Perception of Entangled Tubes for Automated Bin Picking

Gonçalo Leão[1], Carlos Costa[1,2], Armando Sousa[1,2], Germano Veiga[1,2]

[1] FEUP - Faculty of Engineering, University of Porto
[2] INESC TEC - INESC Technology and Science
goncalo.leao@fe.up.pt, carlos.m.costa@inesctec.pt, asousa@fe.up.pt, germano.veiga@inesctec.pt

Abstract. Bin picking is a challenging problem common to many industries, whose automation will lead to great economic benefits. This paper presents a method for estimating the pose of a set of randomly arranged bent tubes, highly subject to occlusions and entanglement. The approach involves using a depth sensor to obtain a point cloud of the bin. The algorithm begins by filtering the point cloud to remove noise and segmenting it using the surface normals. Tube sections are then modeled as cylinders that are fitted into each segment using RANSAC. Finally, the sections are combined into complete tubes by adopting a greedy heuristic based on the distance between their endpoints. Experimental results with a dataset created with a Zivid sensor show that this method is able to provide estimates with high accuracy for bins with up to ten tubes. Therefore, this solution has the potential of being integrated into fully automated bin picking systems.

Keywords: bin picking, industrial robots, linear objects, pose estimation, robot vision

1 Introduction

Bin picking corresponds to the task of taking an object out of a box with an open lid for subsequent manipulation [1]. It is usually decomposed into a set of sub-tasks, including object recognition, pose estimation and computation of an appropriate grasping position for a robot gripper to pick up a piece.

Since this task is present in many industrial processes, its automation will lead to substantial increases in productivity in a wide array of businesses. However, this problem is quite challenging as the items are often arranged in a random fashion, and are thus subject to occlusion and entanglement. One example of an item that is highly prone to entanglement are the wire-harnesses in the automotive industry. Due to its complexity, currently, there is no general solution to this problem.

This paper focuses on tubes and on the bin picking sub-task of pose estimation. The tubes are placed randomly in a bin, making them especially vulnerable to entanglement. The goal is to develop a computationally efficient algorithm to

process point clouds from a depth sensor, with the assumption that all the tubes have a common well-known radius, with a small variability. In most industrial scenarios, this assumption is realistic and acceptable. The algorithm makes no other assumptions regarding the tubes' geometric properties, namely their curvature, so that the solution can be applied to a varied set of industrial contexts.

Section 2 consists of a brief review of the literature regarding pose estimation for bin picking. Section 3 presents the approach to determine the tubes' geometry, detailing the experimental setup, the algorithm and how the solution was evaluated. Some results are also reported regarding the algorithm's computational efficiency and accuracy. Lastly, section 4 concludes with some discussions of the contributions and lines for future work.

2 Literature Review

Research for bin picking dates back 50 years, but much of the work that was published was not dedicated to bin picking itself but rather to one or more of its sub-tasks. The emergence of new technologies for depth sensing, namely active stereo, led to the creation of many vision-based techniques with increased accuracy and robustness.

Many approaches for pose estimation resorted to 3D models of the objects to be perceived. Bolles and Horaud [2] presented one of the earliest approaches to locate objects in bins using 3D sensors. They used the edges from the depth image to extract local features, such as circular arcs, and followed a hypothesize-and-verify paradigm to find matches between the image and the object model.

One recent project on object localization is DoraPicker [3], which participated in the 2015 Amazon Picking Challenge. This system begins by filtering its input cloud by downsampling with a voxel grid and a statistics-based outlier removal. It then computes an initial estimate of the object's pose using LINEMOD and refines it using the well-known Iterative Closest Point (ICP) algorithm.

Kita and Kawai [4] proposed a method to estimate the pose of twisted tubes for bin picking by applying a region growing-based segmentation using the surface normals and then determining the tube's principal axis by examining cross-sections of the segmented region. The estimate is then improved by using the tube's model and a modified version of ICP that uses a skeleton of the tube.

The model-less approaches have the advantage of handling intra-class shape variations, such as tubes with varying curvatures. However, this also requires a clean segmentation of the target objects. One very common method used to obtain the pose of a bin's objects without having a model apriori is by matching with shape primitives.

