times.

3. After n steps, calculate the following two aggregates:

$$\Lambda_j = \frac{1}{n}\sum_{l=1}^{n}\sum_{k=1}^{3}\left(\frac{\lambda_k^j - \lambda_k^{j-l}}{s}\right)^2 \tag{4.18}$$

$$\Theta_j = \frac{1}{n}\sum_{l=1}^{n}\arccos(\vec{v}_j \cdot \vec{v}_{j-l}) \tag{4.19}$$

   If either aggregate surpasses its own predefined threshold, t_Λ or t_Θ respectively, store the current position $\vec{p}_c$, reverse direction $\vec{u}$, and repeat the previous steps, starting once again from the anchor point $\vec{a}_i$. If not, then take another step in the current direction, calculate the new aggregates and evaluate them against their respective thresholds.

Each execution of the above algorithm returns the two points, $\vec{p}_a$ and $\vec{p}_b$ that form an approximation of the desired line segment $\overline{\vec{p}_a\vec{p}_b}$. Hyperparameters are empirically set to $s = 0.1$, $n = 5$, $\epsilon = 1.2$, $t_\Lambda = 0.07$ and $t_\Theta = 0.12$ for all processed point clouds.

The two aggregate variables Λ and Θ measure changes in the local neighborhood structure. In Matveev [8], only the Λ aggregate is presented, which measures changes in the magnitude of eigenvalues. Geometrically, this can be interpreted as a change in the *shape of the ellipsoid* formed by the three eigenvectors, when they are scaled by their corresponding eigenvalue. In this work, it was discovered that it is possible for this ellipsoid to maintain its shape when 'exiting' a bend, while its orientation changes. The complementary aggregate Θ was designed was designed for this purpose.

The algorithm is executed once for each intersection, and all the resulting line segments (a.k.a. the trimmed down intersections) for the MultiBend scan are visualized in red in Figure 4.15 below.

[3] P_i is the set of inlier points for plane π_i

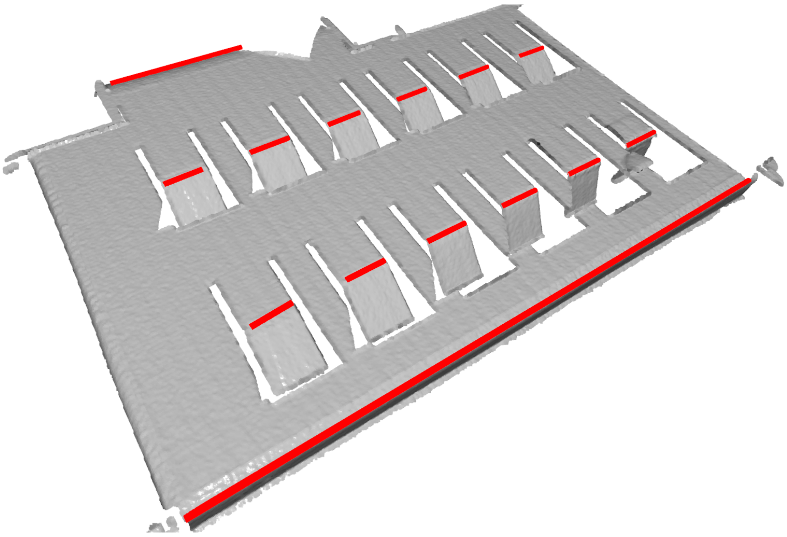


Figure 4.15: A visualization of all the trimmed down intersection lines in the Multi-Bend scan, with each line segment visualized in red.

A visualizer was also developed for this algorithm, in which the the current ϵ-neighborhood N_ϵ and its associated eigenvectors $\vec{v}_1$, $\vec{v}_2$, and $\vec{v}_3$ are shown, with each eigenvector being scaled by the square root of its corresponding eigenvalue. The current values of the two aggregates Λ_j and Θ_j are also displayed at each iteration j. In the following figure, the visualizer is shown.

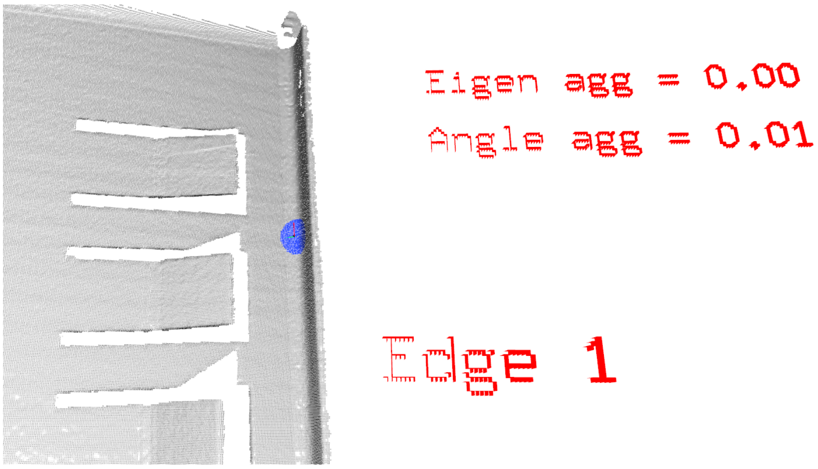


Figure 4.16: A snapshot taken from the BendLength Visualizer, which was developed to visualize the algorithm that trims down the intersections.

Before moving onto the next part of the implementation, it will be briefly mentioned how the angle deviation between a plane π_i and the main plane π_m can now be measured.

Angle Deviation By dividing an intersection line segment $\overline{\vec{p}_a\vec{p}_b}$ in three, three separate sections are created for its corresponding plane. RANSAC is used to fit a new plane to each of the sections, and the average angle difference between each of the three normals of the planes and the main surface normal is that bend's angle deviation. This will be shown as $\pm\theta^\circ$ after each angle measurement in the testing chapter.

