Pipeline Extraction

A master's thesis for the Department of Information and Computing Sciences

Author:
J. van Dieren

Advisors:
M. van Kreveld
A. Vaxman

Universiteit Utrecht

June 2016

Contents

Abstract

In this thesis we present an algorithm that detects and models pipelines in 3-dimensional point clouds. The algorithm starts from a user provided seed and alternates between model building and primitive detection as it tracks the pipeline. The algorithm does not require preprocessing such as registration or subsampling. When provided with a suitable seed and search area size the algorithm produces good models.

1 Introduction

Background

Large (petro)chemical plants rarely consist of only a single building or factory. Such plants are made of a large number of facilities each with their own function. For example: storage depots for raw materials or products and factories for each intermediate step in the production process. These facilities are connected to each other through an extensive network of pipes.

Motivation

It is not uncommon for the plans of such a plant to be either inaccurate due to modifications or to be unavailable entirely. An accurate map of the pipeline network is required for maintenance, emergency response and to prevent complications with future modifications to the plant. There are currently two ways to map the path of a pipeline: have a person physically follow the pipeline (often on foot) or construct a model from a point cloud of the facility.

Producing 3-dimensional point clouds for entire plants is a relatively recent development and processing this data is still done mostly manually. Creating a pipeline representation from a scan requires a user to separate the pipeline from the rest of the scan. Once the pipeline has been isolated from the cloud some parts of it may be modeled automatically. The user will still have to model most of the pipeline themselves and ensure all the parts of the model join to form a single connected pipeline.

Goal

The data used in this thesis was obtained from the Pernis scan repository, see the appendix Section B for a detailed description. The repository contains 19,638 scans of up to 80 million points each. A single pipeline can stretch over hundreds of scans. The goal of this thesis is to develop and analyze an algorithm than can track and model a pipeline across multiple scans with minimal user interaction.

We also require that the algorithm does not need any preprocessing such as the registration of multiple scans of the same scene or removing any non-pipeline points.

1.1 Related work

In Sections 1.1.1 and 1.1.2 we will briefly discuss primitive detection and segmentation methods that are often used as components of pipeline extraction algorithms. In Section 1.1.3 we will discuss several methods that attempt to extract and/or model pipelines in 3-dimensional point clouds. Common terms that are used in this section and in the rest of this work are defined in the appendix Section A.

1.1.1 Primitive detection

The two dominant primitive detection methods in use today are RANSAC, first described by Fischler et al. [6], and the Hough transform introduced by Hough et al. [7].

RANSAC

The RANdom SAmple Consensus algorithm is easily applicable to a wide variety of primitives. Most primitives can be defined by a small number of points on their surface. For example, a plane is uniquely identified by three points on its surface.

To detect a plane in the data RANSAC first has to hypothesize the specific plane that might be present. This is done by sampling three random points. Since three points define a plane the algorithm now has a hypothesis that can be tested. RANSAC will consider the hypothesized plane as detected when a lot of other points in the data support it.

RANSAC is an inherently stochastic method so a single hypothesis test is too little to conclude anything. The algorithm will draw samples, and record the highest support until it reaches some confidence threshold that it will not find a sample leading to more support. Once a plane, or other primitive, is detected it will be added to the output and its supporting points are removed from the data. The algorithm may then continue to find other primitives.

The simplicity of the algorithm has encouraged a large number of improvements and adaptations [5] [18]. We have used the Efficient RANSAC algorithm by Schnabel et al. [15] which features faster support evaluation, more spatially cohesive sampling and computes boundaries for the detected primitives.

Hough transform

The Hough transform maps data points to the parameter space of the primitive to be detected. Each data point is mapped to the whole region in the parameter space where the parameters define a primitive that is supported by the data point.

A primitive is considered detected with a specific parameter configuration if its point in parameter space is intersected by a large number of data point regions. So the problem is reduced to finding peaks in parameter space. Since finding all intersections of a large number of surfaces in parameter space is too time consuming the parameter space is discretized into cells. A point is said to vote for all cells that intersect its region in the parameter space. The discretization and the data structure to hold the discretized space are the most important factors in any implementation.

Borrmann et al. [3] use the Hough transform to detect planes in a 3-dimensional point cloud. Hough transforms scale rather poorly with the number of parameters of the primitives since the parameter space becomes much larger with the addition of a parameter. Because of this its application in 3-dimensions is mostly limited to planes and primitives with a similar number of parameters.

Rabbani et al. [13] use a two-stage Hough transform variation to detect cylinders in point clouds. They found however that large non-cylindrical structures interfered with the detection so the algorithm is not applicable to our problem.

1.1.2 Segmentation methods

Segmentation methods divide a point cloud into separate groups based on some neighborhood condition. They are useful when the data contains shapes that have no practical mathematical representation or the data is of too low quality for primitive detection methods to work.

Region growing

The most prevalent segmentation paradigm for 3-dimensional point clouds is region growing. The first instance of a region growing algorithm is described by Adams et al. [1]. In a region growing algorithm a number of points are selected as starting points for the regions. The regions are then grown by repeatedly adding eligible neighbors from the most recently added points.

Rabbani et al. [14] have developed a region growing algorithm that uses point distance, normal deviation and the residual from the normal estimation in its neighborhood condition. This allows the algorithm to separate the point cloud into smooth regions separated by distance, a sudden change in normal direction or low accuracy normals. We use this algorithm since the results in the paper indicate it should perform well in the industrial setting our data was collected.

1.1.3 Pipeline extraction

The work we will discuss in this section can be classified into two categories. The first category will attempt to extract the pipeline points from a scan before modeling the pipeline. The second category will focus only on modeling the pipeline leaving it to the user or other algorithms to remove the non-pipeline points from the scan.

The Gaussian image

Since some of the methods in this section make extensive use of the Gaussian image applied to cylinders, as introduced by Chaperon et al. [4], we will go over it briefly.

Definition 1.1 *The Gaussian image of a point set is the set of normalized normals mapped to the unit sphere. See Figure 1 for an example.*

As can be seen in Figure 1 the normals of a cylinder define a circle around the origin of the Gaussian image. This circle defines a plane and the normal vector $\vec{u}$ of this plane is the direction of the axis of the cylinder. When the direction of the cylinder is known we can project all points on the plane through the origin with this normal vector. The points belonging to the cylinder will form a circle in this plane [4].

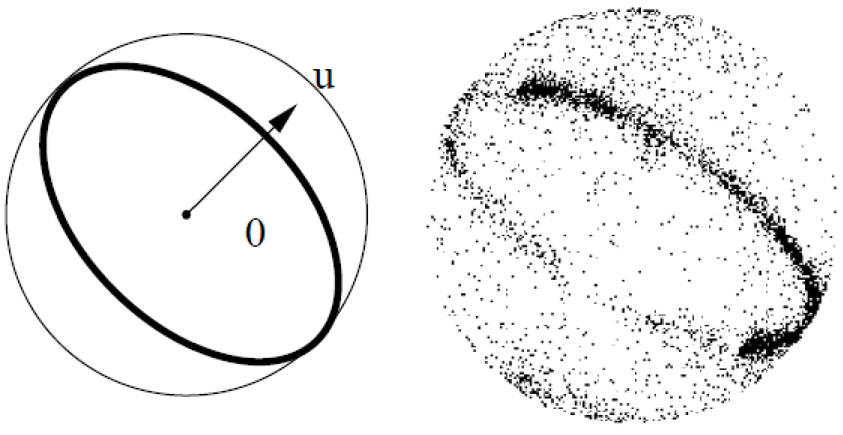


Figure 1: The Gaussian image of an ideal cylinder on the left and a real point set on the right. Taken from [4].

Extraction and modeling

Liu et al. [11] present a pipeline reconstruction method that is based on the Gaussian image. The algorithm works in three stages: detecting the ground normal, detecting cylinders perpendicular to the ground plane, and detecting cylinders parallel to the ground plane.

Once the ground normal is found a plane is fitted through all points agreeing on that normal. All the points are then projected on the ground plane. Cylinders that are vertical will form circles on the ground plane. These circles are found using RANSAC. This stage circumvents the Gaussian image by replacing it with the ground plane normal.

For the cylinders that are not perpendicular to the ground plane the authors use a Gaussian image based method similar to Chaperon et al.[4] to find more planes to map the points on to.

Once the cylinder detection stage is done the algorithm makes a limited effort to model the connectivity of the found cylinders. Elbow pieces are predicted based on a proximity threshold for the cylinders and no effort is made to recover from occlusion. The method does not provide a way to filter out false positives such as handrails and scaffolding.

The assumption that there is a flat ground plane does not hold for the data sets we have used, nor does the assumption that pipelines are either parallel or perpendicular to the ground.

The pipe-run algorithm by Qiu et al. [12] also uses Gaussian image based cylinder detection. They modify it by subdividing the data set into equally sized volumes and constructing a Gaussian image per volume. Since pipelines are often grouped into bundles that travel in the same direction the Gaussian image of a smaller volume is often more indicative of the cylinder directions than a global Gaussian image.

The algorithm handles occlusion by applying morphological opening and closing operations on the found cylinders. Like the other algorithms elbow joints are not detected directly but later reconstructed using a set of heuristics. The drawbacks of this algorithms are that it requires a point cloud of uniform density and large non-cylindrical structures can distort the Gaussian images.

Su et al. [17] use a divide-and-conquer approach to extract and model pipelines. The data set is first subdivided by an octree; then the cells of the octree are merged bottom up to form ever larger groups of points that are either classified as planes or cylinders based on their two principal curvatures. Additionally, cylindrical groups are only merged if the angle between their axes, determined by the largest vector from a principal component analysis, is within a user defined threshold.

While this method requires a large number of parameters, such as octree bin size, merging distance, and curvature thresholds the authors argue that only a few are truly sensitive to the data set. Since most of the work of the algorithm is confined to the cells of the octree the algorithm lends itself well to parallelization. This method has two drawbacks: since it does not calculate the parameters of the cylinders it cannot take domain knowledge such as diameter and length thresholds into account directly and the algorithm does not deal with occlusion.

Modeling

Once the points belonging to the pipelines have been separated from the noise and the non-pipeline surfaces, we need to reconstruct a representation of the geometry and inter connectivity of all the pipes.

Kawashima et al. [8] provide a two step algorithm to reconstruct the connectivity information after all non-pipe points have been (manually) removed. The first step uses RANSAC to detect the straight cylindrical parts. The second step uses region growing to detect elbow sections and a variety of heuristics to detect "T" and "Y" connections.

Lee et al. [9] reconstruct the connectivity information of a pipeline by first constructing its skeleton. The skeleton of a pipeline is its axis. The axis lies inside the cylinder equidistant to every point. This means that it can be approximated by a series of edges in the Delaunay tetrahedralization of the point set.

The heuristically selected Delaunay edges are not necessarily connected. To connect the skeleton edges the authors use a Laplacian smoothing to obtain the remaining skeleton points and use these points to connect the skeleton edges.

Li et al. [10] reconstruct the skeleton of the pipeline and the geometry of its surface at the same time. The method represents the axis of curved cylinders as an ordered series of vertices. With each axis vertex the surface is stored as a closed polygon around the axis. They call this data structure an arterial snake.

The algorithm first fits small straight pieces at places in the point cloud that can be approximated by a small cylinder. These pieces are then grown and merged when they meet.

The growing step is guided by proximity of points, their direction vectors and the trajectory followed so far. The direction vectors are calculated by fitting a plane through each point in such a way that the normal vectors of this point and a set of close neighbors are parallel to the plane. The direction of the point is the normal vector of the plane. All the direction vectors are then smoothed by a global Laplacian operator.

The trajectory of a snake is modeled as a cubic Bézier curve fitted through the last four axis vertices. The speed of the growth is inversely proportional to the curvature of the axis near the growing front. If the direction vectors do not agree due to a drop in point density, intersection or proximity to a non-cylindrical structure the growth direction will be based only on the Bézier curve. In all other cases the growth direction is an average of the trajectory and the surrounding points' directions.

The advantages of this algorithm are that it appears to handle occlusions and intersections well. However, the computation of the direction vectors is quite expensive so this method may not be feasible on larger data sets. Like the other algorithms in this section the algorithm assumes that most non-pipeline points have been removed. The algorithm can handle small non-cylindrical structures but the direction vectors will fail to agree on any direction if there are significant non-cylindrical structures in the data.

1.2 Contributions

The algorithms we have seen in the review in the previous section all explicitly separate building a pipeline model and extracting primitives and/or pipeline points from the data. We propose an algorithm that builds a pipeline model while it detects additional primitives. This allows the algorithm to use the information from the pipeline model in the detection step to discard false positives and to work only with data local to the pipeline instead of working on entire data sets.

The algorithms in the previous section do not allow domain specific knowledge about what constitutes a pipeline to be included naturally. This means all the algorithms implicitly share the assumption that all cylindrical structures are pipelines. Our algorithm uses a single 3-dimensional point provided by the user to obtain initial diameter and orientation values for a pipeline.

Another advantage of our algorithm is that bends are included in the detection just as cylinders are. This allows us to do away with the complicated heuristics that are used in all of the algorithms above to reconstruct them after their detection step.

Finally all the above algorithms work on a single scene consisting of either one scan or multiple registered scans. With a sprawling facility like Pernis it is unfeasible to capture the entire facility in one data set. Since our algorithm already alternates between detection and model construction, moving to a different scan in between iterations is a natural extension.

Our approach also has several drawbacks. The reliance on a user-provided seed means our algorithm is ill suited to smaller scale scenes with many short pipelines. The algorithm has been made with large scale industrial settings in mind with fewer but very long pipelines. Additionally, as we discuss in Section 4, the algorithm can be sensitive to the seed position in certain situations.

2 Algorithm description

In this section we will give a detailed description of the main algorithm and the component algorithms it uses. Section 2.1 describes the algorithm itself. Section 2.1.1 describes the data structure we use to represent the pipeline and the methods used to decide on the connectivity of the found pipeline pieces. Section 2.1.2 describes the methods used to determine which parts of the data set to investigate. Section 2.2 describes the component algorithms and how their parameter values were determined.

2.1 Algorithm overview

Initialization

The algorithm takes two parameters: a seed and a side length. The seed is a point that lies on the pipeline to be tracked, since most point cloud viewers provide a point picking function this is easy to provide. The side length is the length of one side of a cube in the data where the algorithm works. These cubes will be referred to as *search areas*, see Section 2.1.2. The first search area is constructed so it has the seed at its center. This search area is the origin cube of the search area grid. The first area is then added to the *visit stack*. We also initialize an empty pipeline graph, see Section 2.1.1, which we will refer to as the *accumulator*. The accumulator will build the pipeline graph after each detection step.

Main loop

Pop a search area from the visit stack. Perform normal estimation and RANSAC on this search area. Construct an intermediate pipeline graph from the discovered primitives. If this is the first area check if the seed lies on any primitive. If it does annotate that primitive as tracked and all primitives that are in the same connected component of the graph. If this is not the first search area merge the intermediate graph with the accumulator (see Section 2.1.1). The new pieces are annotated with tracked if they are connected to a tracked graph component.

To ensure that the region growing method only tries to grow sensible regions we generate seeds at the extreme ends of all detected primitives that have been annotated as tracked. All the resulting regions are added to the accumulator where they may or may not be connected to existing pipeline pieces.

Check the new primitive and region pieces that are tracked against the boundaries of the current search area and add any adjacent areas to the visit stack where applicable (see Section 2.1.2). If there are no more areas to visit return the accumulator.

Recovery from detection failure

There are two types of failure: RANSAC failure due to low quality data and search area failure due to either occlusion or poor quality data near the boundary.

We attempt to recover from RANSAC failure by using region growing. There must be tracked pipeline pieces in an adjacent area, otherwise this area would not have been chosen. If any of these pieces extend to the boundary of the search area, seeds at their extremities could help grow a sensible pipeline piece in the current search area. So we run the region growing algorithm on seeds generated from tracked pieces in the adjacent areas.

A search area failure is when there are tracked pieces in the current search area but none of them suggest a suitable new search area to investigate. In this case we simply add all unvisited adjacent areas to the visit stack.

2.1.1 The pipeline graph

Inspired by the work of Schnabel et al. [16] we use a graph where each node contains a detected pipeline piece to represent the pipeline. There are three kinds of nodes: straight nodes, bend nodes and region nodes. The three kinds of nodes correspond to cylindrical pipeline pieces, bend pipeline pieces and region grown pipeline pieces. The first two can be collectively referred to as primitive nodes/pieces since they contain a RANSAC generated primitive. Every node contains the support set of its pipeline piece and is either annotated as tracked or not tracked. Additionally, primitive nodes contain the detected primitive of their pipeline pieces.

Unlike Schnabel et al. [16] we do not use the relative position and orientation of the contained primitives to add semantic information about the connections since this is not necessary for tracking the pipeline.

Connecting cylinders

There are two cases in which we need to connect cylinders: either a cylinder that is continuous in the data was split over multiple search areas or part of the cylinder was occluded. The first case is trivial since the cylinders overlap at the shared boundary of their respective search areas. The second case is more involved.

We connect cylinders based on the length of the smallest connecting line segment between their (finite) axes. This segment can be decomposed into a vector along the axes and along an appropriate normal of the axes. For the checks we use the total length of the segment and the magnitude of the normal decomposition.

When two cylinder ends from parallel cylinders align perfectly we can be sure they are connected if the space in between the ends, measured as the length of the connecting segment, is insufficient for the cylinders to bend away from each other. Figure 2 shows that this requires at least the sum of the two diameters. A different way to interpret this check is to view it as a sphere with twice the diameter of the pipe centered at each axis endpoint. If two of these spheres overlap we connect the cylinders.

When two cylinder are parallel but not aligned the length of the connecting segment can be described entirely by a normal of the two axes. In the ideal case this length should be larger than the sum of their radii, otherwise the cylinders would intersect which is physically impossible. Figure 3 shows the situation.

Since RANSAC can be inaccurate in the cylinder diameter in poor quality data we consider that the cylinders are connected when the magnitude of the normal decomposition of the connecting segment for both axes is at most the minimum of the two radii. In this case both cylinders contain the other's axis so they cannot be considered as separate. Otherwise we assume RANSAC has made an error and the cylinders should remain separate.

In real data neither of these situations ever occur exactly so the connecting segment can always be decomposed in a normal and an axis length. We require that both the total and the normal length condition hold for both axes, see Figure 4 for an example of all the distances involved. In "T" splits the normal check is too restrictive, potentially discarding valid connections. In "Y" splits the total length check can accept connections where the pipes could plausibly bend away. These are edge cases however, in real data the "T" splits we should accept are caught by region growing or simply do not suffer enough occlusion to fail the checks. For "Y" splits the normal check often rejects the "Y" splits the length check should have rejected.