4.5 Normal Estimation and Variance Maps

Overview In this part of the pipeline, surface normals are manually calculated for all points in the point cloud. Then, a second pass through all points can provide a per-point normal variance estimate. This can be visualized as a *normal variance map* on the point clouds. Finally the normal gradients for all points in the point cloud are also determined. These will provide the direction of travel required in the arc length detection part of the pipeline. Figure 4.17 shows the normal variance map of the MultiBend scan point cloud.

Technical Explantion

The goal of the pipeline remains to accurately measure the arc length of all bends in a given scan point cloud. An arc length is typically defined as *the distance d between two points $\vec{p}_1$ and $\vec{p}_2$ along a section of a curve γ*. Currently, points along the *bend line* can be accessed through the trimmed down intersections, but which direction $\vec{u}$ must be taken from there in order to establish the curve γ in the above definition, as well as identify the correct two points $\vec{p}_1$ and $\vec{p}_2$? In this section, the question of which direction must be taken will be investigated.

One geometric property of the bend region defined by the set of bend points B is that the orientation of the local surface remains constant under lateral movement (along the bend line), whereas it changes drastically in the perpendicular direction (maximal surface normal change). The latter direction can be described by the surface normal gradient, mathematically noted as $\nabla\vec{n}$. In order to determine this gradient, the surface normal of every point in the point cloud must first be calculated. In this implementation, this is done manually by applying PCA (see Section 3.6) to every point $\vec{p}$ of the point cloud P.

Now that each point $\vec{p}$ has an associated surface normal $\vec{n}$, a second pass through all points allows for normal variance $Var(\vec{n})$ to also be measured at each point. More specifically, for a given point $\vec{p}_i$, its associated ϵ-neighborhood $N_\epsilon(\vec{p}_i)$ and its associated normal $\vec{n}_i$, the normal variance $Var(\vec{n}_i)$ is calculated as:

$$Var(\vec{n}_i) = \frac{1}{|N_\epsilon(\vec{p}_i)|} \sum_{\vec{p}_j \in N_\epsilon(\vec{p}_i)} \|\vec{n}_j - \bar{\vec{n}}_i\|^2 \tag{4.20}$$

where $\vec{n}_j$ is the associated surface normal of point $\vec{p}_j$, and $\bar{\vec{n}}_i$ is the mean normal of the ϵ-neighborhood. The variances are normalized to the range [0, 1]. The following figure is a visualization of the *normal variance map* of the MultiBend scan point cloud, with each point $\vec{p_i}$ being colored based on its associated normal variance $Var(\vec{n}_i)$. High variance is shown in lime-yellow, where low variance is shown in navy blue.

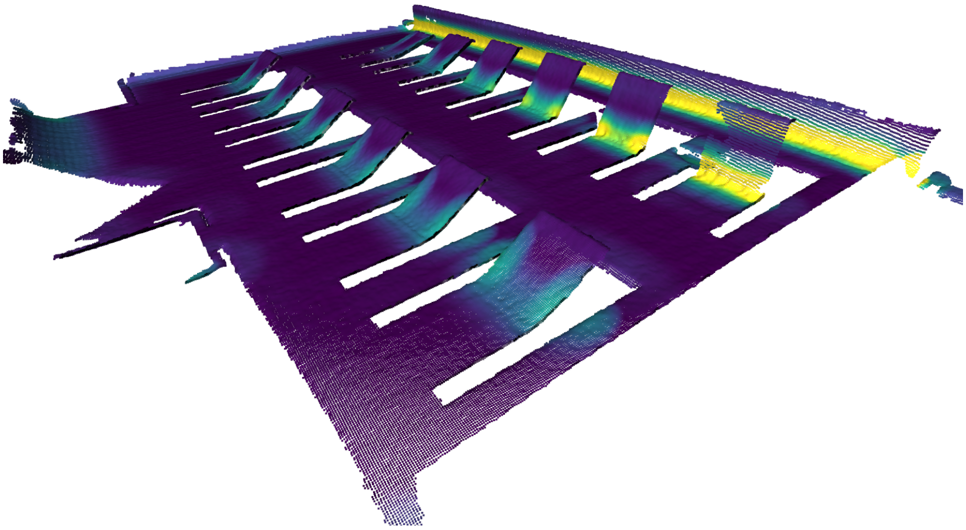
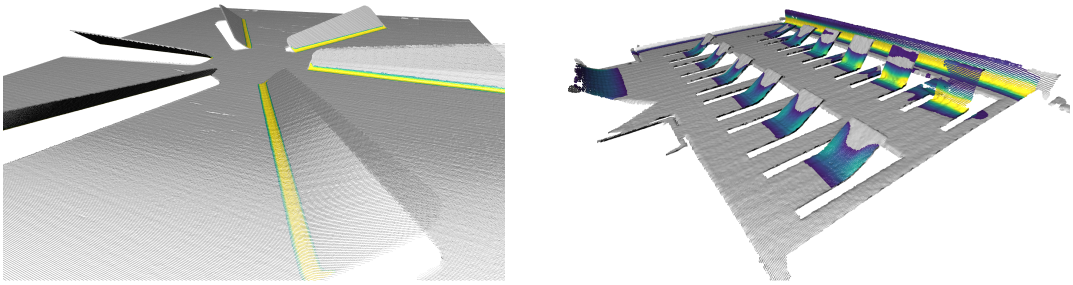


Figure 4.17: The *normal variance map* of the MultiBend part.

In an attempt to isolate all bends, a custom *region growing* algorithm is implemented, where a set $R \subset P$ starts from the anchor point a_1 of a given intersection and grows by recursively including neighboring points k if they meet a normal variance threshold criteria $Var(\vec{n}_i) > t_V$. Ideally, the resulting set R will include all points that belong to the associated bend, such that $R = B$, with the definition of B being given in Section 4.2. In practice, it is not obvious how to grow set R in order for it to perfectly match set B. The region growing algorithm works well for parts whose bends share total inclination angle and arc length, like the CircularVentilationCover part. It works less well for parts whose bends cover a range of total inclination angles and arc lengths. Those require an *adaptive variance thresholding* technique, which is currently being tested. Figure 4.18 below shows a visualization of the resulting regions in the CircularVentilationCover part, as well as the MultiBend part.



(a) Variance based region growing in the scan of the CircularVentilationCover part.

(b) Variance based region growing in the scan of the MultiBend part.

Figure 4.18: This figure shows the result of the variance region growing algorithm on two different scans.

The last step of this part of the pipeline is to calculate the surface normal gradient $\nabla\vec{n}$ of each point in the point cloud. For a point $\vec{p}_i$ in the point cloud, its normal ϵ-neighborhood $N_\epsilon(\vec{n}_i)$ is defined as:

$$N_\epsilon(\vec{n}_i) = \{\vec{n}_j \mid \vec{p}_j \in N_\epsilon(\vec{p}_i)\} \tag{4.21}$$

This translates to the set of the associated normals of the points in the neighborhood of $\vec{p}$. PCA is then applied to $N_\epsilon(\vec{n}_i)$. The eigenvector $\vec{v}$ that corresponds to the largest eigenvalue is considered equivalent to the normal gradient $\nabla\vec{n}_i$ at point $\vec{p}_i$:

$$\nabla\vec{n}_i \equiv \vec{v}, \quad \vec{v} = \underset{\vec{v}_k}{\arg\max}\,\lambda_k \tag{4.22}$$

4.6 Arc Length Detection

Overview In the final part of the pipeline, the arc length of all bends in the point cloud will be measured. By sampling points along the trimmed down intersections and finding the closest neighbor at each position, the measurement can begin. From the closest point cloud neighbor, a small step is taken along the average normal gradient. From there, an evaluation is made, to check whether the bend has ended, by determining the surface normal of the points that lie ahead and comparing it to the surface normals of the two connected planes.

Technical Explanation

The intersection lines between the main plane π_m and all other planes π_i have been trimmed down to the appropriate line segments $\overline{\vec{p}_a\vec{p}_b}$, and the normal gradient $\nabla\vec{n}$ has been determined for every point $\vec{p}$ in the point cloud. The idea for detecting the arc length of a given bend that connects plane π_m and plane π_i is now to sample points along the line segment $\overline{\vec{p}_a\vec{p}_b}$ and at each sampled point, move along the normal gradient $\nabla\vec{n}$, until the surface normal $\vec{n}$ of the local *forward-facing ϵ-neighborhood*

N_ϵ^f matches the surface normal of either of the two planes, that is $\vec{n}_i$ or $\vec{n}_m$.

More specifically, for a given intersection line segment $\overline{\vec{p}_a\vec{p}_b}$ that connects the main plane π_m and another plane π_i, points $\vec{p}_{int}$ are sampled along the segment $\overline{\vec{p}_a\vec{p}_b}$, starting from its beginning $\vec{p}_a$, all the way until reaching the end $\vec{p}_b$, with a step size of s_{int}. At each sampled point $\vec{p}_{int}$, the algorithm for detecting the arc length of the associated bend proceeds as follows:

1. Create an empty set $P_{proj} = \varnothing$

2. Find the closest point $\vec{p}_{start}$ in the point cloud. This point is the current position $\vec{p}_c$. Pick a positive or negative sign $\sigma \in \{-1, 1\}$.

3. Calculate the average normal gradient $\overline{\nabla\vec{n}_c}$ in the ϵ-neighborhood of the current position $\vec{p}_c$:

$$\overline{\nabla\vec{n}_c} = \frac{1}{|N_\epsilon(\vec{p}_c)|} \sum_{\vec{p}_j \in N_\epsilon(\vec{p}_c)} \nabla\vec{n}_j \tag{4.23}$$

   Take a small step with step size s_{gr} along the direction $\sigma \cdot \overline{\nabla\vec{n}_c}$. New position is: $\vec{p}_s = \vec{p}_c + s_{gr} \cdot \sigma \cdot \overline{\nabla\vec{n}_c}$

4. Determine the *forward-facing ϵ-neighborhood* $N_\epsilon^f(\vec{p}_s, \sigma \cdot \overline{\nabla\vec{n}_s})$ of point $\vec{p}_s$ with average gradient direction $\sigma \cdot \overline{\nabla\vec{n}_c}$ (average gradient is not re-calculated for $\vec{p}_s$)[4]. This neighborhood is defined as:

$$N_\epsilon^f(\vec{p}_s, \sigma \cdot \overline{\nabla\vec{n}_c}) = \left\{\vec{p}_j \mid \vec{p}_j \in N_\epsilon(\vec{p}_s) \text{ and } (\vec{p}_j - \vec{p}_s) \cdot (\sigma \cdot \overline{\nabla\vec{n}_c}) > 0\right\} \tag{4.24}$$

   Calculate the surface normal $\vec{n}_f$ of this neighborhood through PCA. Additionally, project $\vec{p}_s$ onto the local surface to get $\vec{p}_{proj}$ and add it to the set P_{proj}.

5. Evaluate $\vec{n}_f$ against $\vec{n}_i$ and $\vec{n}_m$, which are the normal vectors of planes π_i and π_m respectively. If the angle θ_1 between $\vec{n}_f$ and $\vec{n}_i$, or the angle θ_2 between $\vec{n}_f$ and $\vec{n}_m$ is smaller than a predefined threshold t_θ, reverse the order of points in set P_{proj}, return to the original point $\vec{p}_{start}$, pick the opposite sign σ, and repeat steps 3-5. If both signs have been chosen, move on to step 6 instead. If $\theta_1 > t_\theta$ and $\theta_2 > t_\theta$, set current position to the projected point $\vec{p}_c = \vec{p}_{proj}$ and repeat steps 3-5.