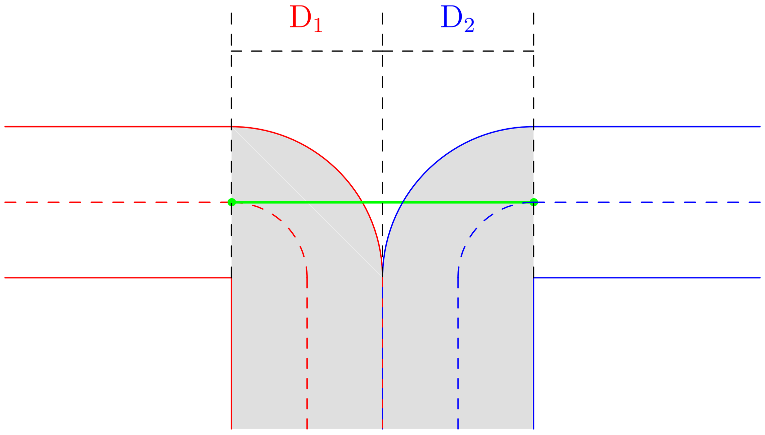


Figure 2: Two aligned parallel pipes need at least the sum of their diameters in between their detected endpoints to bend away. The connecting segment is displayed in green. The shaded areas are not detected

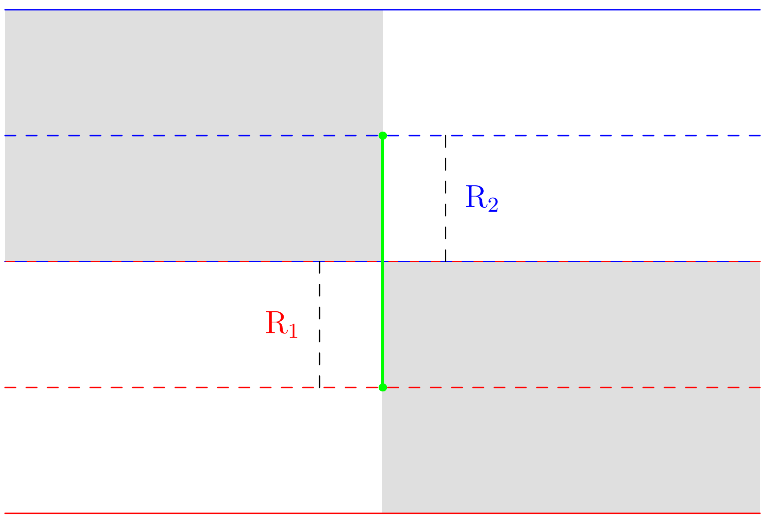


Figure 3: The endpoints of the axes of two misaligned parallel pipes need to be at least the sum of their radii apart in the shared normal direction. The connecting segment is displayed in green. The shaded areas are not detected

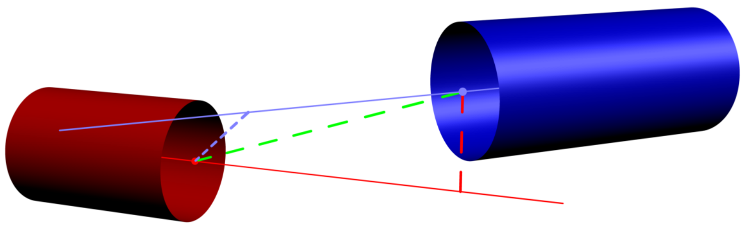


Figure 4: An example of the connection checks for cylinders. Green is the connecting segment and the blue and red dashed lines are the normals used in the checks for their respective axes

Connecting bends to cylinders

Since bends serve as connecting segments in the pipeline we only have to consider connections on the endpoints. We calculate the shortest connecting line segment from each bend endpoint to each cylinder axis and use the method described above to decide on connections. Figure 5 shows how we define a bend. To use the same checks as we use for cylinders we only have to decompose the connecting segment into a vector along an axis and a normal of this axis. We use the tangent lines of the circle at each endpoint as axes for this purpose.

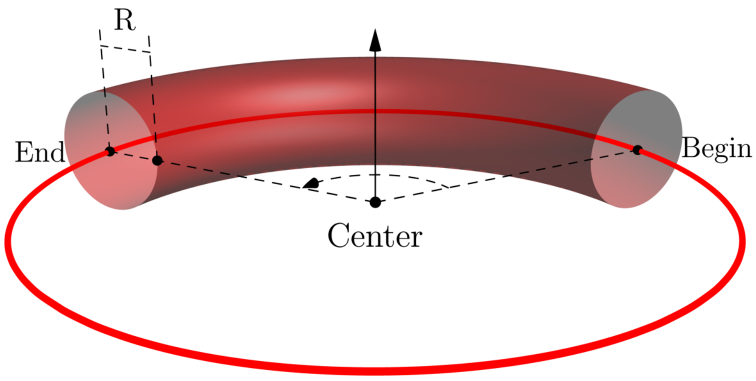


Figure 5: A bend is defined by three points, a normal, and a radius: The large circle center, the axis begin, the axis end, the normal of the large circle plane, and the radius of the smaller circle. To avoid ambiguities the rotation from axis begin to axis end is always positive (anti-clockwise) with respect to a right-handed basis where the normal is the up direction (the third basis vector).

Connecting bends

We calculate the connecting line segments between each pair of endpoints and then apply the same checks as described above.

Connecting regions to primitives

We require that at least one point of a region is considered eligible for connection. A point is eligible for connection when it lies within the RANSAC threshold of a primitive and within the same threshold from a point in the support set of the primitive. This ensures that the region is partly consistent with the primitive and that this is not just a coincidence. Without the distance to the support set check it is possible to connect a region to a primitive that is only close to the primitive due to overestimating the diameter of the bend or cylinder. We do not check the normal since a point that is both on the primitive and agrees on the normal would have been taken by RANSAC before the region growing step.

Connecting two regions
We connect two regions when at least one point neighborhood in the two regions overlaps. The n neighborhood of a point is the set of n points that lie closest to it. We construct a KD tree for each region. For every point in one of the two regions we construct its 10 neighborhood. We then query the other KD tree with this point. If the query returns a point that is closer than at least one point in the 10 neighborhood we connect the two regions. We use the same neighborhood size in the region growing algorithm, see Section 2.2.3.

Diameter threshold
In addition to all the checks mentioned above any connection decision is preceded by a comparison of the diameters of the two pieces. Empirically the largest diameter difference between two connected pipelines is 50%. This happens when a smaller pipeline joins a larger one in a "T" configuration. Primitive pieces have a diameter by default. For region pieces we start a breadth first search through the pipeline graph to find the closest (in number of hops) primitive piece and use that diameter. If the search does not find a primitive piece we ignore the check.

Building graphs
In each iteration of the main loop we build a small intermediate pipeline graph for the current search area. This intermediate graph is built by first determining the connections between the pieces as described above and then identifying through a series of depth first searches which pieces form connected components together.

Merging graphs
At the end of each iteration of the main loop the intermediate graph is merged into the accumulator. For every accumulator piece we try to connect it to every intermediate piece. Once all possible connections have been considered we recalculate all the connected components by a series of depth first searches. If any non-tracked pieces have joined the tracked component they will also be annotated as tracked.

2.1.2 Moving the search area

The search area grid
As mentioned in Section 2.1 the search area grid is initialized by a seed and a side length. The seed and side length are used to construct the first search area. This is a cube with the provided side length and the seed at its center.

This cube is the origin of the search area grid. Positions in this grid are measured in side lengths. For example, the cube at position (1,0,0) has its center one side length in the positive x-axis removed from the seed.

This grid is constructed as needed. When the algorithm requires a new search area the points are taken from the full data set. Separating the data set into smaller cubes has three advantages: it significantly speeds up normal estimation, it allows us to hide the usage of multiple data sets behind the construction of these cubes and it limits the number of irrelevant points the algorithm considers.

Adding new search areas
A search area has six planar boundaries, we can use the same logic for each of these boundaries.

For primitive pieces we extend both axis ends with a length equal to the radius of the primitive. We then move a sphere with the radius of the primitive along the axis, if it intersects a boundary at any point we add the search area that shares the boundary to the visit stack. We do not add areas that have been visited previously. If for example we perform this check at the origin and the ball intersects the boundaries in the positive x axis, the positive y axis, and the negative x axis we add the areas at coordinates (1,0,0), (0,1,0), and (-1,0,0).

Since primitives are usually not axis parallel it is possible we lose track when we only consider axis parallel movements for the search area. In the previous example it is entirely possible the primitive has very little presence in any of the mentioned areas. To solve this we also add the sum of the movements the visit stack. We avoid summing movements that cancel out so in the previous example we also add (1,1,0) and (-1,1,0).

For region pieces we estimate an axis to work with based on the order in which the points are added by the region growing algorithm. Since the region growing algorithm adds points in a breadth-first manner the first points are at the beginning of the region and the last points are at the end. To estimate the axis we take the half-way point of the region and the last point. Like the primitive pieces we move a ball with the piece's radius along the axis.

2.2 Component algorithms

The three component algorithms we use for detection are normal estimation, RANSAC, and region growing. These algorithms all have a number of parameters, however these do not need fine tuning since we have found a set of values that performs well on a large large variety of point densities and sensor noise levels.

Since we assume data sets that are produced by a single scanner point density and sensor noise depend on the distance to the scanner. To get an idea of the performance of the algorithms at different distances we have created three artificial point clouds. The three clouds model conditions at 15, 50, and 100 meters from the scanner and are named near, middle, and far. Each cloud was generated by placing points on the surface of three cylinders and displacing them with the appropriate noise value. We decided not to simulate an actual scanner since this would introduce variables such as scanner position and orientation the effects of which we are not interested in at this point. For more detailed descriptions of the experiments and their results see the appendix Section C.

2.2.1 Normal estimation

Since both the region growing method and RANSAC require normals to function and our data does not include normals we need to estimate them ourselves. We use the method described by Rabbani et al. [14].

The neighborhood size has to be chosen to be large enough to be accurate yet small enough to prevent a strong smoothing effect. We have chosen a neighborhood size of 35 points. The authors of [14] use 30 points themselves however, as can be seen in Figure 6, 35 points gives better accuracy for the near and middle data sets.

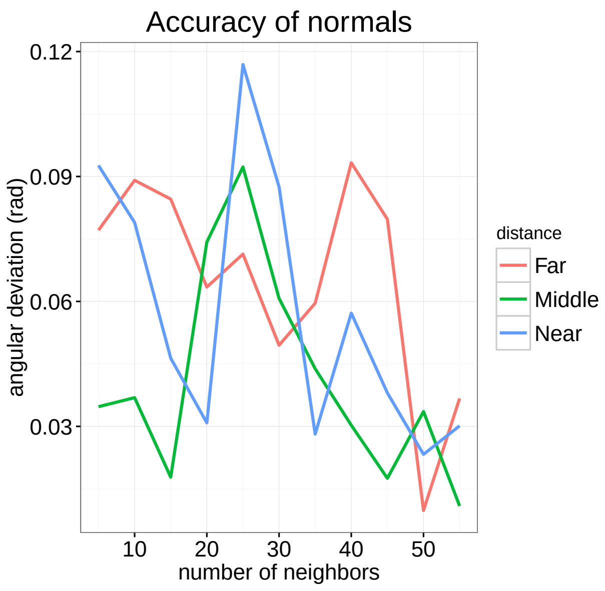


Figure 6: The average accuracy of the normal estimation. 95% confidence intervals are too small to show

2.2.2 RANSAC

We use the RANSAC variant by Schnabel et al. [15]. In broad terms the parameterization of the RANSAC algorithm needs to balance the detection of false positives with failure to detect the true positives. A restrictive parameterization will detect fewer false positives but will fail to detect the true positives more often. A permissive parameterization will detect more false positives but will not fail to detect the true positives as often. For a more detailed description of the parameters of the RANSAC algorithm see the appendix Section C.2.

Since we have a pipeline model to help filter out false positives we aim for a parameterization on the permissive side. We use a very low support threshold of 3%. We use a rather restrictive normal threshold of 0.15 radians to ensure the accuracy of the detected primitives. The normal estimation experiments also indicate this should tolerate all estimation inaccuracies. The distance threshold is set at 36 mm since this allows for quite significant sensor noise without having a significant impact on the accuracy of the results.

The bitmap threshold, the largest gap there may be in the support set of a primitive before it is split, is set at the higher end of the 95% confidence interval for the interpoint distances in the current search area. This value ensures no primitives are split unnecessarily and allows our algorithm to deal with occlusion.

The optimum probability threshold is set at 0.001. Every time a sample is drawn the probability that the best sample has not been drawn is lowered. This threshold determines at which probability the algorithm stops drawing new samples.

2.2.3 Region growing

We use the region growing method described by Rabbani et al. [14]. The algorithm is tested on data very similar to ours and our experiments also showed that their recommended parameters worked well. We use a normal threshold of 0.26 radians, a residual threshold in the 98th percentile. The only parameter where we deviated from the recommended settings is the neighborhood size. A larger neighborhood makes it easier for the algorithm to jump around in poor data. Since we start our regions from trusted seeds we want to limit the algorithm's ability to make large jumps away from these seeds. We use a neighborhood of the 10 closest neighbors for every point. For a more detailed description of the parameters of the algorithm see the appendix Section C.2.

3 Experimental setup

3.1 Ground truth generation

Since there are no accurate models available to use as a ground truth we will use the algorithm itself to generate these.

To generate a ground truth we use a fine tuned seed position and search area size for a specific situation and run the algorithm. We have to verify visually that the resulting model is good enough to be used as a ground truth.

The main drawback of obtaining ground truths this way is that we can only evaluate situations where our method is able to produce high quality models given sufficient fine tuning. This method also prevents us from evaluating errors that persist even after fine tuning.

In essence the experiments using these ground truths tell us how robust the algorithm is against different settings and not how accurate it is in modeling the real truth.

3.2 Parameters

The algorithm takes two parameters: the seed and the search area size. The size of the search area directly affects the performance of the RANSAC algorithm. An area that is too small may not be able to fit the radius of the primitive that RANSAC returns. This is a problem since the detected primitive is guaranteed to intersect several boundaries where each neighboring area is likely to detect the same bit of pipeline again. An area that is too large may contain so many non-pipeline points that the actual pipeline struggles to reach the support threshold leading either to a detection failure or detecting the wrong primitive. Like the RANSAC algorithm the region growing algorithm may struggle to reach its support threshold in larger search areas.

The seed position determines where the first detection round is run. Both the area size and seed position contribute to the positions of the boundaries of the search areas. If the boundaries are placed so complex situations are split over multiple areas, the tracking step of the algorithm will be more prone to mistakes.

Parameter values

Seeds are taken randomly from the support cloud of each ground truth. This cloud is first subsampled to ensure its point density is uniform. We will generate 10 seeds from every ground truth.

We will take the side length of the search cube as a measure of the search area size. We will take lengths from 1 to 5 meters with a step size of 0.5 meters.

For every combination of seed and side length we will run the algorithm 20 times and compute the average performance.

3.3 Analysis methodology

The goal of the analysis is to establish how the parameters and input of the algorithm affect the accuracy of the models it produces. Additionally we aim to give some guidance on how to choose the parameters and describe some of the behavior demonstrated by the algorithm in the detection process.

We will not consider the parameters that the RANSAC and region growing methods require. Empirically it appears that the chosen values for these parameters perform well for a wide range of point densities. There are a total of 7 additional parameters for these methods which makes a thorough study of their effect on the algorithm a daunting task.

We separate the analysis in a quantitative and a qualitative part. The quantitative part aims to fulfill the first two goals, the qualitative part addresses the last goal.

3.3.1 Quantitative

Accuracy

We have two accuracy categories: false positives and false negatives. If we have a run where neither occurs the constructed model matches the ground truth perfectly.

Both the ground truth and the model are made of pieces that contain a primitive and pieces that only contain a region grown point set. To calculate the false negatives we need to determine which pieces of the ground truth have no corresponding pieces in the detected model. Similarly, to calculate the false positives we need to determine which pieces in the model have no corresponding pieces in the ground truth.

Matching primitives

To match two cylinders we calculate the distance between the two axes. If the difference is less than 36 mm (the RANSAC threshold) we consider them matched.

To match a cylinder and a bend we check if either endpoint of the bend lies within the RANSAC threshold of the cylinder axis, or if either of the endpoints of the cylinder lies withing the RANSAC threshold from the bend axis.

To match two bends we use the same endpoint-based check as the previous paragraph.

Matching regions

Two regions are considered matched if their point sets overlap in at least one point. A region grown piece is considered matched to a primitive piece on the same condition.

Primitive measures

We measure the false positives and negatives of the primitive pieces in meters.

Since both primitive pieces have axes we can measure how well a piece is matched by how much of its axis is 'covered'.

To calculate the coverage of a piece we project the axes of its matches onto its axis. We then scan the axis from begin to end and record a distance as covered when at least one axis is projected on to it.

For region grown matches we sort the points according the relevant parameter of the axis and select the minimum and maximum value as begin and end point to project on the axis.

The combined uncovered axis length in the ground truth is a measure of the false negatives, in the detected model it is a measure of the false positives.

Region measures

Since regions have no primitives to measure any geometrical properties with we can only report how many points in the region overlap with the support sets of its matches.

The number of unmatched points in the ground truth is a measure of the false negatives, in the detected model it is a measure of the false positives.

3.3.2 Qualitative

Since the quantitative evaluation can only comment on the performance of the algorithm relative to a known good result of itself we need a qualitative analysis to make some observations on the absolute accuracy of the algorithm, what kind of errors it encounters and what factors may affect it.

4 Quantitative analysis

In this section we will examine the performance of the algorithm with parameters and ground truths as described in Section 3. For all the figures in this section that show point clouds the blue parts are classified as cylinders, green as bends and the red parts are region grown.

4.1 Cross pipeline

This particular pipeline, see Figures 7 and 8, is interesting since it features a substantial variation in point density and geometric complexity. The point density is highest near the middle, see seed 1 in Figure 7, and drops off near the ends. The area of the pipeline near seed 10 features a number of bends in quick succession combined with a lot of surrounding structures.

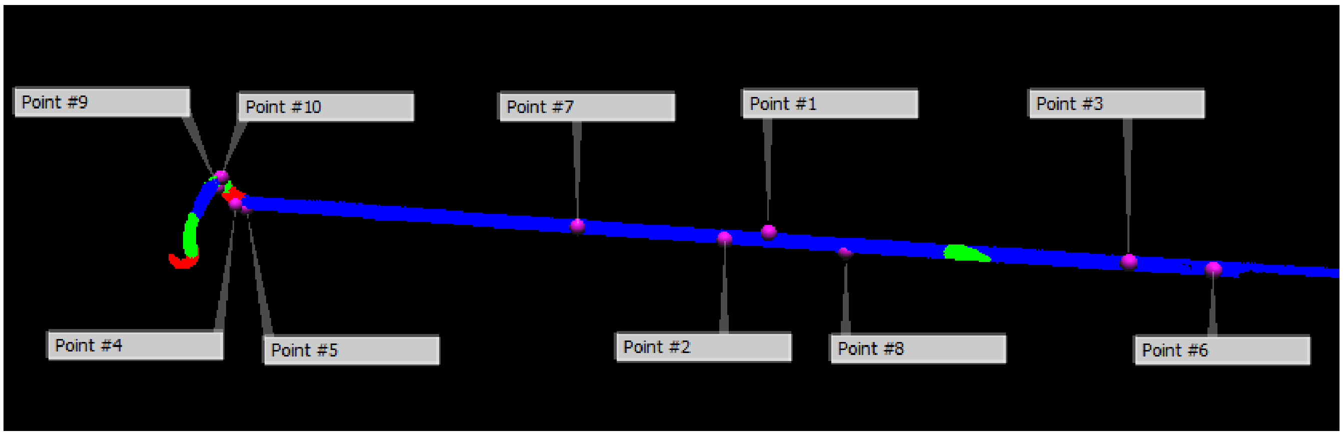


Figure 7: Seeds for the cross pipeline data set.

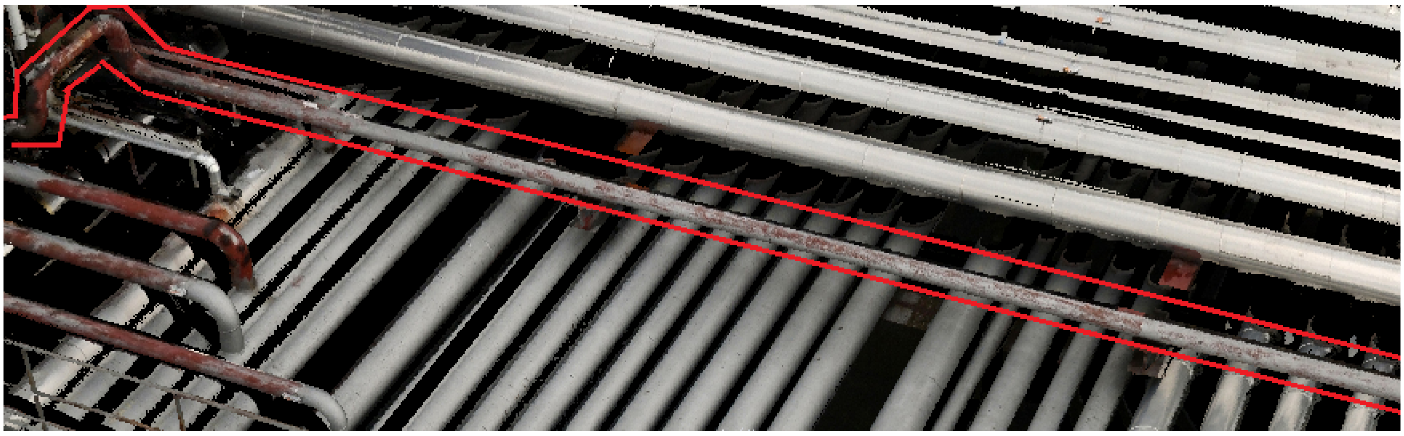


Figure 8: The cross pipeline in its full data set.