6. Once this process has been completed for both $\sigma = -1$ and $\sigma = 1$, sum the distances of all sequential points $\vec{p}_{proj}$ in set P_{proj}:

$$\alpha = \sum_{j=1}^{|P_{proj}|-1} \left\| \vec{p}_{proj}^{\,j+1} - \vec{p}_{proj}^{\,j} \right\| \tag{4.25}$$

[4] Geometrically, this neighborhood can be interpreted as the set of points present in the forward/front hemisphere of radius ϵ around point $\vec{p}_s$.

The total sum α is the measured *arc length* from point $\vec{p}_{int}$ in the intersection line segment $\overline{\vec{p}_a\vec{p}_b}$. Another step across the intersection line is taken, and the arc length from that position is measured again, until the end $\vec{p}_b$ of the line segment has been reached. The mean arc length μ_α and the arc length standard deviation σ_α are calculated for the given bend. This process is repeated for each intersection line segment $\overline{\vec{p}_a\vec{p}_b}$.

A simple interactive visualizer is created, where this process can be controlled. In the following figure, stored projected points $\vec{p}_{proj}$ are shown in pink (magenta) and the forward facing neighborhood $N_\epsilon^f(\vec{p}_s, \sigma \cdot \overline{\nabla\vec{n}_c})$ is shown in green.

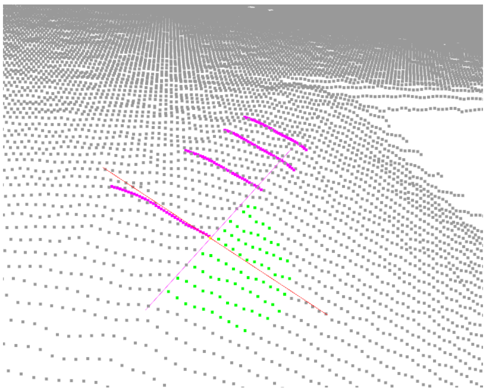


Figure 4.19: A snapshot taken from the ArcLength Visualizer, where the pink points are previous projected points and green points are the forward facing neighborhood in the current position.

Chapter 5

Testing

In order to assess the accuracy of the proposed inspection method, its error must be isolated from the manufacturing error of the fabricated sheet metal parts, as well as the error that originates from the scanning process. To achieve this, in the first section of testing, the CAD models of all three sheet metal parts in the dataset will be densely sampled using the ray-casting technique described in Section 4.1.

In the second section of testing, the CAD models of the three sheet metal parts will be sampled with point densities equivalent to the scans, and gaussian noise will be introduced in different magnitudes. This test will reveal how robust the method is to noise in the point cloud.

Finally, in the third section, the method is tested on the real scans, to show that reasonable measurements can be made on real data as well.

5.1 Testing on Densely Sampled CAD Models

In this test, a region containing one bend is defined in each CAD model, and it is densely sampled using ray-casting. This process will create ideal data for the method to be tested against. If the measured total inclination angles and arc lengths are accurate for this data, it means that conceptually, the inspection method developed in this work is sound. In the following three tables, the results of this experiment can be seen, with each table corresponding to one of the three sheet metal parts in the dataset.

CircularVentilationCover		
Metric	Ground Truth	Measured
Total Inclination Angle (°)	35.00	35.00 ± 0.00
Arc Length (mm)	1.83	1.8267 ± 0.0077

Table 5.1: Table showing the measured total inclination angle, as well as the measured arc length for one of the bends in CircularVentilationCover. Sampling density used $s = 0.03$, average point density $\bar{\rho} = 3100$, $\epsilon = 0.11$

VentilationGrate		
Metric	Ground Truth	Measured
Total Inclination Angle (°)	45.00	45.00 ± 0.00
Arc Length (mm)	3.93	3.9287 ± 0.0018

Table 5.2: Table showing the measured total inclination angle, as well as the measured arc length for one of the bends in VentilationGrate. Sampling density used $s = 0.03$, average point density $\bar{\rho} = 2960$, $\epsilon = 0.12$

MultiBend		
Metric	Ground Truth	Measured
Total Inclination Angle (°)	35.00	35.00 ± 0.00
Arc Length (mm)	1.22	1.2209 ± 0.0096

Table 5.3: Table showing the measured total inclination angle, as well as the measured arc length for one of the bends in MultiBend. Sampling density used $s = 0.03$, average point Density $\bar{\rho} = 2584$, $\epsilon = 0.15$

Discussion The total inclination angles measured for the ideal data were exactly accurate. The angle uniformity was also found to be ideal in all the densely sampled CAD models. The arc length detection was shown to be within ±0.04 mm of the true value. Based on these experiments, the method is shown to be conceptually sound.

After testing the method with a variety of different point densities $\bar{\rho}$, it is clear that there exists a relationship between the optimal ϵ hyperparameter of the arc length detector and the point density $\bar{\rho}$ of the point cloud. For the above tests, ϵ was slightly tuned in the range of [0.10 and 0.15 mm], with the chosen ϵ for each part shown in the caption of the respective table. For a given sampled CAD model, the arc length measurements deviated around ±0.07 mm, based on the chosen ϵ, which is minimal, but still notable. The mathematical relationship between point density $\bar{\rho}$ and the ϵ hyperparameter is currently being investigated.

The method developed is now shown to work on ideal data, but this is almost never be needed in reality. In the following section, the method will be tested against more realistic data.