4.1.1 Primitive pieces

False negatives

Figure 9 shows the percentage of the total pipeline length that could not be matched by any primitive pieces in the model.

There are a number of observations we can make from this figure. Firstly it appears that all seeds perform well, if not at their best, with a side length of 1.5 meters. Secondly there are a number of seeds that fail quite quickly after that and a number of seeds that continue to perform well regardless of the side length. In essence the algorithm will perform well if either a good seed or a good side length is chosen.

The poor performance of seeds 4, 5, 9, and 10 is due to the high complexity of their surroundings. Their pipeline section is curved, low density and surrounded by other structures. The bends cause RANSAC to split the pipeline into smaller primitives which fail to meet their 3% support threshold very quickly once the larger search areas begin to include more of the surrounding structures. When RANSAC fails in the first search area the algorithm cannot continue.

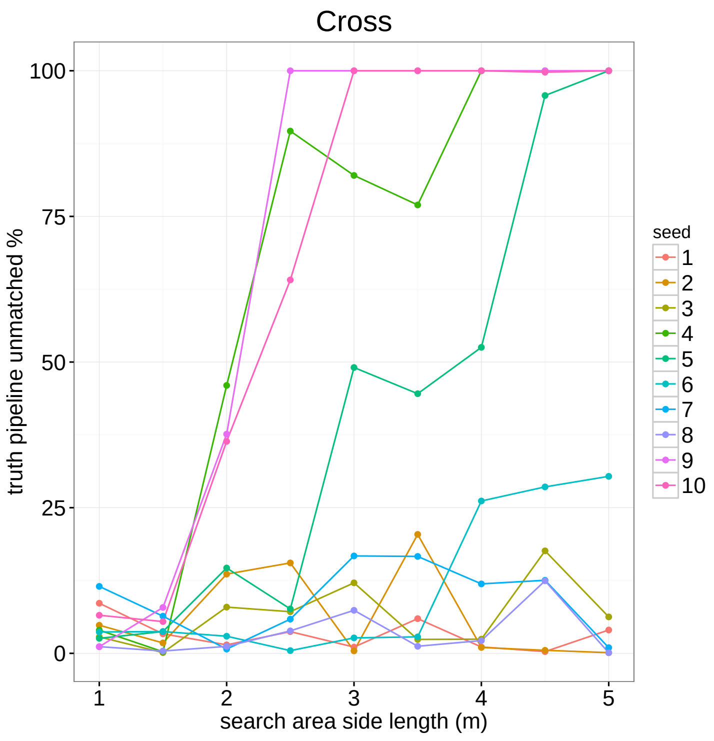


Figure 9: The false negatives for the primitive pieces.

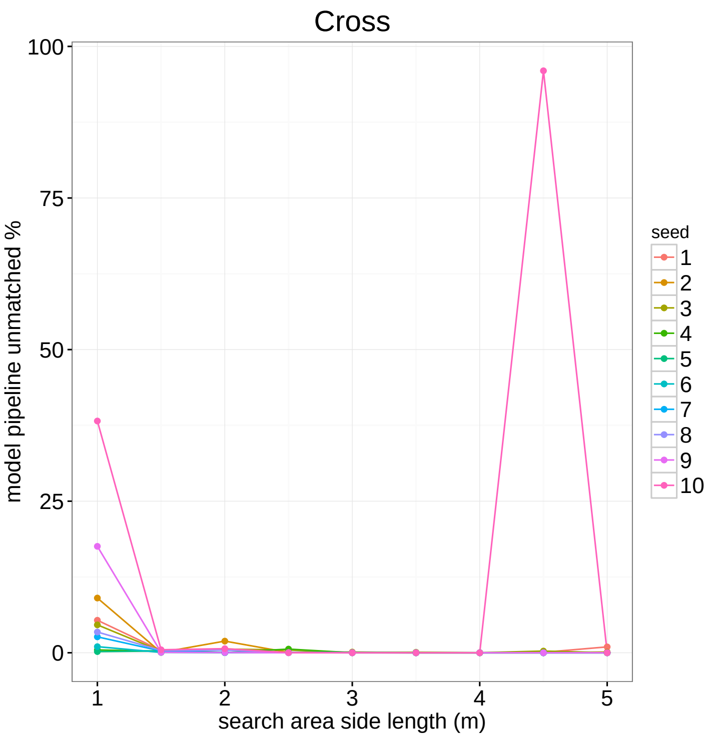


Figure 10: The false positives for the primitive pieces.

False positives
Figure 10 shows the percentage of the total model pipeline length that had no corresponding pieces in the ground truth. Note that the percentage reported in this plot is relative to the constructed model so large peaks may not correspond to large lengths.

The lack of surrounding pipeline pieces gives the algorithm very little opportunity to generate false positives. The large peak at 4.5 m in Figure 10 is due to RANSAC detecting a very small, 14 cm in length, false cylinder that just happens to include the seed.

The peaks at 1 meter are due to the search area size. A small area also means a low support threshold which makes it more likely that a false positive is detected and included in the model. The small size also means that any included false positive is more likely to cause the algorithm to investigate adjacent search areas.

4.1.2 Region pieces

Figure 11 shows that the behaviour of the region growing for this data set is very chaotic. This is due to the regions of the ground truth being nonessential to the pipeline, they are small enough for their neighboring pieces to be connected without them. Additionally the regions lie in an area with both low point density and high curvature which makes it difficult for the region growing algorithm to perform consistently across different seeds. Since the regions are nonessential, detection can continue regardless of which fractions of the two regions are grown. If the regions were essential all the variation would likely be well below the 50% since failure for one region would mean not reaching the other.

The high variation seen in the region grown pieces does not carry over to the primitive pieces in Figure 9 for the same reason: a region error has no effect on the following detection steps. The difficulty of the data also explains the absence of false positives for the region growing as seen in Figure 12. Only the small areas give any false positives since their small size contributes to a very low support threshold.

4.2 Climb pipeline

The main interesting feature of this pipeline, see Figures 13 and 14, is the vertical segment. This segment begins with a reasonably high density straight part and then continues into the surrounding support structures with a few bends. The vertical segment also features a large gap due to occlusion.

The regions on the side of the horizontal section are due to a combination of unusually high sensor noise, support structures that are not occluded, and high density. These region are at this point unavoidable.

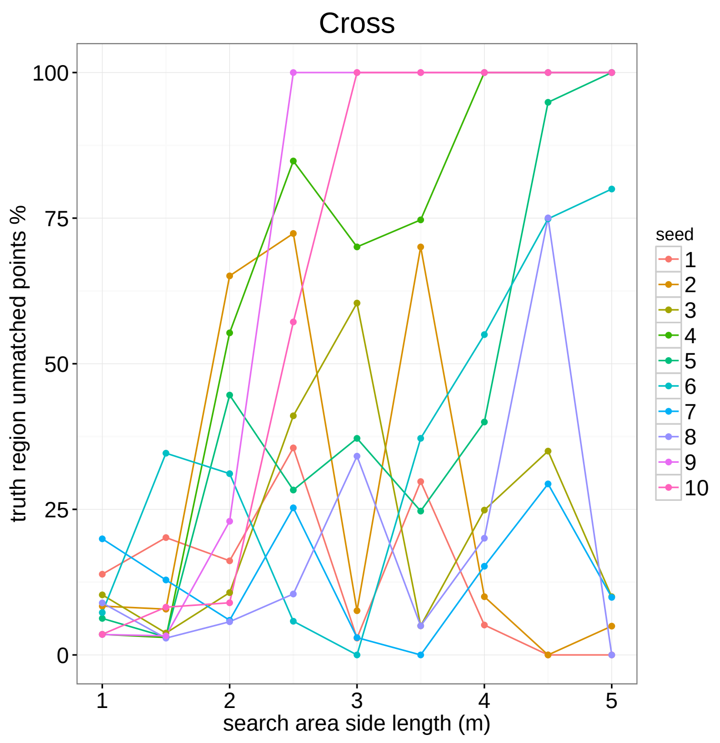


Figure 11: The false negatives for the region pieces.

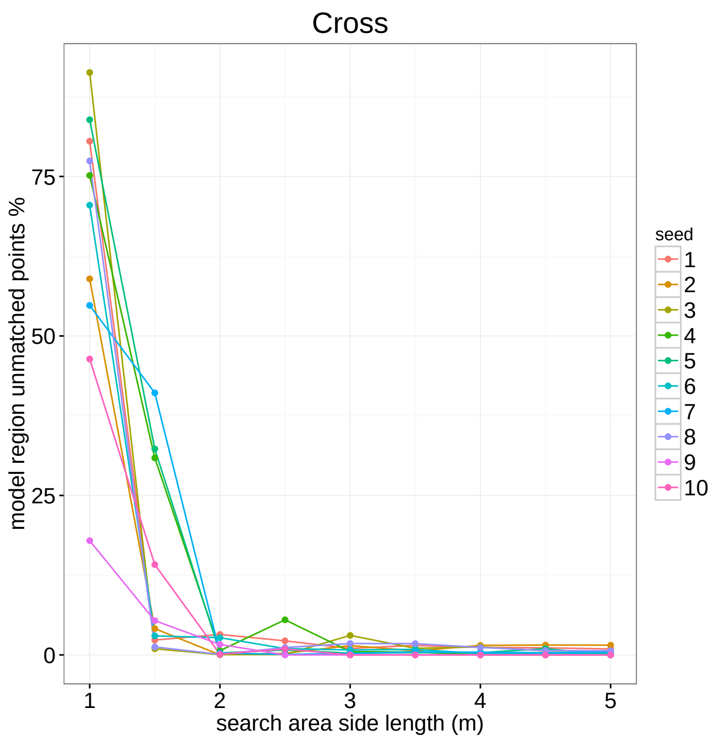


Figure 12: The false positives for the region pieces.

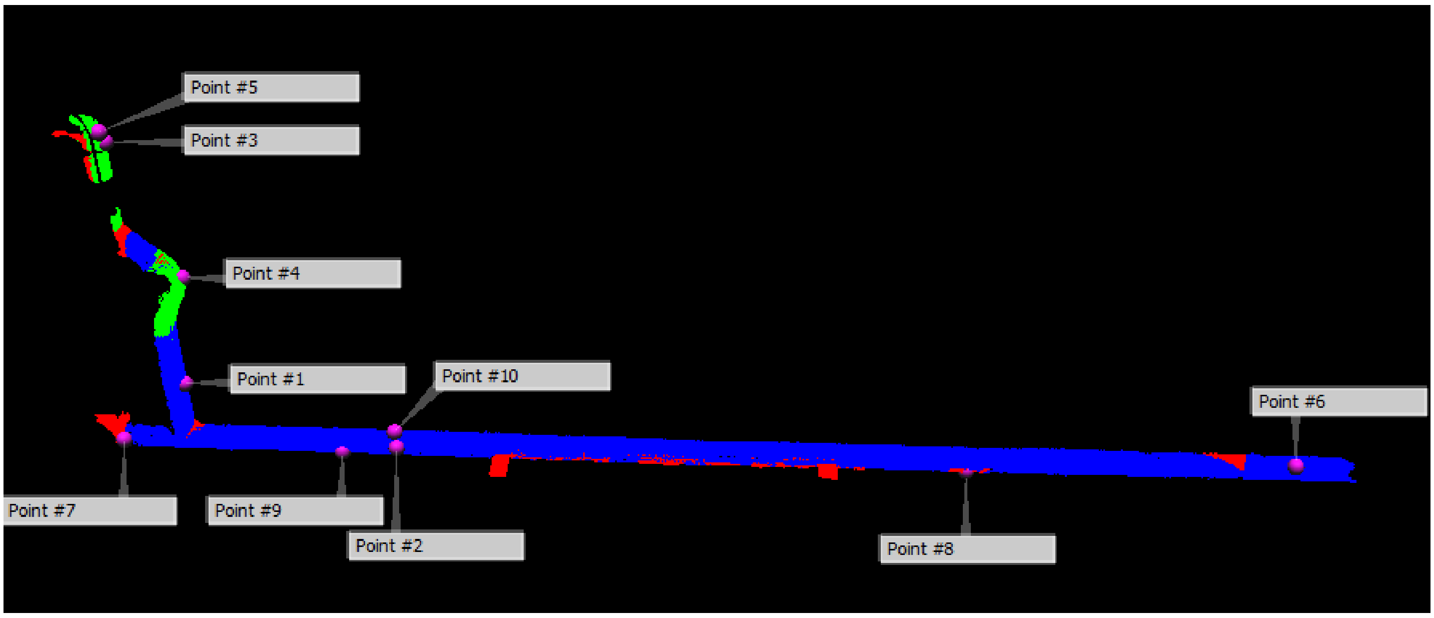


Figure 13: Seeds for the cross pipeline data set.

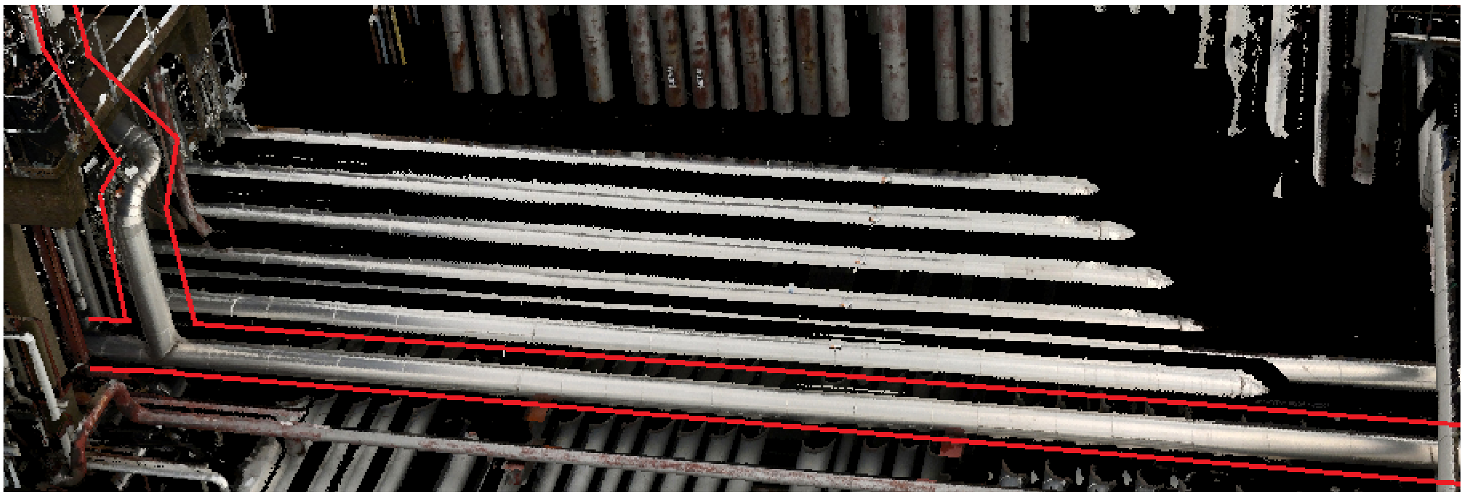


Figure 14: The climb pipeline in its full data set.

4.2.1 Primitive pieces

False negatives

Figure 16 shows the percentage of the total pipeline length that could not be matched by any primitive pieces in the model.

Similar to the last data set there appear to be some seeds (3, 4, and 5 in Figure 16) that perform poorly on all side lengths while the other seeds perform well on all but the largest side lengths.

The poor seeds again lie on the most difficult section on the pipeline as can be seen in Figure 13. The degradation in performance that all seeds show at the 5 meters side lengths is due to none of the pieces in the vertical pipe meeting the support threshold. The false negative rate remains beneath the 50% since the simpler horizontal part of the pipeline remains easily detectable.

Seed 6

The remarkable performance loss we see for seed 6 with a side length of 4 meters is due both the size of the search area and two unfortunate boundary positions. In Figure 15 the red region on the left side marks the location where both RANSAC and region growing struggle to reach the support threshold. A large search area here picks up a lot of points from neighboring pipes, making the tracked pipe a rather small percentage of the total points. Additionally the search area boundaries are placed so only a small length of tracked pipe is in the corner of the search area. The only way the region growing can reach the threshold here is to include the small protrusion that is technically not part of the pipeline.

Even when the detection makes it past this hurdle the search area boundary cuts the green bend off so it appears to point right instead of up. This causes the algorithm to investigate the wrong search area.

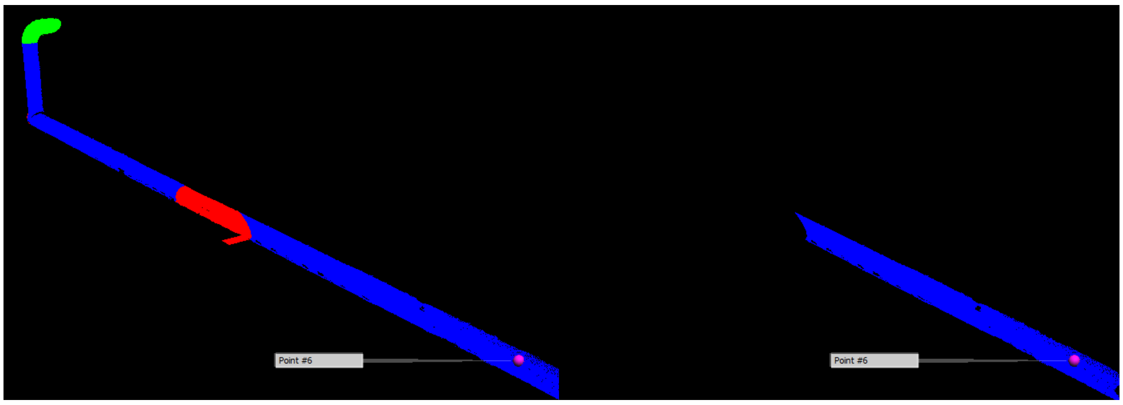


Figure 15: The two possible outcomes for seed 6 with a side length of 4 meters

False positives

Figure 17 shows the percentage of the total model pipeline length that had no corresponding pieces in the ground truth. The large peaks seen in this plot do not correspond to large lengths. The percentage is relative to the constructed model so small models will easily produce large peaks.

This shows behavior similar to the cross data set in Section 4.1.1 as well. The only difference is that the bad seeds do show false positives unlike they did in the cross pipeline. The bad seeds in this pipeline were surrounded by much closer supporting structures which makes it more likely for a false positive to end up with a seed.

The only significant peak is seed 3 at 2.5 meters. Not only is the model completely composed of false positives, it is also 4 times as long as the ground truth. This is due to the seed being surrounded by other structures in a low density area. The detection starts off with a false positive and takes off from there.

4.2.2 Region pieces

The region pieces, Figures 18 and 19 show a far less chaotic picture than the cross data set. This is due to the majority of the region points being located on or near the easier horizontal section in Figure 13. The regions on the more challenging vertical section are very small in comparison.

The higher variability in both the false positives and the false negatives for the region pieces compared to the primitive pieces is due to the majority of the regions actually being located on a primitive instead of growing away from the end.

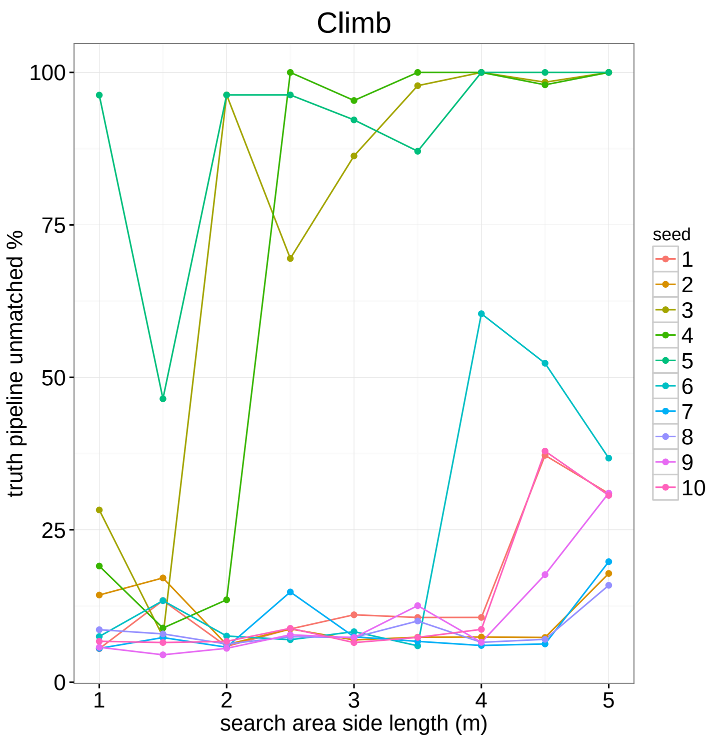


Figure 16: The false negatives for the primitive pieces.

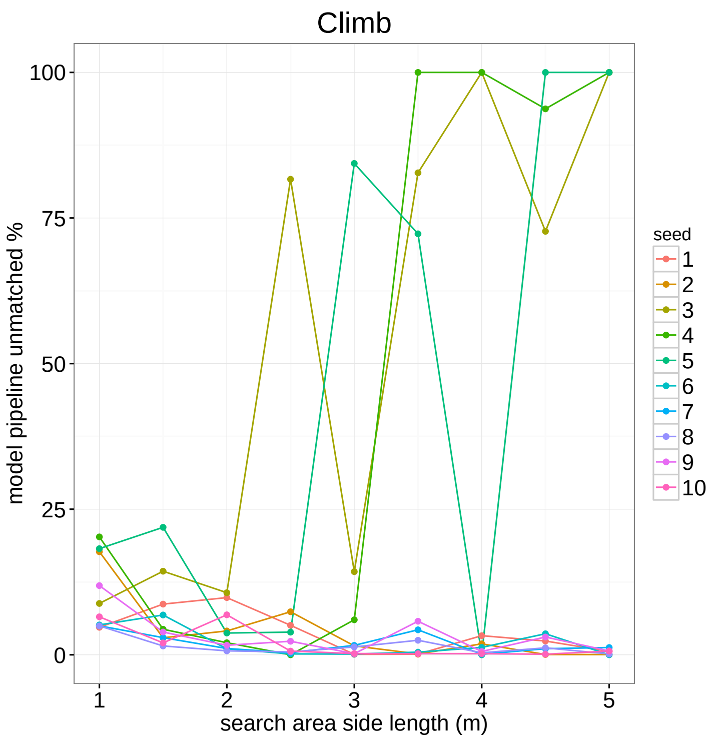


Figure 17: The false positives for the primitive pieces.

Even a minute diameter or orientation difference in the primitives that RANSAC produces will have a noticeable effect on both the starting seed choices and the available points for the region to grow into. This is also the reason that no seeds ever get to 0% false negatives, there are always some points missed on the side of the horizontal section in Figure 13.

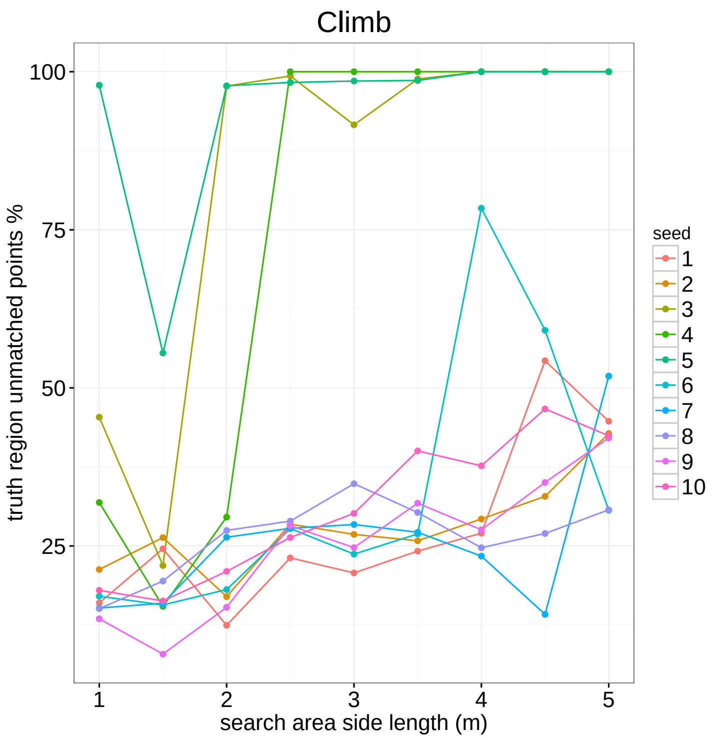


Figure 18: The false negatives for the region pieces.

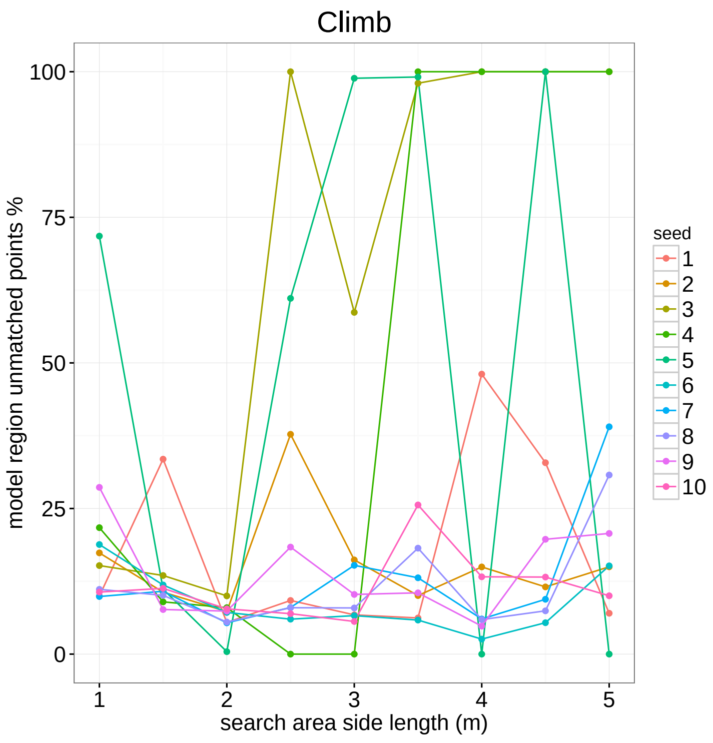


Figure 19: The false positives for the region pieces.

4.3 Join pipeline

The join pipeline, see Figures 20 and 21, is interesting since it has a section that is flanked closely by adjacent parallel pipelines (the horizontal section in Figure 8). The section is then joined by a pipe with a much smaller diameter which results in a complex connection of two cylindrical pieces, a bend and a region in a very small space (just to the right of seed 10 in Figure 20).

4.3.1 Primitive pieces

Figure 22 shows a different behavior from the previous two data sets. In this case there are no clear good or bad seeds.

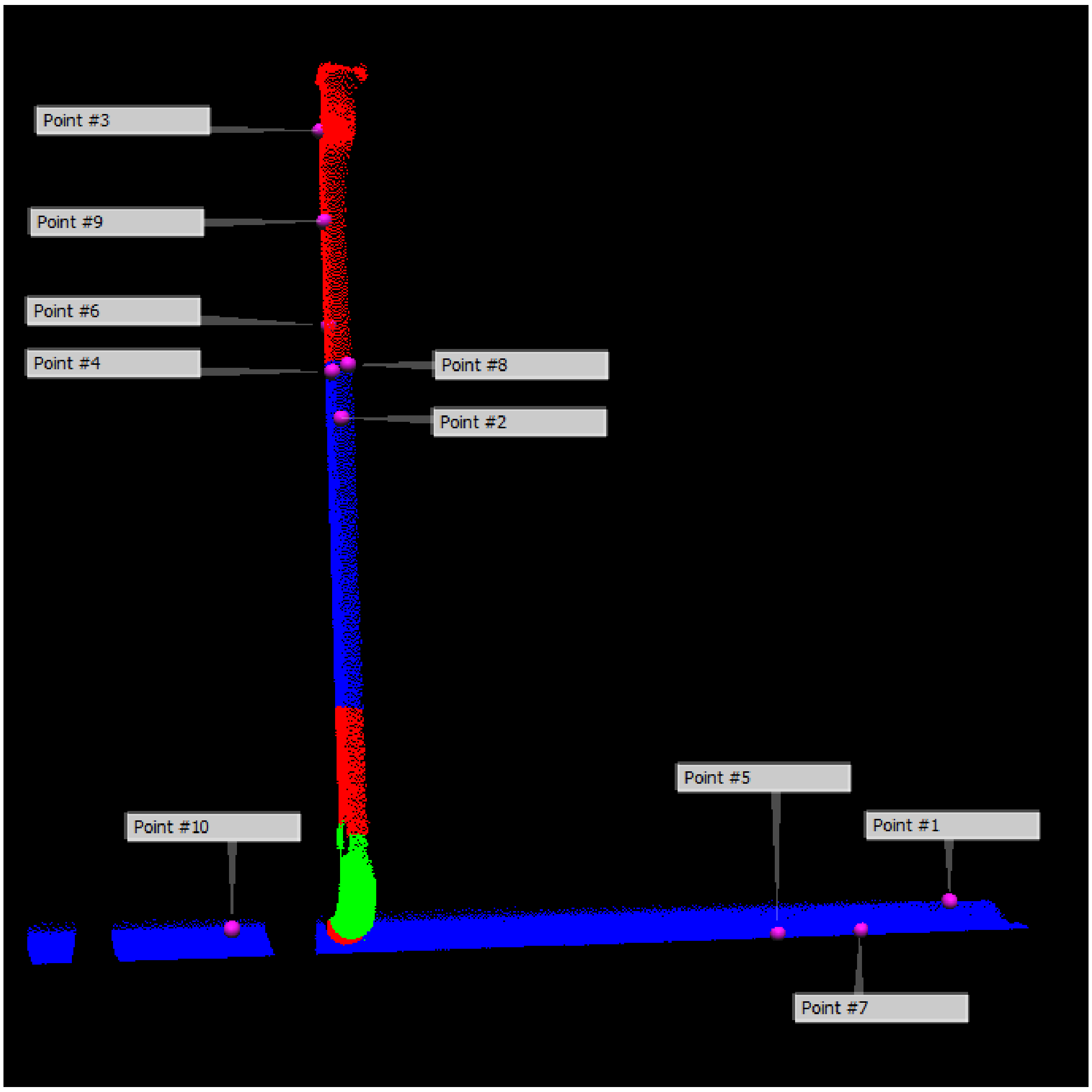


Figure 20: Seeds for the join pipeline data set.

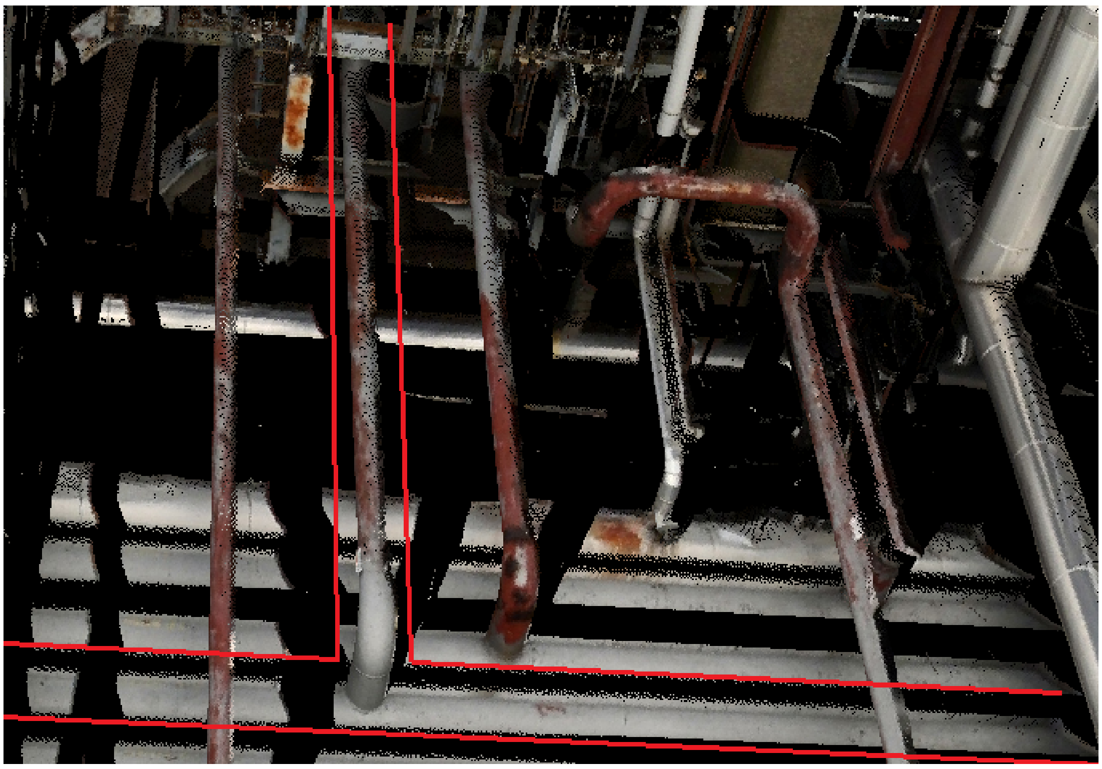


Figure 21: The join pipeline in its full data set.

For this data set it appears that all seeds are rather sensitive to the side length of the search area.This is due to the data set lacking any 'easy' starting areas. The best performance can be found in the 2 to 3.5 meters range.

The horizontal pipes in Figure 20 are flanked closely by parallel pipes as can be seen in Figure 21. This means that even though their point density is high they have very limited non-occluded surface area, which makes detection more difficult.

The vertical pipe in the figure presents different challenges. Even though it does not suffer from occlusion it has a rather low point density. The green bend piece cannot grow with the search area like cylinders can, which means that it will fall below the minimum support threshold on the larger side lengths.

All these issues combined mean that this data set has no safe choice for a seed position. The lack of consistently good seeds also means that the search area size has to be chosen carefully to work with the seed as well, making this a very difficult data set.

Seed 3

Seed 3 is the first to reach total detection failure. This is due to it being in the lowest density area of the scan. RANSAC will perform, albeit unreliably on the smaller search areas. As they grow larger we see false negatives and false positives rise. The false positives rise since the real pipeline will get less likely to reach its support threshold while false positives arise from the neighboring structures due to poor normal quality.

The high false positives shown in Figure 23 are due to the total model size being very low. Even the highest peak amounts to less than 1 meter. The same goes for the peak of seed 10 at 4 meters.

Seeds 2,4,8,6

Its interesting to note that seeds 2,4,6,8 all rapidly rise in the number of false negatives around the 4 meters. These seeds have in common that they lie on a transition from point density that is just sufficient for RANSAC to a density that RANSAC cannot handle. The cylinder that the seeds lie on can not grow with the search area since on one side the search area grows into low quality data. This means the cylinder is less likely to reach the support threshold in the larger areas.

4.3.2 Region pieces

As seen in Figure 24 the region pieces behave very much the same as the primitive pieces. This is to be expected since in this data set the region pieces actually make up a significant portion of the total pipeline. This supports the hypothesis from the cross pipeline that the chaotic behavior of its region grown pieces stems not only from the difficult data but also from their nonessential role in the model.

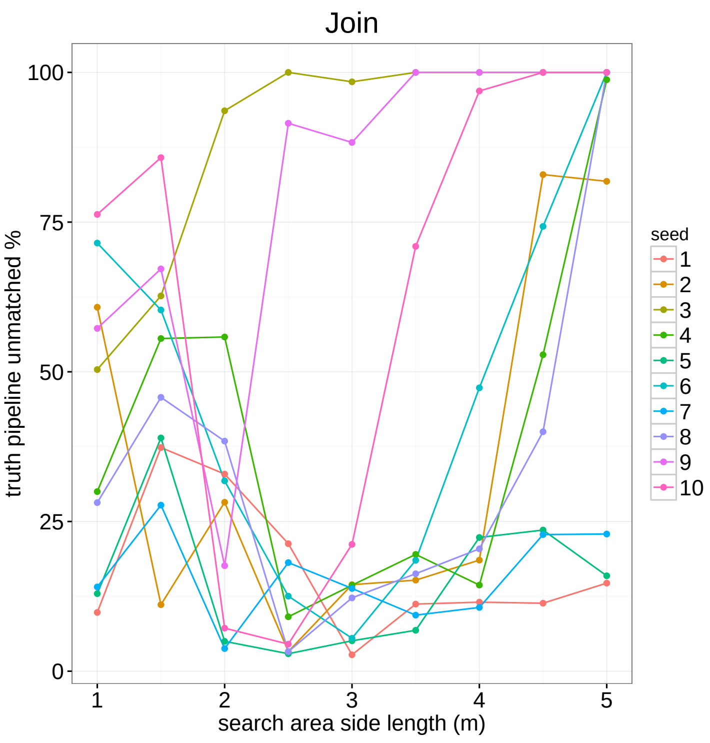


Figure 22: The false negatives for the primitive pieces.

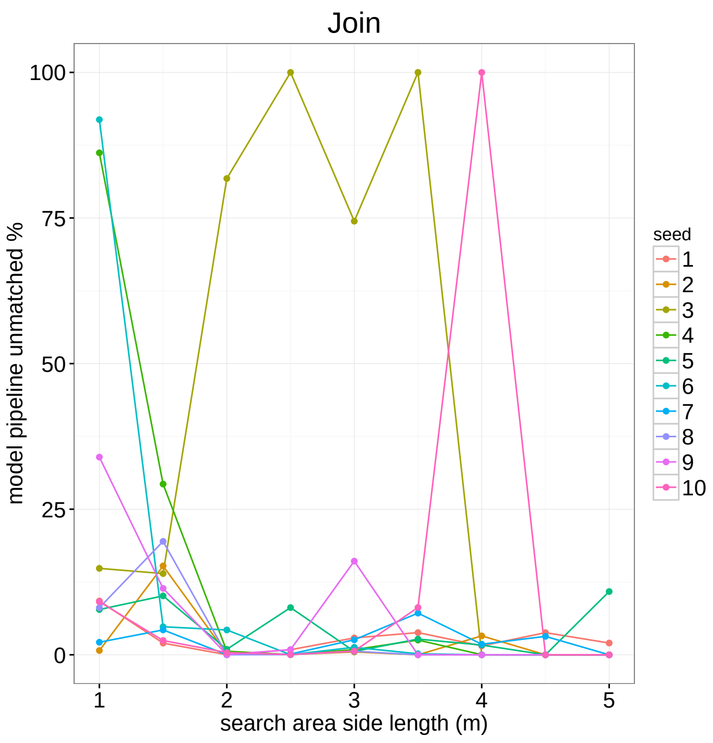


Figure 23: The false positives for the primitive pieces.

The peaks we see in Figure 25 are again due to the model containing very few region points, which allows just a few points to cause a 100% peak.