5.2 Testing on Realistically Sampled Noisy CAD Models

In this test, the CAD model of the MultiBend part is sampled such that it matches the point density of its scan counterpart ≈ 290 points/mm^3. For comparison, the bend sampled for the MultiBend part in the previoust test had an average point density of 2584 points/mm^3, which is nearly ten times larger.

The inspection method is applied to the *realistically sampled* CAD model and the measurements (total inclination angle and arc length) for all 13 visible bends are acquired. Afterwards, gaussian noise of standard deviation $\sigma = 0.01$ mm is applied to the same sampled CAD model. This means that each point in the point cloud will now have a *positional uncertainty* of 0.01 mm. This value is chosen as the induced random per-point positional uncertainty mimics the one generated by the Wenglor scanner. The method applied to the noise-induced sampled CAD model, and the measurements are acquired. The same process is repeated once again for noise of standard deviation $\sigma = 0.02$ mm. The following tables 5.4 and 5.5 contain the the measured angles, as well as the measured arc lengths for all three levels of noise (noise-free, $\sigma = 0.01$mm, $\sigma = 0.02$ mm).

Total Inclination Angle				
Bend ID	GT (°)	Noise-free	Noise $\sigma = 0.01$ mm	Noise $\sigma = 0.02$ mm
1	90.00	86.47 ± 0.29	86.60 ± 0.10	87.12 ± 0.61
2	35.00	34.79 ± 0.00	34.86 ± 0.06	34.83 ± 0.18
3	35.00	34.79 ± 0.00	34.86 ± 0.03	34.80 ± 0.10
4	35.00	34.79 ± 0.00	34.86 ± 0.09	34.76 ± 0.16
5	35.00	34.79 ± 0.00	34.85 ± 0.06	34.77 ± 0.10
6	35.00	34.79 ± 0.00	34.87 ± 0.06	34.76 ± 0.09
7	35.00	34.79 ± 0.00	34.86 ± 0.09	34.76 ± 0.10
8	15.00	15.00 ± 0.00	14.99 ± 0.05	14.98 ± 0.10
9	30.00	29.99 ± 0.00	29.99 ± 0.05	29.97 ± 0.12
10	45.00	44.94 ± 0.00	44.95 ± 0.02	44.95 ± 0.12
11	60.00	59.68 ± 0.00	59.69 ± 0.05	59.59 ± 0.16
12	75.00	74.67 ± 0.04	74.41 ± 0.15	74.46 ± 0.16
13	90.00	89.55 ± 0.00	89.52 ± 0.06	89.40 ± 0.09

Table 5.4: This table shows the angle measurements made on the realistically sampled MultiBend CAD model, with three different noise levels applied.

Arc Length				
Bend ID	GT (mm)	Noise-free	Noise $\sigma = 0.01$ mm	Noise $\sigma = 0.02$ mm
1	3.14	3.8760 ± 0.5597	2.8469 ± 0.6196	2.8611 ± 0.5816
2	1.22	1.0909 ± 0.0000	1.0710 ± 0.2570	1.2274 ± 0.3635
3	1.22	1.0909 ± 0.0000	1.1352 ± 0.2398	1.0696 ± 0.3909
4	1.22	1.1212 ± 0.0260	1.0302 ± 0.1983	1.0988 ± 0.3073
5	1.22	1.1236 ± 0.0274	1.0274 ± 0.1581	1.2141 ± 0.3539
6	1.22	1.1236 ± 0.0274	1.1111 ± 0.3077	1.1873 ± 0.3297
7	1.22	1.1236 ± 0.0274	1.0195 ± 0.1622	1.1818 ± 0.4150
8	0.26	0.1214 ± 0.0000	0.4034 ± 0.2881	0.7462 ± 0.7752
9	0.52	0.3947 ± 0.0000	0.3696 ± 0.1872	0.5268 ± 0.4428
10	0.79	0.7262 ± 0.0178	0.6519 ± 0.2480	0.6145 ± 0.3000
11	1.05	1.0413 ± 0.0100	0.9165 ± 0.2311	1.1277 ± 0.5975
12	1.31	1.2924 ± 0.0004	1.0406 ± 0.2411	1.1409 ± 0.3249
13	1.57	1.5668 ± 0.0005	1.4104 ± 0.3101	1.3399 ± 0.3615

Table 5.5: This table shows the arc length measurements made on the realistically sampled MultiBend CAD model, with three different noise levels applied.

Discussion In Table 5.4, the first thing to notice is that the angle deviation (shown as $\pm\theta°$ grows with more noise introduced to the data. However, most of the measured angles are within 0.5° of the ground truth. The next thing that can be easily noticed, is how all measured angles are, without exception, equal to or slightly smaller than the ground truth. After visually inspecting the fitted planes, the reason is rather obvious. In Figure 5.1 below, it is shown how due to the thickness of the CAD model, points are picked up from the side surface of the material when ray-casting from top down. The same pattern is also visible in the scanned point clouds.

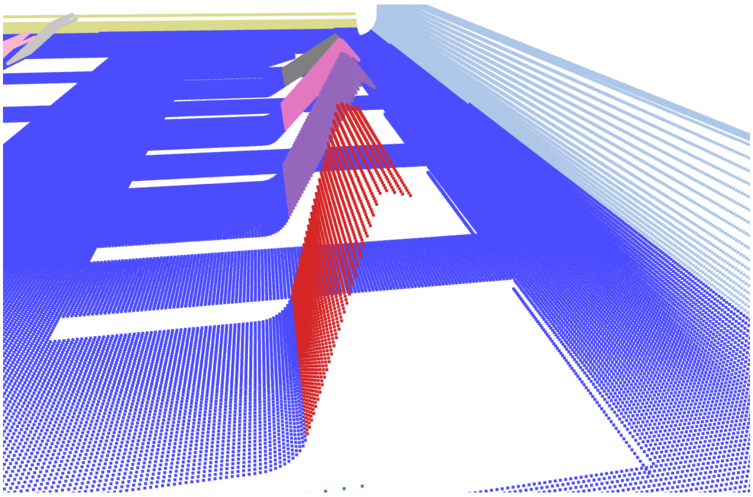


Figure 5.1: In this visualization it is shown the side surface of the CAD model is also caught by the top-down ray-casting process. This causes RANSAC to fit planes that are slightly incorrect.