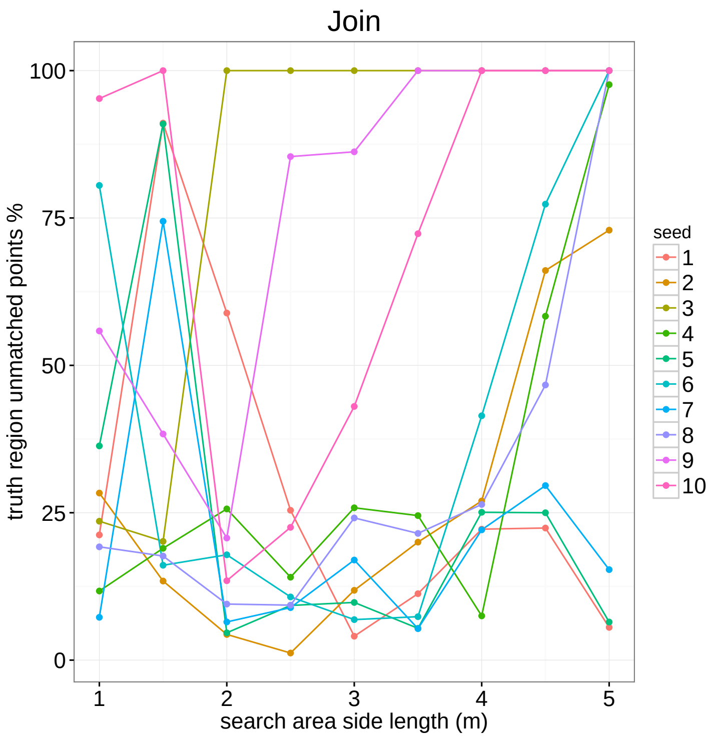


Figure 24: The false negatives for the region pieces.

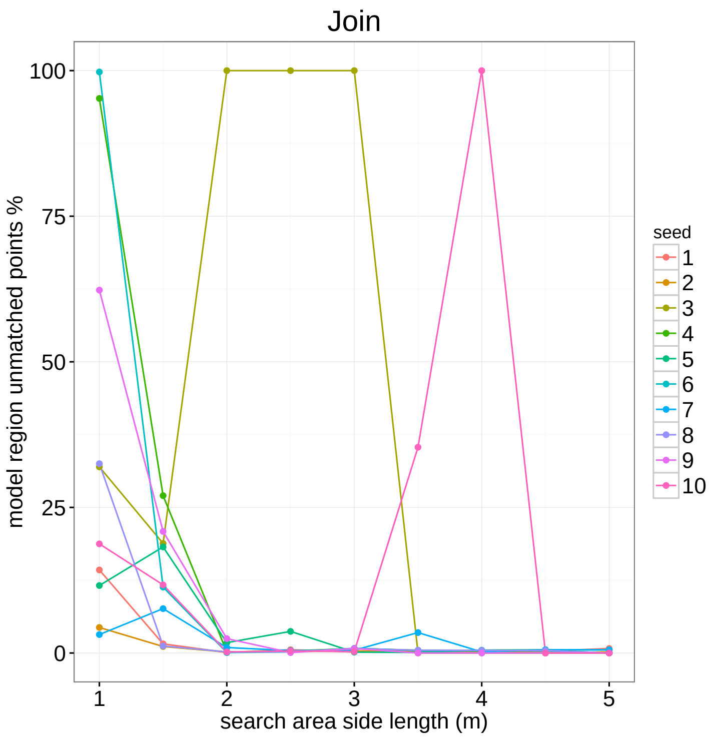


Figure 25: The false positives for the region pieces.

4.4 Bend pipeline

The bend pipeline, see Figures 26 and 27, features the highest change in point density out of all data sets. It is flanked by similar pipelines for its entire length. The left of the pipeline in Figure 27 is only 2.5 meters removed from the scanner which gives it a point density of approximately 75064 points per m^2 of cylinder. The far right is 23 meters removed from the scanner and incoming laser rays have a very large incident angle resulting in a density of approximately 250 points per m^2 of cylinder. That RANSAC manages to detect primitives for both these extremes is a testament to its robustness under the chosen parameter values.

4.4.1 Primitive pieces

Figure 28 shows a similar picture to the join pipeline in Figure 22. There are no universally good seeds or side lengths however, side lengths in the 1.5 meters to

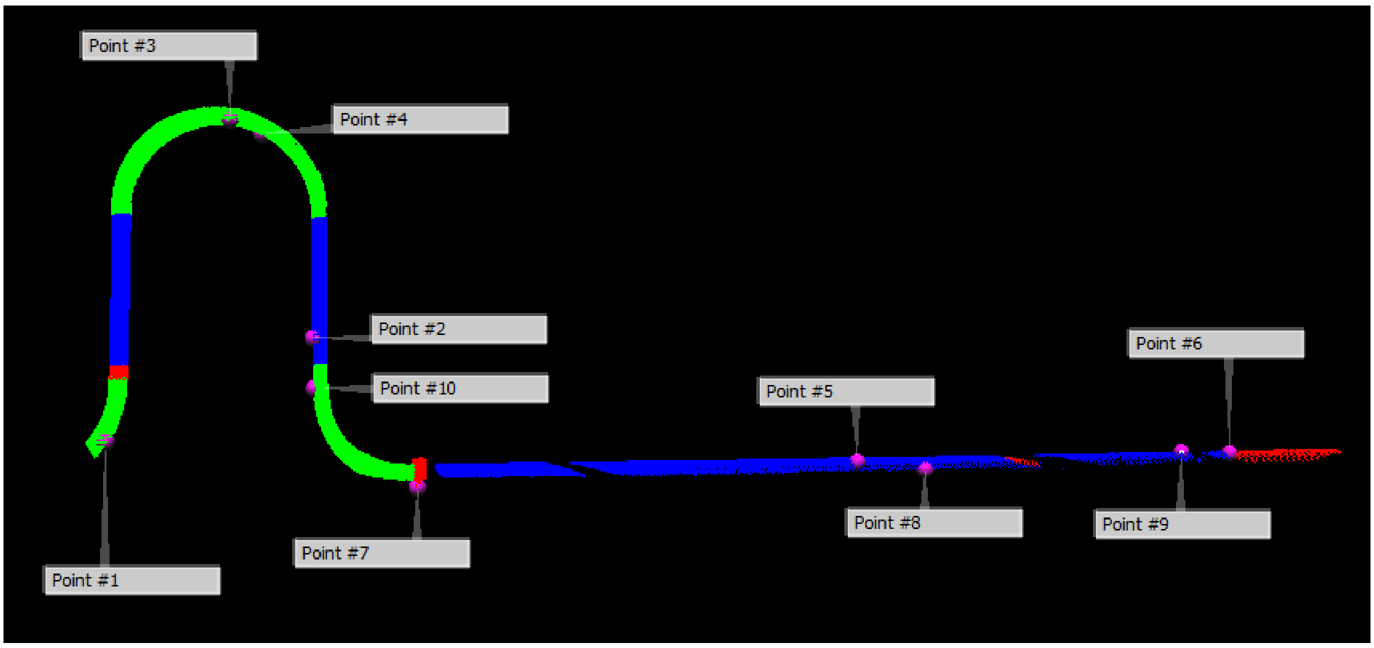


Figure 26: Seeds for the bend pipeline data set.

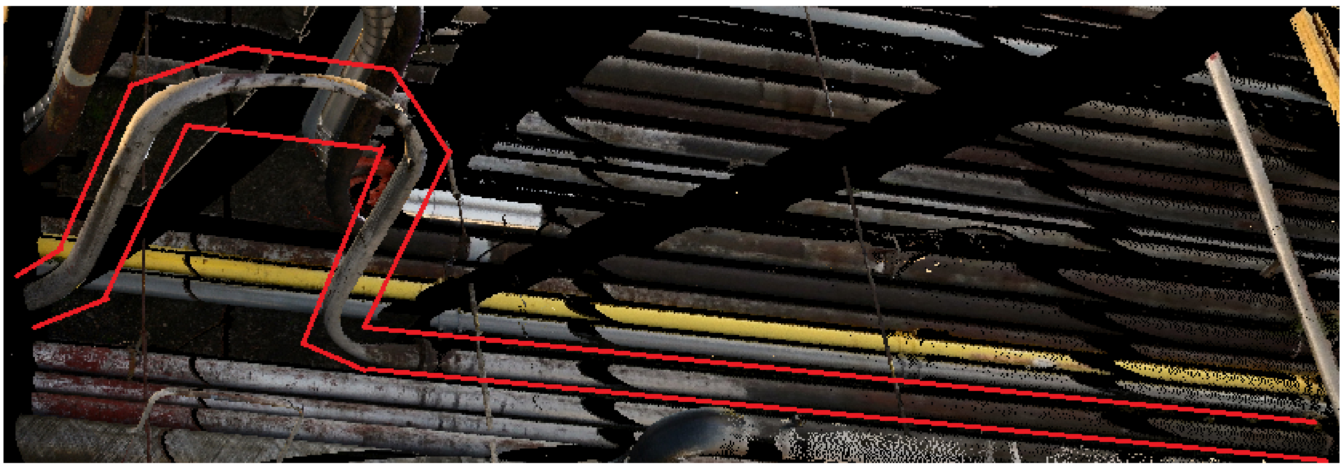


Figure 27: The bend pipeline in its full data set.

2.5 meters range do well for a number of seeds.

The largest similarity between these two pipelines is that they are both flanked by similar pipelines on both sides. In this situation the algorithm requires a search area that is large enough to fit the diameter of the large pipeline with some tolerance but exclude as much of the flanking pipelines as possible.

Seed 7

The poor performance of seed 7, the blue line at the top of Figure 28, can be explained by its position on a valve, see Figure 26. Since the diameter of the valve is larger than the surrounding pipes the seed will never be found on a correct RANSAC result which means the algorithm stops immediately.

It does detect a small percentage of the pipeline with a side length of 1 m since the support threshold for such a small area is very low. This allows the valve to be detected as a small cylinder which connects to the pipeline. The small area will struggle immediately to the left and right since it cuts off the bend on the left and the low point density on the right leads to such low support thresholds that the RANSAC results are simply nonsensical.

These nonsensical results can be seen up to a side length of 2 m in Figure 29. After that detection simply fails. Note that the large peaks in the figure are again from very small models.

Seed 1

Seed 1 is the second seed to get total detection failure. Even though its location in Figure 26 seems good, it has high density and low curvature/complexity. The issue with this seed is that the bend it lies on cannot grow with the search area. The pipeline that extends upwards does not continue the bend and will be recognized as a different primitive. Downwards from seed 1 the data simply stops. When a primitive cannot grow with the search area it will fail to reach its support threshold more often as the side length increases.

Seed 6

Seed 6, rightmost spike in Figure 29, demonstrates a tendency to generate exclusively false positives when the search area grows. This is due to the very low point density in the data. The low point density means that the support threshold of 3% is a small number of points. The low density also means the normal estimation is of lower quality which opens op the possibility for large numbers of false positives.

Normally these are filtered out by the existing model. When the seed lies in the low density area, like seed 6, the model may be initialized by a false positive which is then extended by even more false positives.

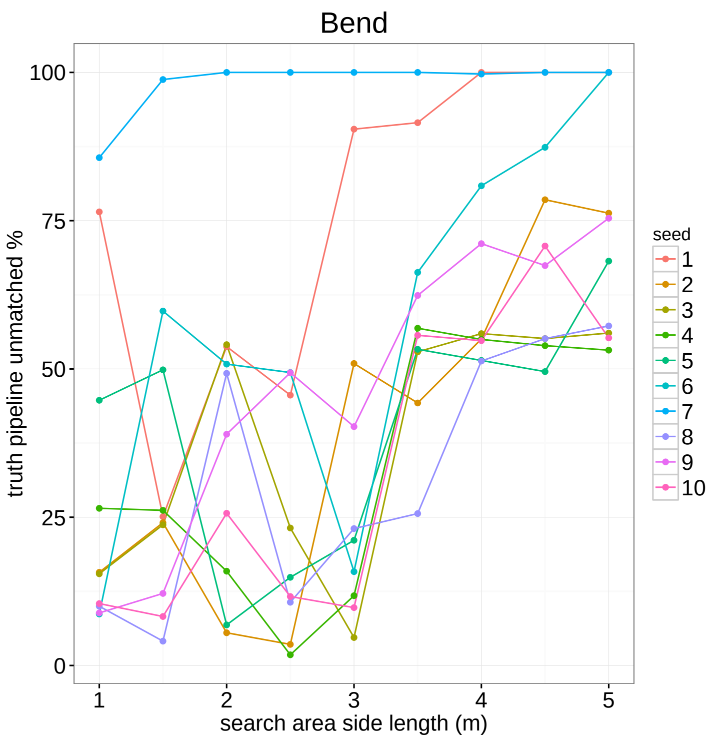


Figure 28: The false negatives for the primitive pieces.

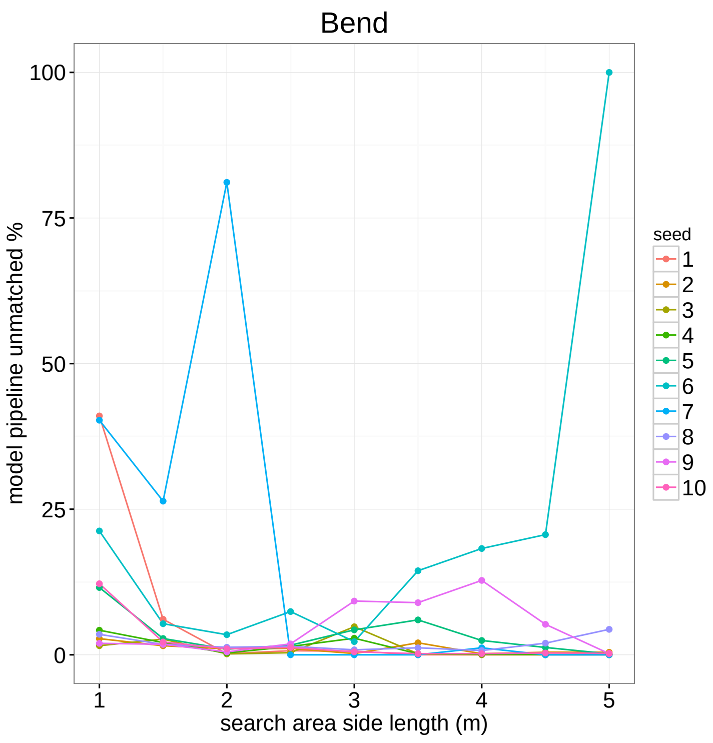


Figure 29: The false positives for the primitive pieces.

4.4.2 Region pieces

Like the other data sets where regions are not a significant part, Figure 30 appears rather chaotic. In this case there is a clear division between the seeds. In the first group are seeds 1,2,3,4, and 10 that lie in the high density area on the left of Figure 26. The other seeds lie on the right in the lower density area. The small region above seed 1 is responsible for just over 60% of the total region points in the pipeline and the region at seed 7 is responsible for approximately 35%. Since the low density seeds are simply not reaching the first region and only occasionally the second their average false negative rates are quite high.

The large variation seen in some of the low density seeds is due to them alternating between reaching and not reaching the two regions depending on the search area boundaries. Since the region's actual pipeline length is very small a detection run that is just centimeters shorter than another might not get the appropriate seeds to grow that region. This explains why we see bigger fluctuations in Figure 30 than in Figure 28.

As in the other data sets the peaks seen in Figure 31 are from very small models.

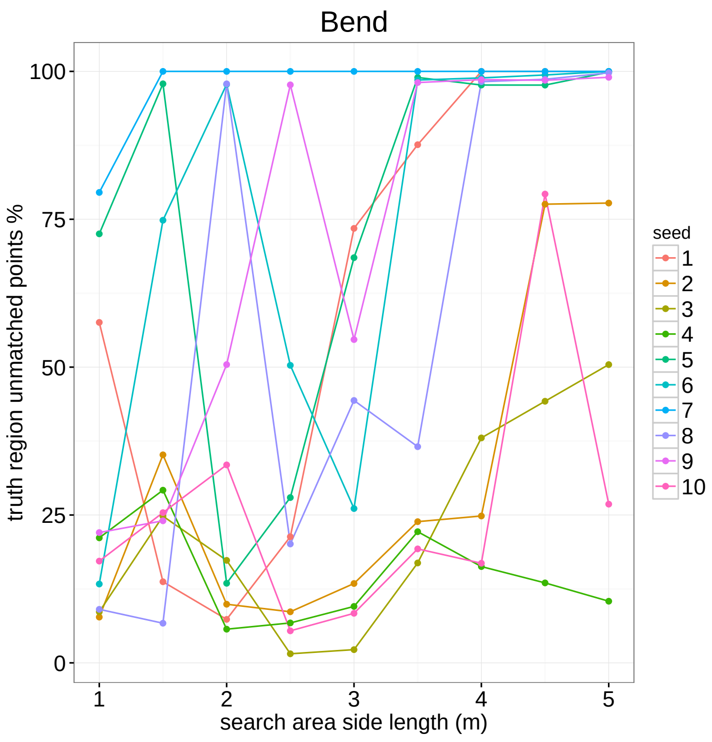


Figure 30: The false negatives for the region pieces.

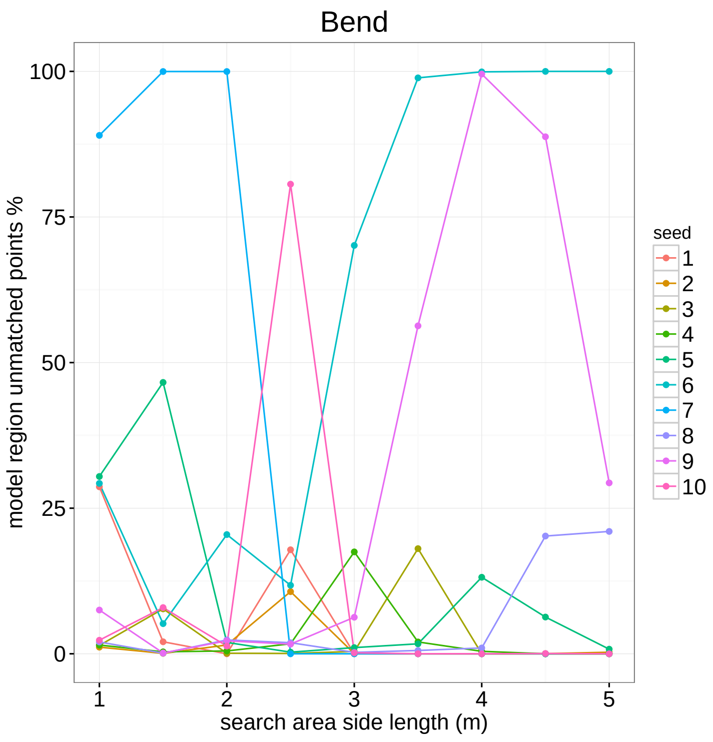


Figure 31: The false positives for the region pieces.

5 Qualitative analysis

In this section we will examine two situations where the algorithm struggles to create good models and suggest possible solutions to the encountered problems. The figures in this section show unclassified points in blue, cylinders as green, bends as yellow, and regions as red.

5.1 Gradually deteriorating data

A common situation in the scans of the Pernis facility is a very long, straight pipeline that runs away from the scanner with minimal occlusion. This situation is not challenging due to complex geometry or clutter by unrelated points but due to the point density dropping gradually with the distance from the scanner.

All the pipelines in the previous section had a natural cutoff point where the data simply stopped before point density got very low. The situation in Figure 32 is an example of a situation without such a natural cutoff. The point density drops from right to left.