In an attempt to include as many points in the fitted plane as possible, RANSAC tilts the fitted plane slightly to include these points, causing the angles between the main plane and all other planes to always be slightly smaller. In future work, this effect can be investigated. This effect was not present in the previous test (Section 5.1, because due to the ideal nature of the data, the RANSAC threshold hyperparameter was set to be minimal (0.001 mm), and thus these points were not included in the fitted planes. In this test, the hyperparameters used for real data were chosen for all methods, which is why this effect occurs.

Moving on to the arc length measurements, the same pattern of the standard deviation growing as more noise is introduced can be seen. In the noise-free sampled CAD model of real density, the method developed for measuring the arc length was quite accurate, with most measured arc lengths being within ±0.15 mm of the ground truth. An interesting underestimating effect occurs for bends with Bend ID from 2 to 7, which are designed to have the same angle and arc length, where the method measured their corresponding arc lengths to be around 0.1 mm less in all cases, with minimal standard deviation. This could mean either that the point density that the CAD models were sampled in happens to be small enough such that there are not enough points to fully describe these bends, or that a slight modification in the method could improve the measurement across all of these bends at once.

Other notable arc length measurements include the longer bend, with Bend ID 1, which had large standard deviations across all three noise-levels, as well as the smallest arc length bend, with Bend ID 8, which the method had trouble measuring in all three noise-levels. Some possible explanations for these deviations include incorrect RANSAC plane detection, as well as intersections being 'overly' trimmed by the method described in Section 4.4.

Despite some small inconsistencies, it is overall assessed that the method performed quite well on noise-free data for both angle and arc length detection. It also provided reasonable results on noise-induced data, which demonstrates robustness to noise.

In the final section of testing, the method will be applied to all three sheet metal part scans in the dataset.

5.3 Testing on the Scans in the Dataset

In this test, the inspection method is applied to the three scans in the dataset. Tables 5.6, 5.7 and 5.8 display the measured angles, as well as the measured arc lengths for all detected bends in the scans.

CircularVentilationCover				
Bend ID	CAD Angle	Measured Angle	CAD Arc Length	Measured Arc Length
1	35.00°	35.05 ± 0.45°	1.83 mm	1.2080 ± 0.2366 mm
2	35.00°	35.55 ± 0.62°	1.83 mm	0.7084 ± 0.4397 mm
3	35.00°	35.10 ± 0.60°	1.83 mm	1.0684 ± 0.1093 mm
4	35.00°	35.06 ± 0.37°	1.83 mm	1.0299 ± 0.2173 mm
5	35.00°	34.98 ± 0.69°	1.83 mm	1.4929 ± 0.4138 mm
6	35.00°	34.95 ± 0.28°	1.83 mm	1.0692 ± 0.1281 mm

Table 5.6: This table shows the measured angle and arc length for all bends in the CircularVentilationCover scan.

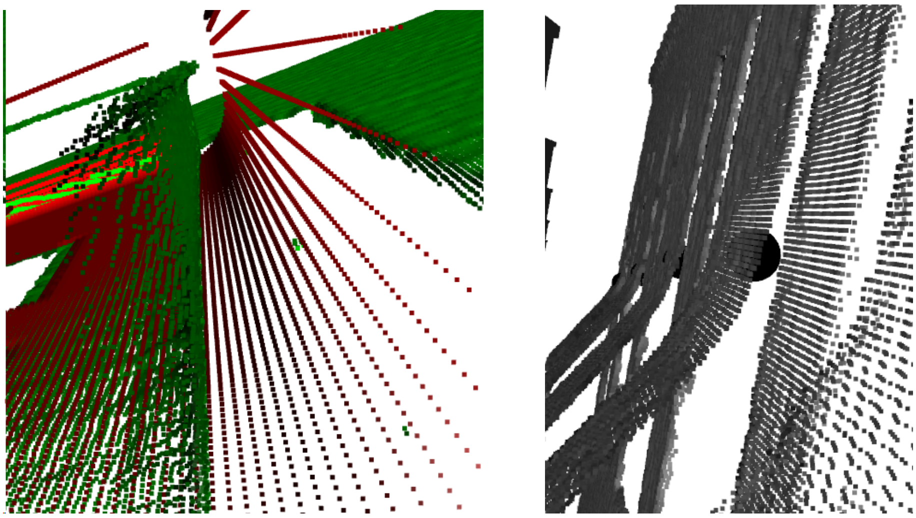
VentilationGrate				
Bend ID	CAD Angle	Measured Angle	CAD Arc Length	Measured Arc Length
1	45°	42.56 ± 0.71°	3.93 mm	1.2431 ± 0.4500 mm
2	45°	44.10 ± 0.95°	3.93 mm	1.3611 ± 0.3622 mm
3	45°	44.31 ± 2.14°	3.93 mm	1.4925 ± 0.3918 mm
4	45°	43.27 ± 0.49°	3.93 mm	1.2930 ± 0.3616 mm
5	45°	43.12 ± 0.99°	1.57 mm	1.5008 ± 0.7212 mm
6	45°	42.70 ± 1.04°	1.57 mm	1.0996 ± 0.1077 mm
7	45°	45.42 ± 0.93°	1.57 mm	1.3076 ± 0.2107 mm

Table 5.7: This table shows the measured angle and arc length for all bends in the VentilationGrate scan.