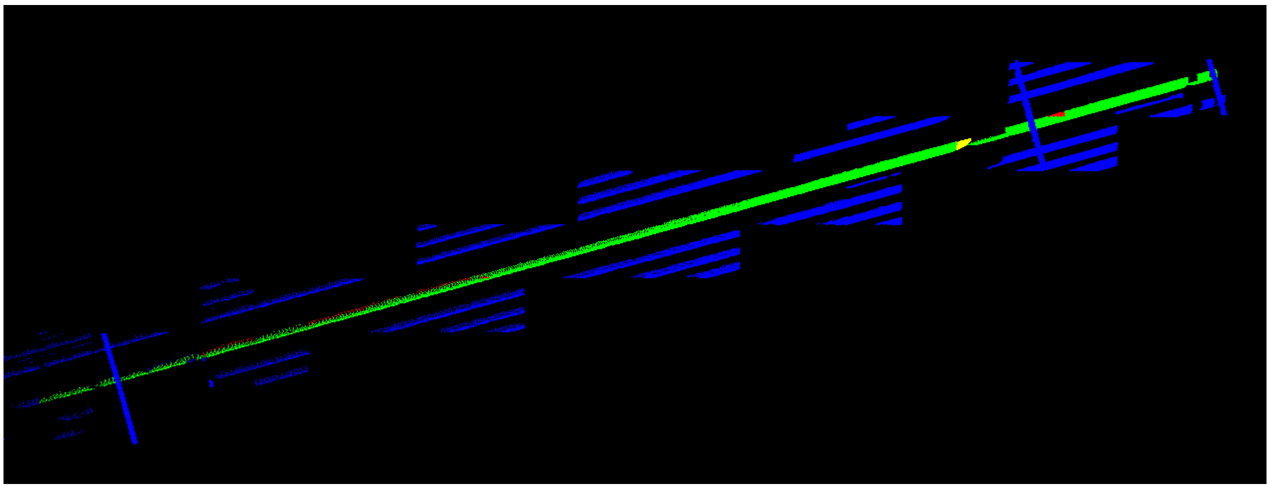


Figure 32: Successful detection of a long pipeline

The constructed model for this pipeline is shown in Figure 33. The right of the model is consistently accurate, as the algorithm moves left in the figure RANSAC becomes increasingly inaccurate. The lower the point density the more RANSAC overestimates the radius of the pipeline. Figure 34 clearly shows the diameter differences.

These diameter inaccuracies are not necessarily an issue since they do not interfere with tracking the pipeline. However, they do make the algorithm more prone to connecting false positives. All the connection checks in Section 2.1.1

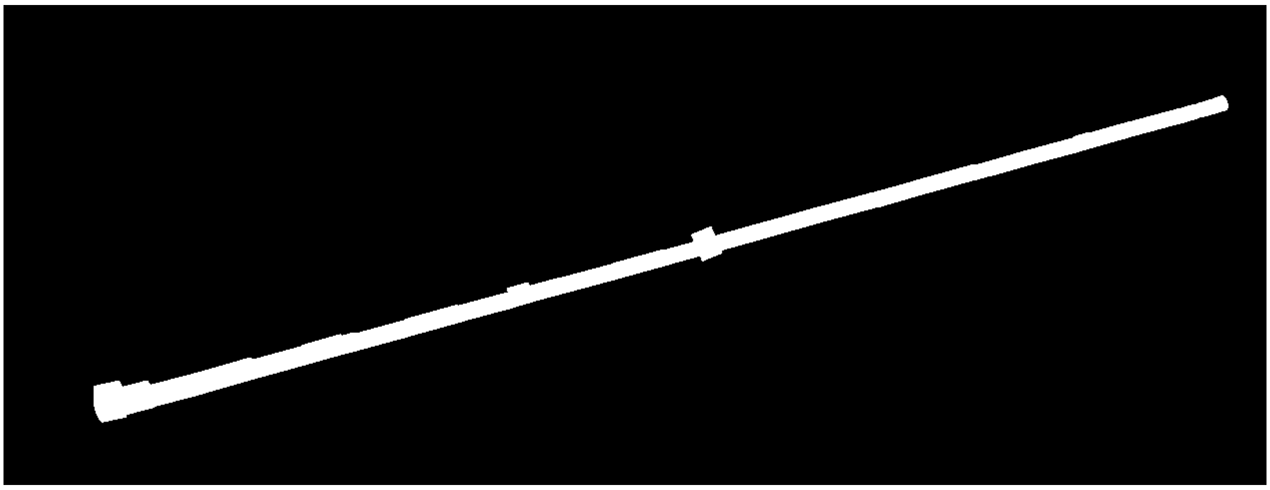


Figure 33: The model generated for the detection result in Figure 32

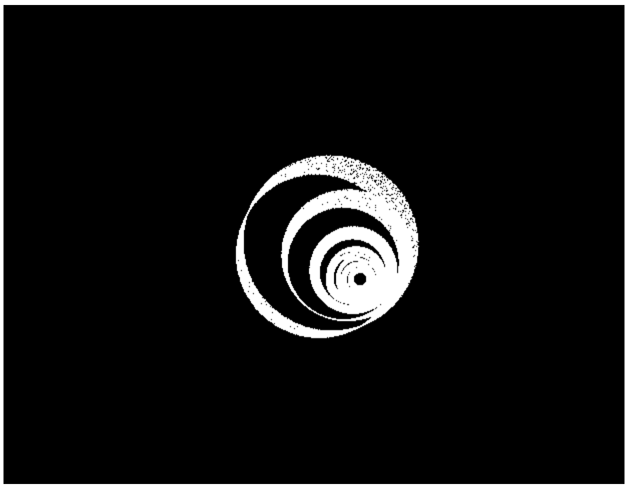


Figure 34: The front view of Figure 33

rely on the estimated diameter of the primitives. When the overestimation gets too large, false positives will be connected. Additionally these false positives can bridge the gap between the model and previously detected pieces which can extend the pipeline with several meters of false positives with only a single detection error. The error also builds over multiple connections. Since every connection has a 50% diameter threshold the allowed diameter can grow significantly after a few accepted overestimations.

The result of such an error is shown in Figures 35 and 36. A large diameter inaccuracy at the very left of the model led to a connection to a false positive. This false positive happened to be positioned just so it could connect to the neighboring upper pipeline. The bottom pipeline was connected through a vast overestimation of its diameter which led it to completely contain the main pipeline.

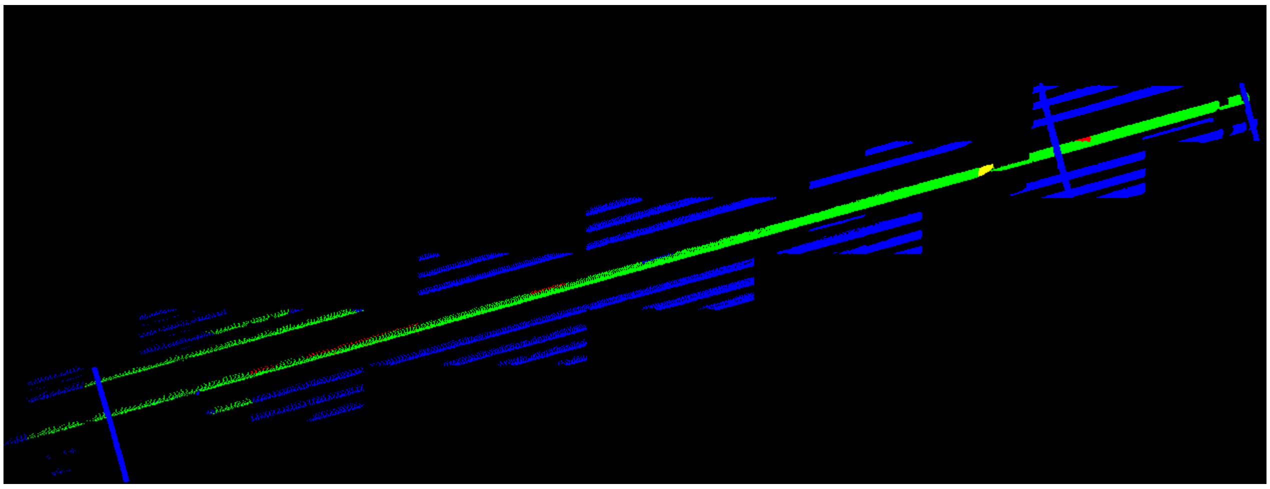


Figure 35: Detection run that includes false positives

Possible solutions

The most straightforward solution to this problem would be to stop the algorithm at a predefined point density threshold. This solution would require fine-tuning for every scan and possibly even for every pipeline in a scan to avoid stopping too early.

Another option would be to discard false positives based on their diameter to length ratio. All of the obvious nonsensical parts of the model in Figure 36 have a larger diameter than length. This property could be exploited to filter them out. While this would require tuning the ratio at which a primitive is discarded it is likely the chosen ratio would function well across all scans since the shape of the false positives is inherent to the RANSAC method and not to the data. Since the

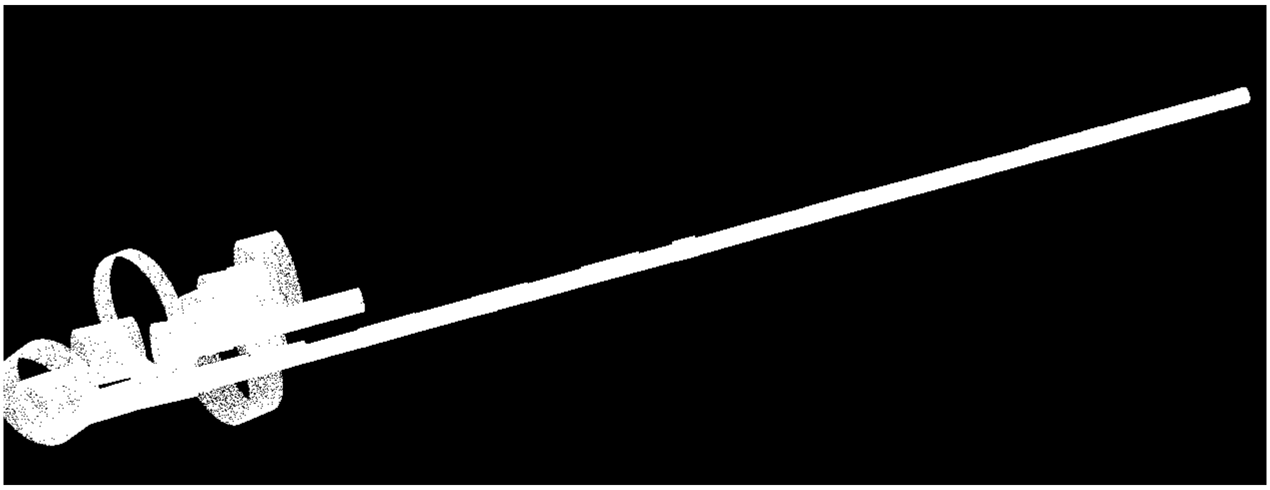


Figure 36: The model generated for the detection result in Figure 35

region growing algorithm could step in to recover from primitives that were filtered out by mistake it should not be an issue if the filtering is a little too aggressive.

A more sophisticated approach would be to not just take the current primitive into account for the diameter threshold but also previously connected pieces. Since most of the pieces have an accurate diameter estimate this should reduce the likelihood that a false positive is connected. The problem with this approach is that there are areas of the pipeline that are supposed to differ in diameter. For example, a "T" split usually connects a narrow and a wide pipe as we saw in the join pipeline in Section 4.3. To make sure the algorithm only takes the relevant model pieces into account for the diameter threshold, the algorithm would have to reliably identify "T" splits.

5.2 Non-pipeline regions

For a non-pipeline region to grow there need to be seeds at the end of a cylindrical or bend piece that lie very close to a non-pipeline structure and the transition from pipe to non-pipe needs to be sufficiently smooth. This can happen in two ways: either the pipe meets a support structure that occludes so much of it that the detection cuts the pipe off where they meet or an unfortunate search area boundary cuts off a pipe right as it meets a support structure.

In both cases the point density needs to be either very high or low for the transition to be considered smooth. For a very high point density the 35 neighborhood of the normal estimation slowly transitions from pipe to non-pipe when moving from point to point. Since the normal is determined by the entire neighborhood it will also slowly transition from pipe aligned to region aligned. For a very low point density the normals are simply inaccurate so it is possible for a few inaccurate normals to line up well enough to allow the region to grow.

High density

Figure 37 shows an example of the first case where a support structure occludes all the pipes that are behind it. The point density near the wall on the left of the figure is very high which allows the normals to transition smoothly even though the actual geometry features a sharp edge. In this case detection started at the indicated seed, reached the wall and proceeded to grow into all cylinders that also touch the wall.

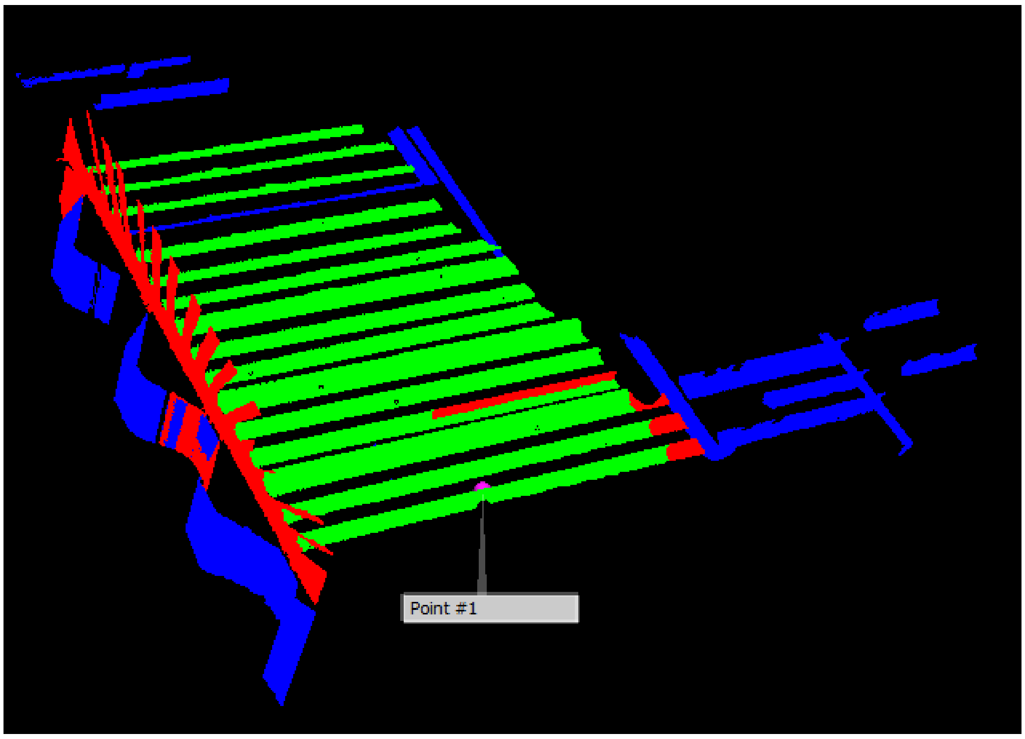


Figure 37: A detection run taking the wall in which the pipes are embedded into the model.

Low density

The join pipeline which we have analyzed in Section 4.3 can include a non-pipeline region when a search area boundary occurs at a very specific spot. Figure 38 shows the situation. The search area boundary occurs at the transition from blue to red points.

The jump to the non-pipeline region occurs at the white dot, which coincidentally corresponds to seed 6 in Figure 20. The points above and the single point to the left of the white dot lie isolated from both the pipeline and the region. This makes the normals inaccurate and just happens to bring them close enough to be in the 10 neighborhood of each other.

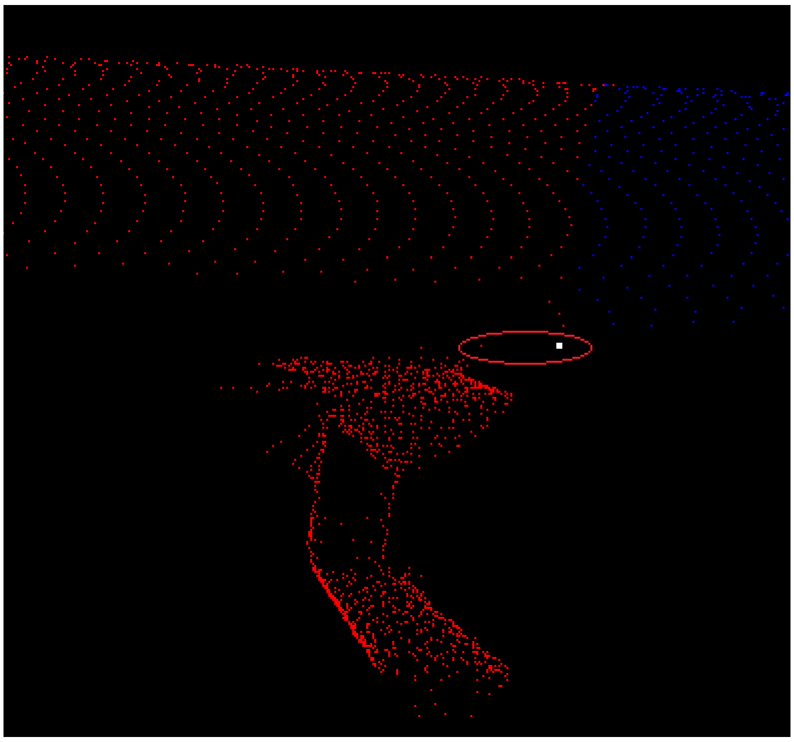


Figure 38: A side view of a possible non-pipeline region on the join pipeline. The white dot corresponds to seed 6 in Figure 20. The red oval marks the two points that cause the jump to the non-pipeline region

Possible solutions

The high density and low density errors are symptoms of different issues. In the high density situation the normal estimation neighborhood is too large which leads to smoothly varying normals even at high curvature. Decreasing this size however will lead to more problems in low density areas where the normals need a large neighborhood to be approximated well.

The low density errors are due to bad luck, they only happen if the normal calculation is inaccurate in just the right way for several points. We could make the region growing method more restrictive in its normal, residual and distance thresholds. The drawback to that is that this will lead to more detection failures in medium density and high curvature situations where we need region growing to recover from RANSAC difficulties.

Developing a normal estimation algorithm that can provide a certainty measure of its estimation invariant to neighborhood size or point distribution would be a major advance in the field. All current methods either fit a plane to the neighborhood or construct a mesh for it and can only provide either the residual of the fit or some mesh characteristics. Neither of which can make any direct statements on the accuracy of the estimation.

In both situations the regions might be discarded if we applied a curvature analysis akin to [17]. The authors use the two principal curvatures of point sets to determine if they qualify as pipelines. This may help since most non-pipeline regions are distinctly non-cylindrical. It would introduce more parameters to our algorithm that need tuning and introduce significant additional complexity. The current region growing algorithm only involves distance, normal and residual checks. Curvature analysis would introduce quadratic surface fitting which undermines the simplicity of the current algorithm.

6 Conclusion and future work

6.1 Conclusion

We have presented an algorithm that can both track and model a pipeline across multiple data sets that does not need any preprocessing or assumptions about pipeline characteristics. Section 4 showed that the algorithm is too sensitive to the chosen seed and side length to work completely unsupervised.

Parameter guidelines

The seeds that performed well in Section 4 all shared three characteristics: high point density, low curvature and if possible little clutter in the surrounding area. In essence the goal is to give RANSAC the best chance at obtaining an accurate result in the first area. The side length should be chosen to fit the diameter of the tracked pipeline with enough margin to also contain significant parts of bends.

In Section 4 we have seen that in most situation seeds in the range of 1.5 to 2.5 meters work well. This is roughly 4 to 6 times the diameter of the pipelines in the scans. Since the seeds all lie on the surface of the pipelines the smallest margin between primitive and area boundary ranges from 1 to 2 pipeline diameters.

Interactive use

The iterative and local nature of our algorithms makes it suitable as an interactive tool. On small areas of interest each iteration takes under 10 seconds, which can be reduced drastically if the normals are precomputed. If an iteration introduces false positives the algorithm could be paused in between iterations to allow a user to fix the mistake. Since the model allows the user to work on the level of primitives and regions instead of points this work flow would still be a significant improvement over the manual extraction which works at a point level.

6.2 Future work

6.2.1 Model analysis

We have suggested some improvements to the algorithm in Section 5. The most promising of these is that we could use a more detailed analysis of the model to filter out bad detection results.

No other algorithm in the field currently uses an iterative approach where constructing a model aids in interpreting subsequent detection results. In other disciplines where data is generally available as a time series the detect-model-predict mindset is more common.

For example, Antich et al. [2] present a Kalman filter-based method for visual tracking of underwater pipelines. The Kalman filter is not directly applicable to our algorithm since above ground pipelines can change abruptly in both diameter and direction at "T" splits. However, making use of the available model to better interpret detection results presents in interesting research direction.