MultiBend				
Bend ID	CAD Angle	Measured Angle	CAD Arc Length	Measured Arc Length
1	90°	83.31 ± 0.84°	3.14 mm	1.3374 ± 0.6825 mm
2	35°	32.93 ± 1.75°	1.22 mm	1.4542 ± 0.2677 mm
3	35°	32.73 ± 3.33°	1.22 mm	1.4667 ± 0.3288 mm
4	35°	32.33 ± 1.79°	1.22 mm	1.2116 ± 0.4772 mm
5	35°	32.21 ± 2.41°	1.22 mm	0.9865 ± 0.6368 mm
6	35°	32.21 ± 2.59°	1.22 mm	1.4505 ± 0.7416 mm
7	35°	32.25 ± 1.46°	1.22 mm	1.9078 ± 0.0306 mm
8	15°	15.51 ± 3.05°	0.26 mm	1.1023 ± 0.3656 mm
9	30°	27.46° ± 2.11°	0.52 mm	1.4731 ± 0.6444 mm
10	45°	39.63 ± 2.83°	0.79 mm	1.4505 ± 0.7416 mm
11	60°	51.93 ± 0.24°	1.05 mm	0.7299 ± 0.4862 mm
12	75°	63.48 ± 0.83°	1.31 mm	0.5556 ± 0.2773 mm
13	90°	78.48 ± 0.99°	1.57 mm	2.0953 ± 0.0608 mm

Table 5.8: This table shows the measured angle and arc length for all bends in the MultiBend scan.

Discussion Here, the deviations between the measurements and the CAD model can originate from multiple sources, as was described in Section 2.4. Almost all measured angles are within ±5° of their corresponding CAD model specification, whereas the measured arc lengths deviate percentage-wise on average more from their respective CAD model specification. This does not mean that the inspection method incorrectly assessed the arc lengths in these scans. In fact, visual inspection of the aligned CAD models on the scans reveals that the measured arc lengths are in some cases much closer to reality than the CAD model specifications. Figure 5.2a below shows such an example, where in an otherwise low RMSE alignment from CAD model to scan, the bent surface in the CAD model (in red), requires a much larger arc to reach the same angle as the scan (in green). Figure 5.2b on the right shows an example, where the whole part of the main surface that is connected to a bend is lifted, which will cause the arc length measurement to be incorrect.



(a) Visual difference between the CAD model and the scan

(b) I am sideways, help![1]

Figure 5.2: The figure on the left shows how the arc length of the manufactured and scanned sheet metal part (shown in green) is different than its CAD model counterpart (shown in red). The figure on the right shows an example of the main surface connection to a bend being lifted off itself, which will result in an incorrect arc length measurement.

[1] Easter egg, for careful readers.

Chapter 6

Discussion and Conclusion

In this chapter, a discussion containing some reflections of the project, as well as some future work will be made. Then, a brief conclusion will be given.

6.1 Discussion

This work started with the initial problem formulation:

How can the total inclination angle, as well as the arc length of all bends be measured, when provided with a 3D point cloud that represents a sheet metal part?

To the current knowledge of the author, there do not exist any publications or articles that attempt to solve this problem explicitly. Thus, a somewhat creative solution had to be designed. Almost every aspect of the implementation that was not described in the Methods section was manually developed, and that was a lengthy process. Ideally, more experiments and tests should be conducted. This can help determine accurate relationships between the different hyperparameters in the pipeline, and make the system entire automatic, as currently, some hyperparameter tuning is still performed between scans of different point densities.

The laser manufacturing system that creates the sheet metal parts inspected in this work is designed for high-flexibility small-batch production, and the types of sheet metal parts it can manufacture are practically endless. It will be an interesting task to integrate this pipeline into such a system and have it generalize across as many sheet metal part designs as possible. There exist sheet metal parts that this system can fabricate that contain bends across their entire surface. The definition of a bend used in this work will not be sufficient to assess the quality of those parts.

6.2 Conclusion

In this project, it was determined that it is possible to measure the total inclination angle, as well as the arc length of all bends in 3D point clouds that represent sheet metal parts. The primary focus of the work is now to integrate this quality inspection pipeline in the manufacturing system, which other than the final inspection of the sheet metal parts, will provide a control loop to the manufacturing process that ensures the quality of all fabricated sheet metal parts.

Bibliography

[1] GN Nikolov, AN Thomsen, AF Mikkelstrup, and Morten Kristiansen. Computer-aided process planning system for laser forming: from cad to part. *International Journal of Production Research*, pages 1–18, 2023.

[2] Christos Kantas. Inspecting the quality of sheet-metal parts. Technical report, Aalborg Universitet, 2024.

[3] Evangelos Kalogerakis, Derek Nowrouzezahrai, Patricio Simari, and Karan Singh. Extracting lines of curvature from noisy point clouds. *Computer-Aided Design*, 41(4):282–292, 2009.

[4] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer graphics forum*, volume 37, pages 75–85. Wiley Online Library, 2018.

[5] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[6] Christopher Weber, Stefanie Hahmann, and Hans Hagen. Sharp feature detection in point clouds. In *2010 shape modeling international conference*, pages 175–186. IEEE, 2010.

[7] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9601–9611, 2019.

[8] Albert Matveev, Ruslan Rakhimov, Alexey Artemov, Gleb Bobrovskikh, Vage Egiazarian, Emil Bogomolov, Daniele Panozzo, Denis Zorin, and Evgeny Burnaev. Def: Deep estimation of sharp geometric features in 3d shapes. *ACM Transactions on Graphics*, 41(4), 2022.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] GoPhotonics. Yls-3000. https://www.gophotonics.com/products/lasers/ipg-photonics/29-152-yls-3000, 2024. Accessed on 25/06/2024.

[11] Novanta Photonics. Elephant 3-axis scan head. https://novantaphotonics.com/product/elephant/, Year not stated.

[12] KUKA. Kr 120 r2700-2 | kr c5 dualcab. https://kuka-circle.com/products/kr-120-r2700-2-kr-c5-dualcab, Year not stated.

[13] Wenglor. Mlwl153 2d/3d profile sensor. https://www.wenglor.com/en/Machine-Vision/2D3D-Sensors/MLWL/2D3D-Profile-Sensor/p/MLWL153, Year not stated.

[14] Beckhoff. C6920-0060 | control cabinet industrial pc. https://www.beckhoff.com/en-en/products/ipc/pcs/c69xx-compact-industrial-pcs/c6920-0060.html, 2024.

[15] R. Paschotta. Ytterbium-doped laser gain media. https://www.rp-photonics.com/ytterbium_doped_laser_gain_media.html, 2024.

[16] Anders Faarbæk Mikkelstrup, Georgi Nikolaev Nikolov, and Morten Kristiansen. Three-dimensional scanning applied for flexible and in situ calibration of galvanometric scanner systems. *Sensors*, 23(4):2142, 2023.

[17] SolidWorks. Homepage. https://www.solidworks.com/, 2024. Accessed on 14/06/2024.

[18] CloudCompare. Homepage. https://www.danielgm.net/cc/, 2024. Accessed on 14/06/2024.

[19] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[20] Radu Rusu, Zoltan Marton, Nico Blodow, and Michael Beetz. Learning informative point classes for the acquisition of object model maps. In *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 643–650, 12 2008.

[21] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981.

[22] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[23] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[24] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[25] Jim Lawrence, Javier Bernal, and Christoph Witzgall. A purely algebraic justification of the kabsch-umeyama algorithm. *Journal of Research of the National Institute of Standards and Technology*, 124, October 2019.

[26] Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[27] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2:559–572, 1901.

[28] Marco Zuliani, Charles S Kenney, and BS Manjunath. The multiransac algorithm and its application to detect planar homographies. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–153. IEEE, 2005.

Appendix A

Initial tests showed that applying center-to-center translation:

$$\vec{t} = \frac{1}{|P_{scan}|} \cdot \sum_{p_i \in P_{scan}} \vec{p_i} \; - \; \frac{1}{|P_{cad}|} \cdot \sum_{p_j \in P_{cad}} \vec{p_j} \tag{1}$$

followed by ICP, does not provide accurate alignment.

Two hyperparameter sweeps were conducted in order to determine if there is an optimal *maximum correspondence distance* hyperparameter for ICP to align the cad point cloud (acquired through ray casting) with the scan point cloud. The first sweep is a short-range sweep, covering values between 0.1 and 4 mm. The second sweep is a longer range sweep, designed to cover a wider range, for longer distance correspondences. The tables below report; 1. the maximum correspondence distance hyperparameter used (denoted as MCD), 2. the fitness, which is the percentage of points with correspondences, and 3. the RMSE, which is the average distance between correspondences after transformation. An effective transformation is characterized by a large fitness score and a low RMSE. Table A.1 displays the results of the short-range sweep. In the short-range sweep, there is a clear pattern, where larger correspondence distances lead to an increased fitness score, but at the same time a larger error. A simple way to explain this is; when two points are allowed to be 'matched' further away, then more points in total will be matched, but those matches are 'worse' matches. This means that the rigid transformation matrix does not work for well to reduce the distances between all the correspondences. Perhaps the correct correspondences are further away than 4 mm apart. Table A.2 shows the fitness and RMSE of the transformation for maximum correspondence distances between 1 and 150 mm. As it can be seen from the table above, the RMSE continues to grow as more points are matched. From these two sweeps, it is concluded that ICP alignment will not work on its own for aligning the CAD point cloud with the scan point cloud.

MCD (mm)	Fitness	RMSE
0.100000	0.001456	0.063864
0.316667	0.088575	0.097986
0.533333	0.098519	0.166395
0.750000	0.106534	0.238000
0.966667	0.113915	0.314097
1.183333	0.120543	0.392408
1.400000	0.126943	0.476322
1.616667	0.134648	0.582272
1.833333	0.140401	0.664964
2.050000	0.145886	0.749026
2.266667	0.151053	0.831382
2.483333	0.155738	0.911052
2.700000	0.160108	0.989993
2.916667	0.164608	1.075381
3.133333	0.168994	1.160970
3.350000	0.173126	1.244760
3.566667	0.177102	1.328530
3.783333	0.181587	1.425598
4.000000	0.187016	1.533677

Table A.1: Short-range ICP hyperparameter sweep

MCD (mm)	**Fitness**	**RMSE**
1.000000	0.115036	0.326711
7.208333	0.249622	2.921819
13.416667	0.382394	6.467658
19.625000	0.516290	10.170027
25.833333	0.635309	13.528224
32.041667	0.727789	16.300888
38.250000	0.837523	19.727735
44.458333	0.903388	22.006144
50.666667	0.945877	23.730942
56.875000	0.970386	24.936312
63.083333	0.984686	25.770180
69.291667	0.990047	26.156438
75.500000	0.996036	26.678554
81.708333	0.999789	27.057537
87.916667	0.999997	27.080734
94.125000	0.999997	27.080734
100.333333	0.999997	27.080734
106.541667	0.999997	27.080734
112.750000	0.999997	27.080734
118.958333	0.999997	27.080734
125.166667	0.999997	27.080734
131.375000	0.999997	27.080734
137.583333	0.999997	27.080734
143.791667	0.999997	27.080734
150.000000	0.999999	27.081524

Table A.2: Long-Range ICP Threshold Sweep Results