6.2.2 Search areas

The current search area implementation limits the search areas to axis aligned bounding cubes. While this makes intersection tests and ensuring no areas overlap easier it is by no means ideal. The ability to adjust search area size, shape and orientation will reduce the number of unfortunately placed boundaries.

6.2.3 Jumping between scans

Since our algorithm works on one search area at a time it is easily extended to switching scans in between iterations. Besides the obvious benefit that this allows the algorithm to track very long pipelines it could also aid the algorithm when dealing with the gradually deteriorating data seen in Section 5.1. When the point density gets too low the algorithm could switch to a scan where the scanner was positioned closer to the currently tracked pipeline section. The models from the two scans will overlap where the algorithm switched scans. Since the model pieces created in the low density scan are less accurate they could be replaced by the pieces created in the higher density scan. Filtering out false positives could be done by checking which low density pieces have no high density matches. Since we do not use points from different scans together in the same RANSAC or region growing runs and our connection methods tolerate a few centimeters misalignment we do not have to consider registration of the scans.

7 References

[1] Rolf Adams and Leanne Bischof. Seeded region growing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):641–647, 1994.

[2] Javier Antich and Alberto Ortiz. Underwater cable tracking by visual feedback. In *Pattern Recognition and Image Analysis*, pages 53–61. Springer, 2003.

[3] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):1–13, 2011.

[4] Thomas Chaperon, François Goulette, and C Laurgeau. Extracting cylinders in full 3d data using a random sampling method and the gaussian image. In *VMV*, volume 1, pages 35–42, 2001.

[5] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Pattern Recognition*, pages 236–243. Springer, 2003.

[6] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[7] V Hough and C Paul. Method and means for recognizing complex patterns, December 18 1962. US Patent 3,069,654.

[8] K Kawashima, S Kanai, and H Date. Automatic recognition of a piping system from large-scale terrestrial laser scan data. *Proc. Int. Society for Photogrammetry and Remote Sensing Laser Scanning 2011*, 2011.

[9] Joohyuk Lee, Hyojoo Son, Changmin Kim, and Changwan Kim. Skeleton-based 3d reconstruction of as-built pipelines from laser-scan data. *Automation in Construction*, 35:199–207, 2013.

[10] Guo Li, Ligang Liu, Hanlin Zheng, and Niloy J Mitra. Analysis, reconstruction and manipulation using arterial snakes. *ACM Transactions on Graphics-TOG*, 29(6):152, 2010.

[11] Yong-Jin Liu, Jun-Bin Zhang, Ji-Chun Hou, Ji-Cheng Ren, and Wei-Qing Tang. Cylinder detection in large-scale point cloud of pipeline plant. *Visualization and Computer Graphics, IEEE Transactions on*, 19(10):1700–1707, 2013.

[12] Rongqi Qiu, Qian-Yi Zhou, and Ulrich Neumann. Pipe-run extraction and reconstruction from point clouds. In *Computer Vision–ECCV 2014*, pages 17–30. Springer, 2014.

[13] Tahir Rabbani and Frank van den Heuvel. Efficient Hough transform for automatic detection of cylinders in point clouds. *ISPRS WG III/3, III/4*, 3:60–65, 2005.

[14] Tahir Rabbani, Frank van den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):248–253, 2006.

[15] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[16] Ruwen Schnabel, Raoul Wessel, Roland Wahl, and Reinhard Klein. Shape recognition in 3d point-clouds. In *The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, volume 8. Citeseer, 2008.

[17] Yun-Ting Su, James Bethel, and Shuowen Hu. Octree-based segmentation for terrestrial lidar point cloud data in industrial applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 113:59–74, 2016.

[18] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

Appendices

A Common terms

To ensure there can be no confusion when we use terms common throughout the literature we will define them in this section.

Definition A.1 *A primitive is a simple geometric shape whose surface can be described by a limited number of parameters.*

A sphere, for example, is a primitive whose surface can be described by a center position and a diameter. More complex shapes such as trees generally can't be described by parameters. Representing them requires a format that explicitly describes its surface such as a triangle mesh.

Definition A.2 *A point set is subject to sensor noise when the measured points show small random deviations from what their position on the measured object should be. See Figure 39.*

No measuring device is perfect so all non-synthetic point sets will be subject to sensor noise to some degree.

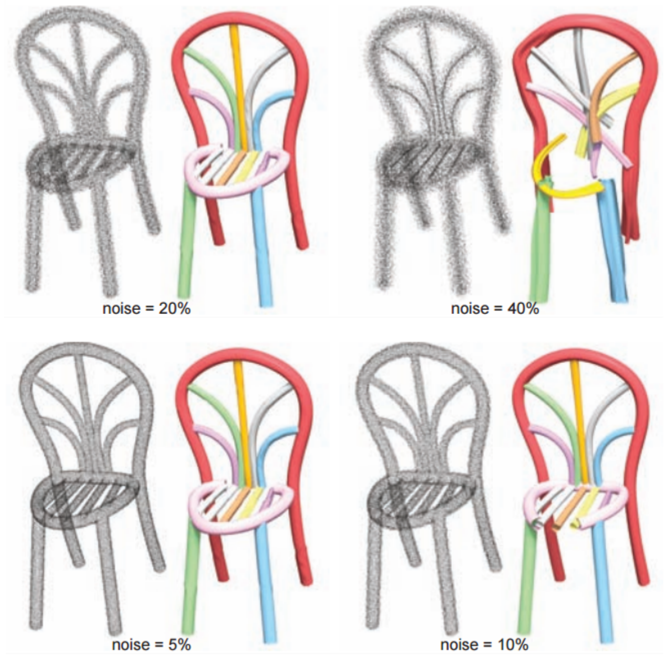


Figure 39: Synthetic sensor noise on a chair and the resulting reconstruction. Taken from [10].

Definition A.3 *A point set is subject to spatial noise when points inconsistent with any surface are distributed throughout the entire point set. See Figure 40.*

Spatial noise can be completely random like Figure 40 or show a distinct pattern due to a quirk in the scanner. For example, the scanner used to produce the point cloud in Figure 41 tends to produce noise in a half sphere above ground level at the maximum sensor range.

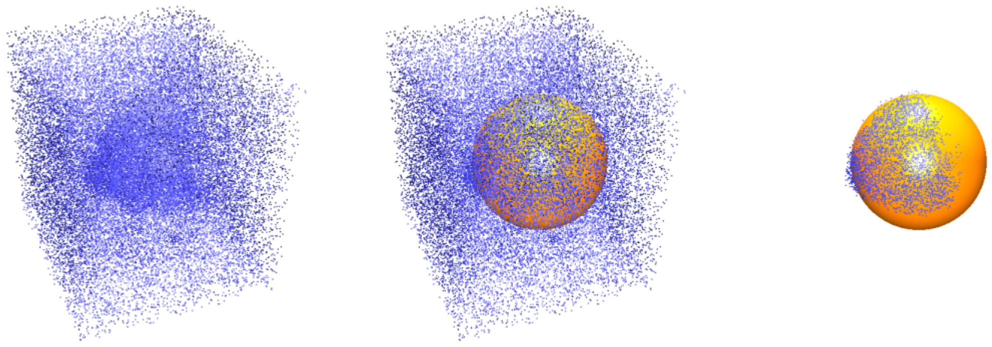


Figure 40: Synthetic spatial noise around a sphere. Taken from [15].

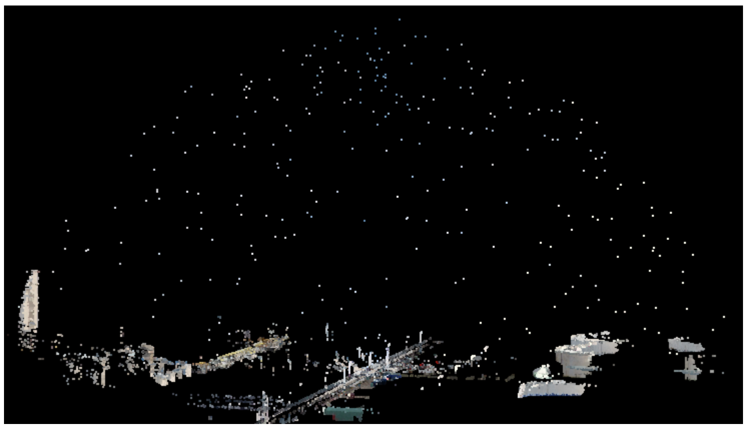


Figure 41: Spatial noise generated in a half sphere in the sky by a scanner that does not handle a lack of reflection well.

Definition A.4 *An artifact consists of one or more points that do not appear to belong to any measured surface.*

Artifacts are usually caused by sensor errors and are generally quite rare. In some cases artifacts can be larger structures. For example, when something moves in front of the sensor during scanning. See Figure 42. The key difference between artifacts and spatial noise is that artifacts are local structures while spatial noise is a global effect.

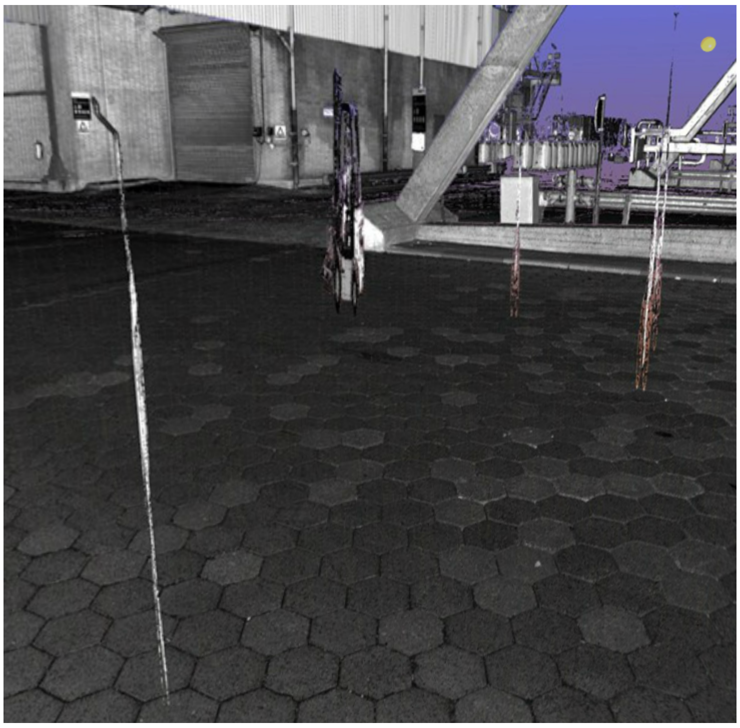


Figure 42: Artifacts caused by pedestrians and a forklift moving in front of a laser scanner.

Definition A.5 *An outlier is a single point that is inconsistent with any measured surface, or with the current hypothesized primitive(s) if the ground truth is unknown.*

The term outlier is used when the origin of the point (noise, artifacts, or erroneous hypotheses) is not important.

Definition A.6 *A point supports a primitive when it lies on its surface, such points are also called inliers.*

Most algorithms define a maximum distance to the surface for points to be considered as support since sensor noise is inescapable.

B Data set description

To make informed decisions on the values for the parameters of the component algorithms (Section C) we need to know the characteristics of the point clouds these algorithms have to work on. This section will give a detailed description of the data sets used in this thesis.

The characteristics that describe a point cloud are

- sensor noise (as std. deviation in mm)
- spatial noise (as percentage of total points)
- point density (as points per m^2 of surface)

Sensor noise

Sensor noise produced by a laser scanner comes from two sources [1] range noise and angular accuracy. Range noise originates from the laser sensor itself and depends on the reflective properties of the surface and the distance to the surface. For a plot of the range noise over increasing distance see Figure 43. These noise measurements give the standard deviation of a Gaussian distribution with mean 0.

Angular accuracy is the difference between the actual mechanical orientation of the scanner and the orientation as perceived by the scanner software. For this scanner that is 0.007° (std. deviation), which gives an error of 2.2 cm at 187 m (the maximum scanner range). This error scales linearly with distance from the scanner.

The sensor noise values we use in Section C are the sum of the range noise and angular error values. We use the grey surface, green line in Figure 43, for the range noise in these calculations.

Spatial noise

The scanner used in this project does not appear to produce significant amounts of spatial noise. The only instances of this kind of noise are a small number of points at maximum scanner range in the sky. Artifacts are rare as well. Figure 41 gives an example of a common kind of spatial noise. Figure 42 gives an example of a common kind of artifact.

[1]All data in this section comes from the fact sheet of the scanner at `http://www.zf-laser.com/fileadmin/editor/Datenblaetter/Z_F_IMAGER_5010C_Datasheet_E.pdf`

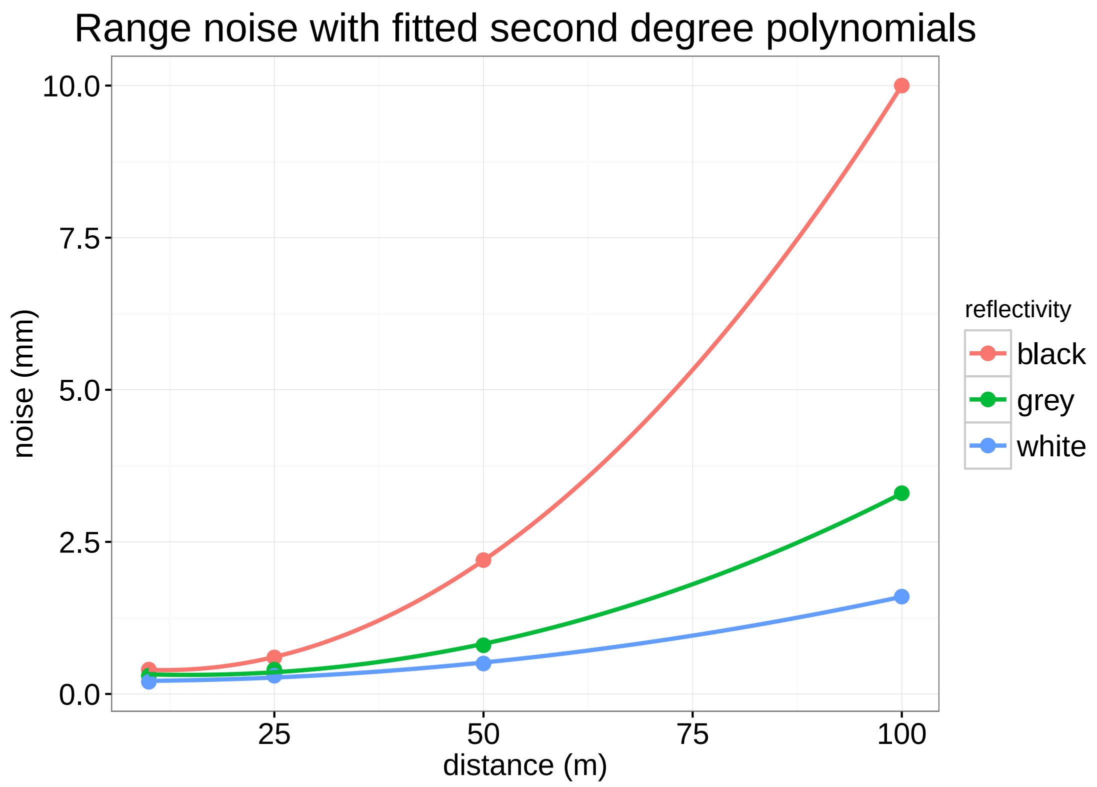


Figure 43: Range noise of the scanner, second degree polynomials have been fitted to the available data.

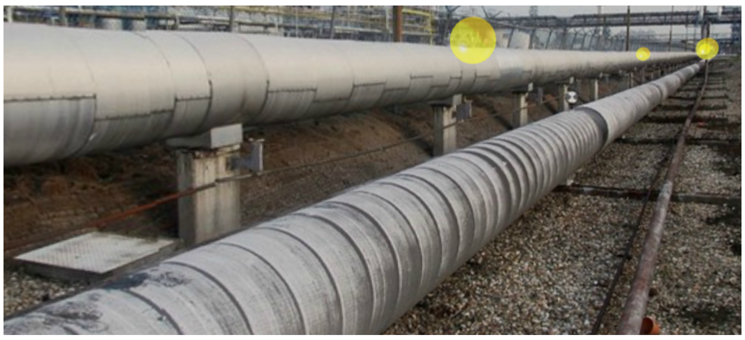


Figure 44: Pipelines stretching across multiple scans. This view is taken from one scanner, the yellow spheres are the positions of nearby scanners.

Point density
The scanner produces between 25 and 81.5 million points per scan, depending on the settings of the scanner. Point density on a surface depends on the distance to the scanner, the angle the surface makes with the incident laser beams and the total number of points the scanner produces.

Volume based density estimates (in points per m^3) can produce inaccurate estimates. If a volume of space is very close to the scanner but contains only a very small piece of pipeline the point density in the entire volume is low, while the point density on that pipeline is very high.

Instead we use a surface based estimate (in points per m^2). For the synthetic data in Section C we approximate the point density by assuming all points are projected onto a sphere surrounding the scanner. When we want to approximate the density at, for example, 15 m we calculate the surface area of a sphere with a 15 m radius divided by the number of points. To compensate for the fact that the scanner can not scan below itself we take 90% of the calculated surface area into account.

B.1 The Pernis repository

As mentioned in the introduction the Pernis repository contains 19,638 scans in total. These scans are stored in a way that preserves their relative positions. The exact coordinates of each scanner are stored which allows an algorithm to use points of multiple scans to track a pipeline. For an example of how a user interface presents the relative position of scans see Figure 44.

The coordinate systems of the different scanners should be reasonably well aligned. All the scans were done with the same GPS hardware and since the scanner stands still for upwards of 1 hour the GPS position can be averaged over a large number of measurements. The misalignment between two scans should be in the order of a few centimeters.

C Component algorithm experiments

C.1 Synthetic data generation

The synthetic point clouds are generated by placing a number of points on the surface of three parallel cylinders 15 meters in length. These touch each other at the sides. Noise is added by displacing the points in a random direction by a distance sampled from a normal distribution with mean 0 and the chosen noise value as the standard deviation.

The characteristics in the point cloud generation are:

- sensor noise (as std. deviation in mm)
- point density (as points per m^2 of surface)

These two characteristics will be fixed in three configurations representing conditions 15, 50 and 100 meters from the scanner. These configurations will be referred to as near, middle, and far. We do not include spatial noise since it does not tend to be a significant factor in the kind of data we use.

The values, see Section B for data and calculations, for the characteristics are (near, middle, far):

- sensor noise: 2.1, 6.92, 15.5
- point density: 14157, 1273, 318

C.2 Algorithm parameters

C.2.1 RANSAC

For RANSAC the parameters that need to be investigated are:

- minimum support (s): [0.02...0.2] step size: 0.02
- normal threshold (α): constant, see Section 2.2.1
- size of bitmap cells (β): constant, see below
- distance threshold for support (ϵ): [2 mm...20 mm] step size: 2 mm

Minimum support (s)

This parameter determines how much support a primitive must have to be considered as detected. A low value encourages under-segmentation, a high value encourages over-segmentation. Measured as ratio of total point set.

Normal threshold (α)

This parameter determines how much the normal of a point may deviate from the normal of the hypothesized primitive at this point. A high value encourages under-segmentation, a low value encourages over-segmentation. Additionally a low value may cause inaccurate hypotheses to gain too much support leading to false positives. Measured in degrees.

Size of bitmap cells (β)
The algorithm defines a bitmap across the surface of every primitive. Primitives are bounded by calculating the largest connected component of neighboring pixels that have a point in them. This parameter determines the size of the pixels in the bitmaps. A low value causes over-segmentation by splitting a single detected primitive in multiple parts. A high value causes under-segmentation by failing to split apart primitives that are grouped together by the detection step. We will set this value so points that are at an average distance from each other will not have an empty pixel in between them. To do this we set the side length of a cell to the upper end of the 95% confidence interval of the interpoint distances of the data set.

Distance threshold for support (ϵ)
This parameter determines how far a point may lie from a primitive to still be considered as support. A low value may lead to detection failure as no hypotheses receive sufficient support, a high value could lead to false positives being added to the output.

Optimum probability threshold
The algorithm will continue to draw samples until it is very unlikely the best sample has not yet been drawn. Schnabel et al. [15] put this at 0.01. We found that computation times were not significantly affected by lowering the threshold to 0.001.

C.2.2 Region growing

For region growing the parameters that need to be investigated are:

- minimum support (s): [0.02...0.2] step size: 0.02
- normal threshold (α): constant, see Section 2.2.1
- residual threshold (ϕ): [0.8...0.98] step size: 0.02
- nearest neighbor threshold (β): [10...55] step size: 5

Minimum support (s)
This parameter determines how many points a region must contain to be added to the output. A low number may lead to the output being swamped in insignificant results, a high number may lead to the exclusion of important regions from the output.

Normal threshold (α)
This parameter determines how much the normal of a point may deviate from the normal of the seed point that is considering adding it to the region. A high value may cause under-segmentation as regions grow past sharp bends, a low value may lead to over-segmentation as regions are cut off at smooth bends. This parameter is not used to control behavior in noisy areas since inaccurate normals could still fall within the threshold. Measured in degrees.

Nearest neighbor threshold (β)
This parameter determines how easily a region will grow across gaps in the data by defining how many neighborhood points are considered for addition to the region. A low value may cause over-segmentation as a region may be stopped by a few missing or misplaced (by noise) points. A high value may cause under-segmentation as regions are prepared to grow across large volumes without points.

Residual threshold (ϕ)
This parameter determines how inaccurate the normal of a point may be to still be able to become a seed point after being added to a region. A low value may lead to over-segmentation as regions are cut off at areas with slight curvature, noise or drop in point density, a high value could cause under-segmentation as regions grow right through noise, high curvature and missing data. This parameter does not control behavior at bends since the residuals stay constant on smooth bends and recover quickly (within the β neighborhood) on sharp bends. The parameter is measured in percentiles: $\phi = 0.98$ means that a residual must be lower than the top 2% of all residuals to pass.

C.3 Experimental procedure

We run every algorithm 100 times on every data set for every parameter configuration. For every configuration we report the average of the following performance measures:

- degree of segmentation
- parameter accuracy (RANSAC only)
  1. axis angular difference
  2. axis distance
  3. width difference
- region completeness (region growing only)

The degree of segmentation is 1 when the detection algorithms segment the data in exactly as many regions as are in the data. It is less than 1 when the detection algorithm under segments the data, it is more than 1 when the data is over-segmented.

Region completeness is 1 when a region contains as many points as we expect, we know how many points are generated on each cylinder. There may be fewer points in the region if it was preceded by a number of small regions that did not reach sufficient support to be considered for output. In contrast to RANSAC the region growing algorithm removes points from the data even if their region does not reach sufficient support.

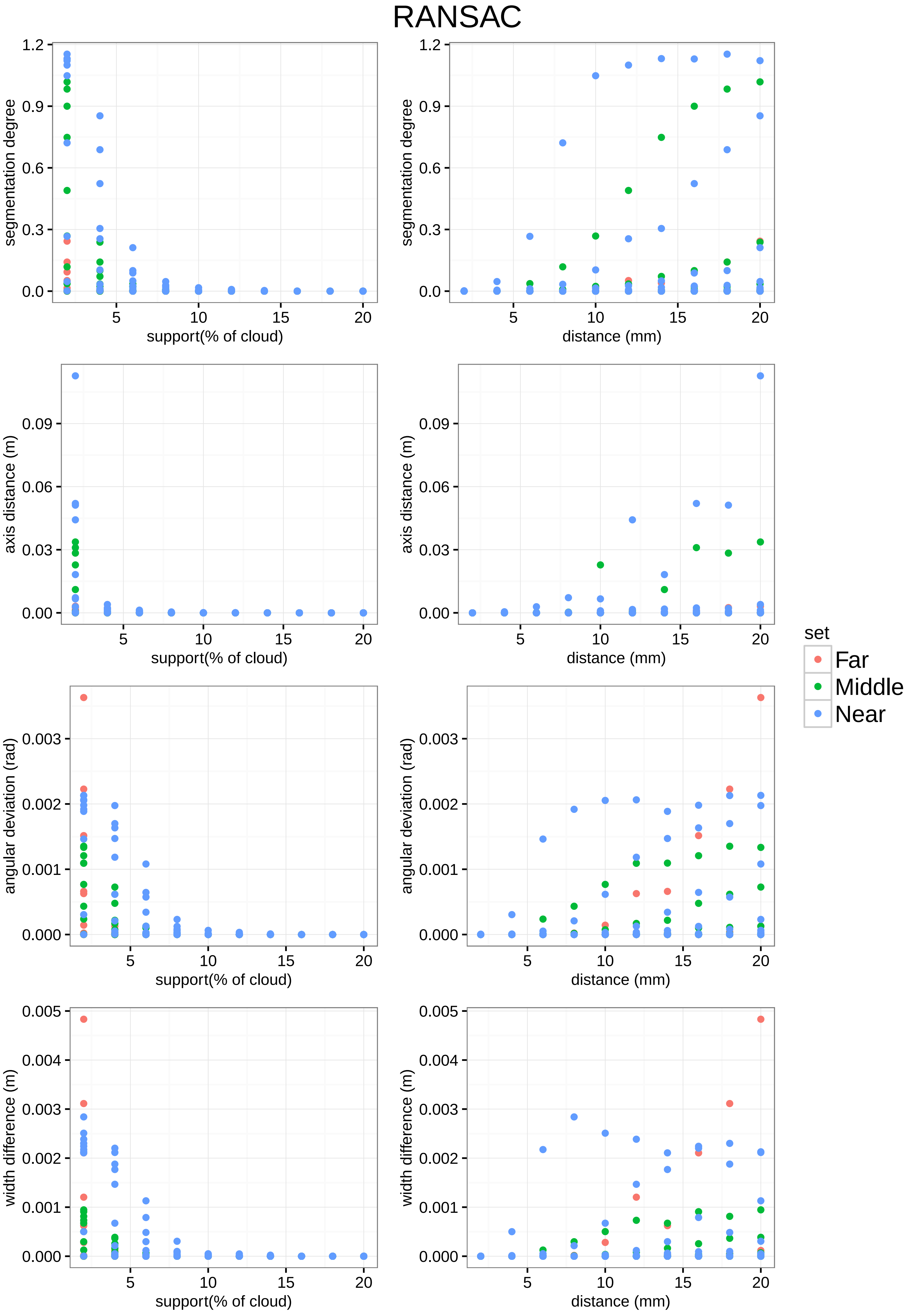
C.4 Results

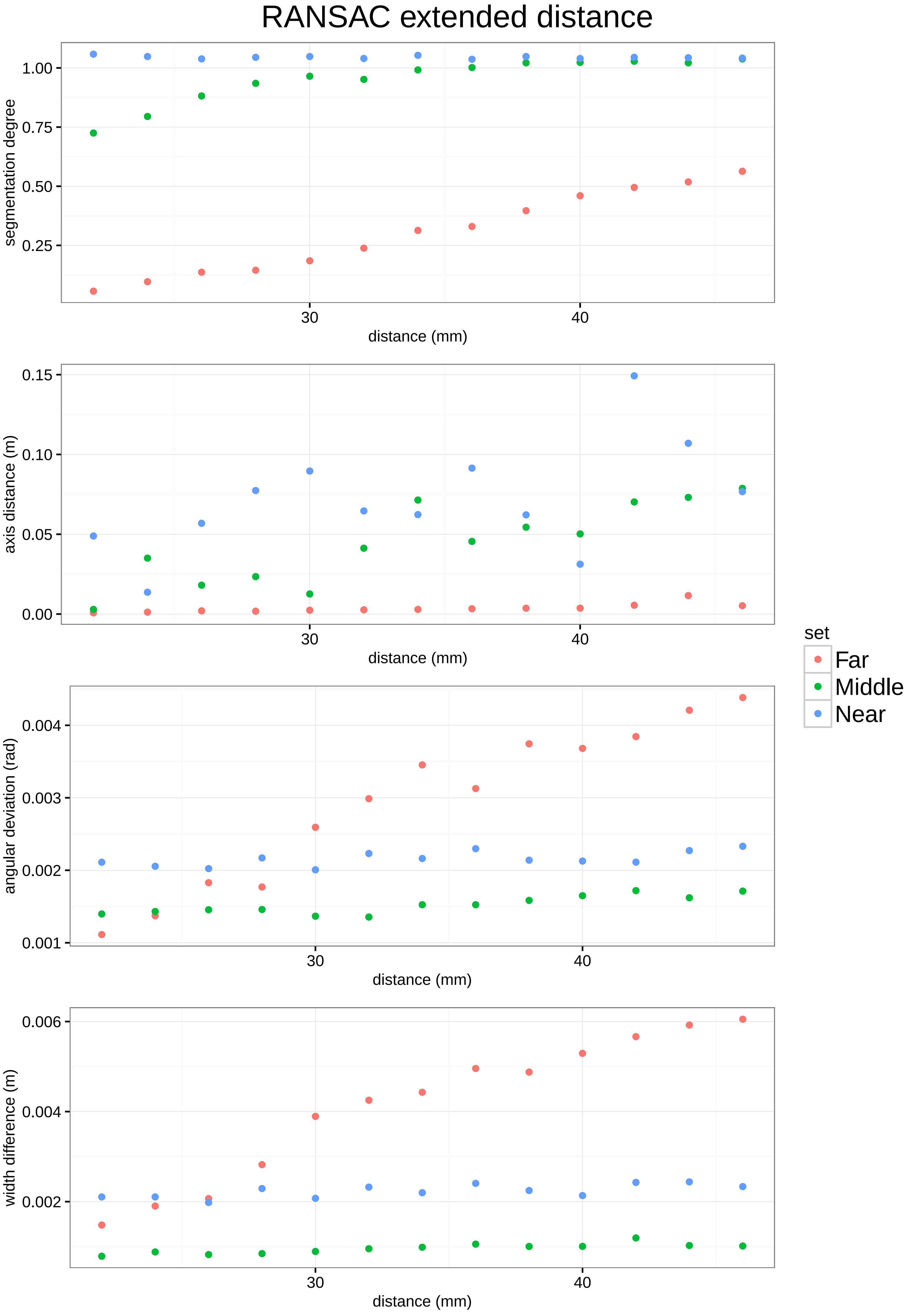
C.4.1 RANSAC

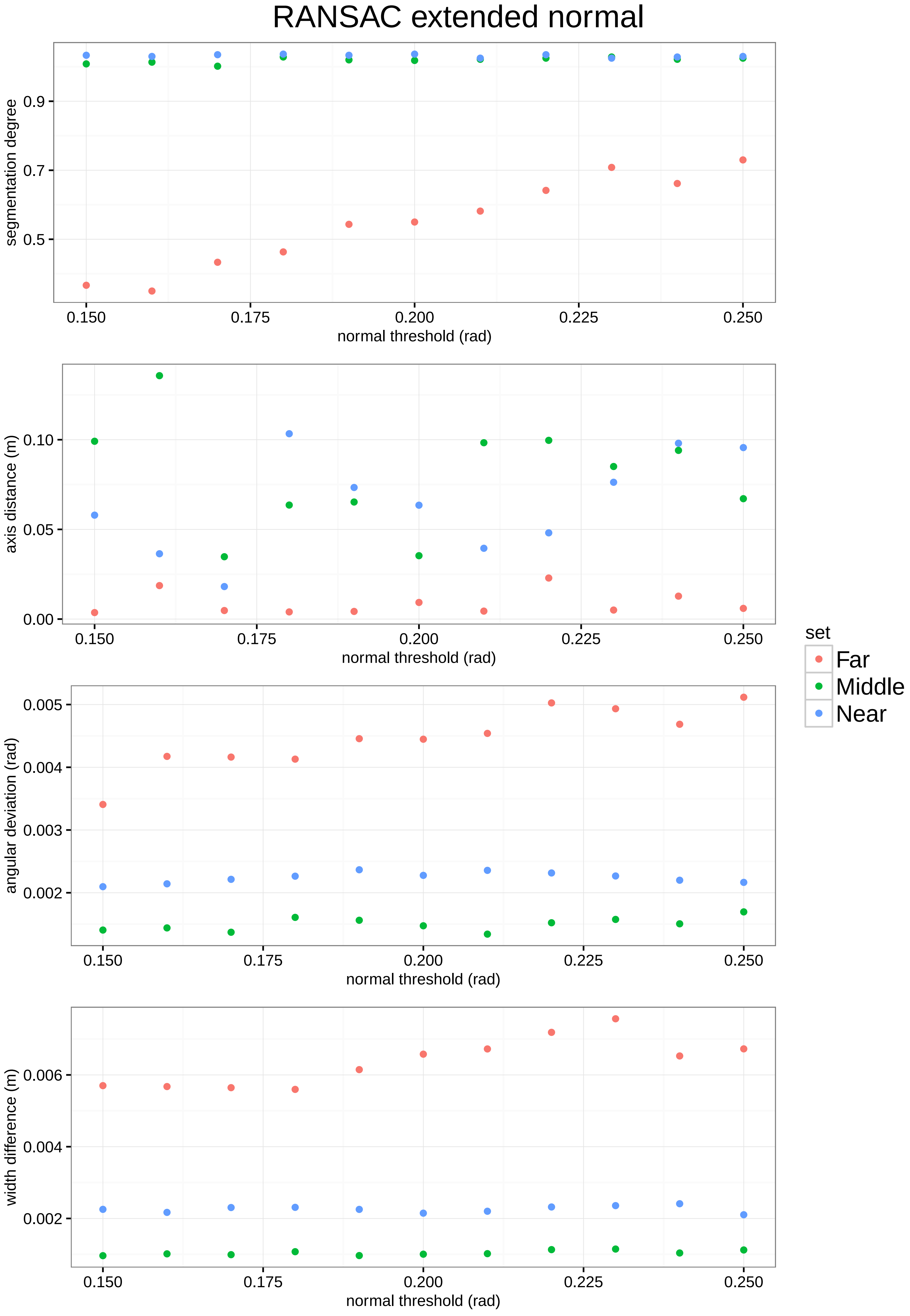
The most important observation from the results on the following page is that the only way to get a segmentation degree of one is to have a very low support threshold. Since 2% causes some over-segmentation and 4% under-segmentation we chose 3% as our final value.

To see what values we should choose for the other two parameters we extended their range and performed experiments with the normal threshold fixed at 0.15 radians and the support fixed at 3%. The plots can be seen on the second and third following pages. From the extended distance experiments we can conclude that 36 mm gives a good segmentation degree without sacrificing accuracy.

We fixed the distance threshold at 36 mm for the extended normal experiments. These experiments show that there is very little to be gained by increasing the threshold in combination with the previously mentioned values for the other parameters. Therefore we use the original value, 0.15 rad, in the algorithm.

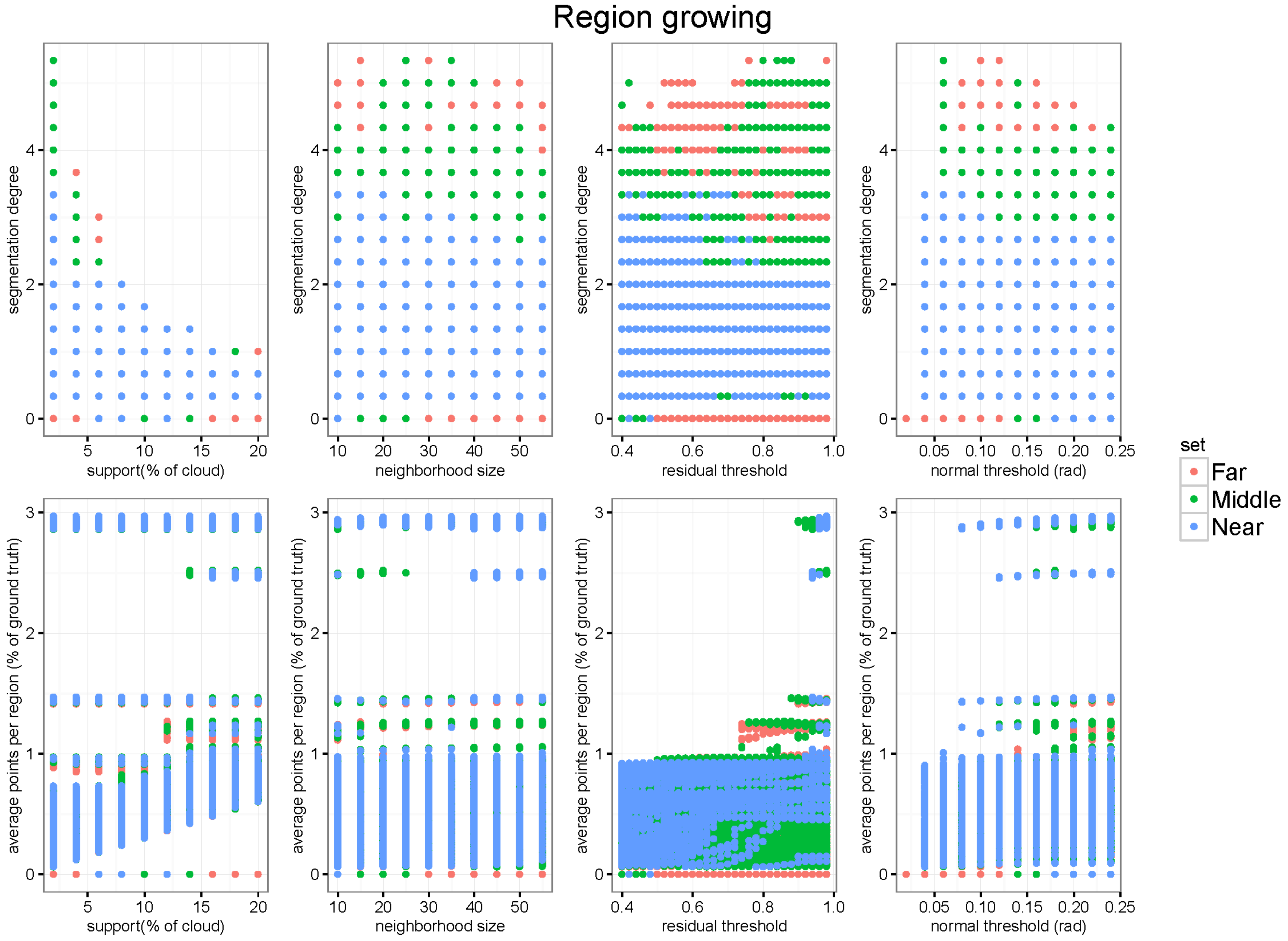

RANSAC
segmentation degree
axis distance (m)
angular deviation (rad)
width difference (m)
support(% of cloud)
distance (mm)
set
Far
Middle
Near


RANSAC extended distance
segmentation degree
1.00
0.75
0.50
0.25
30
40
distance (mm)
axis distance (m)
0.15
0.10
0.05
0.00
angular deviation (rad)
0.004
0.003
0.002
0.001
width difference (m)
0.006
0.004
0.002
set
Far
Middle
Near


RANSAC extended normal
segmentation degree
0.9
0.7
0.5
axis distance (m)
0.10
0.05
0.00
angular deviation (rad)
0.005
0.004
0.003
0.002
width difference (m)
0.006
0.004
0.002
0.150
0.175
0.200
0.225
0.250
normal threshold (rad)
set
Far
Middle
Near

C.4.2 Region growing

From the plots on the following page it appears that no parameter configuration gives a segmentation degree of one. The data set only provides three plausible spots to stop a region from growing, at the lines where the cylinders touch. Since the normals are not oriented this is very difficult for the algorithm to recognize. Since all segmentation results are multiples of 1/3 we can be sure the only region splits happened at the touching areas of the cylinders. Since these results do not give us any reason to deviate from the recommended values of [14] we do not deviate from these recommendations.


Region growing
segmentation degree
support(% of cloud)
neighborhood size
residual threshold
normal threshold (rad)
average points per region (% of ground truth)
set
Far
Middle
Near