Cylinders are sometimes employed for the pose estimation of tubes. Taylor and Kleeman [5] describe a split-and-merge segmentation algorithm that begins by identifying smoothly connected regions with the surface normals. Surface elements, such as ridges and valleys, are then found to check if a given region's shape is consistent with a cylinder. If so, fitting is performed with a least squares solver, and regions are merged iteratively when the model for the combined

region has a lower residue. If the region is not consistent, it is split into smaller parts and the algorithm is applied recursively.

Qiu et al. [6] focused on reconstructing industrial site pipe-runs and performed a statistical analysis of the surface normals for global similarity acquisition, which they claim is more robust to noise than local features. This analysis involves using Random Sample Consensus (RANSAC) to obtain the cylinders' principal direction. Cylinder positions are then extracted via mean-shift clustering to detect circle centers on the orthogonal plane that contains the projection of the points belonging to cylinders with a given direction. Finally, pipe sections are joined using several criteria such as the distance and skew between cylinders.

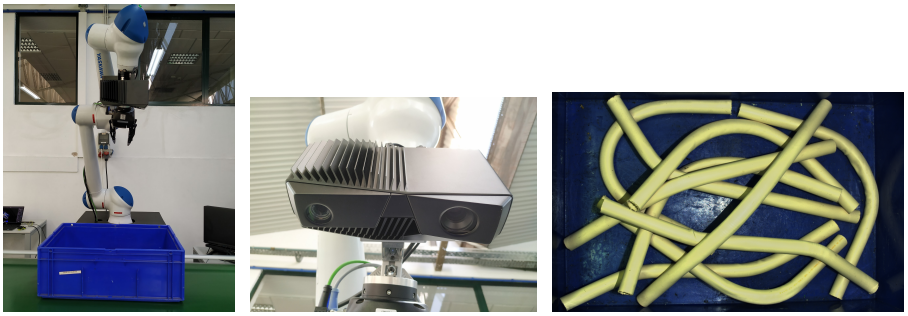
An alternative to cylinders is to fit splines to model curved tubes. Bauer and Polthier [7] used a moving least squares technique to compute a spine curve, using a partial 3D view of the tube. The spine is then approximated to an arc-line spline using heuristics.

3 Algorithm and experimental validation

3.1 Experimental Setup

A Zivid One Plus L depth sensor (Figure 1b) was chosen due to its high accuracy, having a spatial resolution of 0.45 mm (on the plane that is perpendicular to the sensor's optical axis) and a depth precision of 0.3 mm at 1.2 m. This high-end commercial device uses active stereo with structured light to acquire depth information and is also able to capture color, which was not used to show that this method is agnostic to the object's color.

The depth sensor was positioned looking downwards towards a bin filled with tubes for electric installations, with a vertical distance of 85 cm relative to the bin's bottom (Figure 1a). They were made out of Polyvinyl chloride (PVC), with a diameter of 2.5 cm and a length of 50 cm. The tubes were bent with arbitrary angles (Figure 1c). The bin's dimensions are 55 cm (length) x 37 cm (width) x 20 cm (height).



(a) Overview (b) Zivid sensor (c) Bin with bent PVC tubes

Fig. 1. Experimental setup

3.2 Point Cloud Processing

The program responsible for obtaining a model for the tubes from the sensor's point cloud was developed using Point Cloud Library (PCL), which contains a wide array of useful algorithms for point cloud processing.

The solution is divided into four main steps: filtering, segmentation, cylinder fitting and tube joining. Algorithm 1 presents an overview of the solution. Figure 3 shows an example of the proposed steps.

Algorithm 1 Overview of the tube perception algorithm

Input: $Cloud$ - Point cloud
Output: $Tubes$ - Set of tubes

1: $Cloud \leftarrow applyFilters(Cloud)$
2: $Segments \leftarrow regionGrowingSegmentation(Cloud)$
3: $Tubes \leftarrow \emptyset$
4: **for each** $segment \in Segments$ **do**
5: $\quad tube, inliers, outliers \leftarrow fitCylinder(segment)$
6: $\quad$ **while** $size(inliers) >= min_{inliers}$ **do**
7: $\quad\quad Tubes \leftarrow Tubes \cup tube$
8: $\quad\quad tube, inliers, outliers \leftarrow fitCylinder(outliers)$
9: $EndPointPairs \leftarrow \emptyset$
10: **for each** $pair \in EndPoints(Tubes) \times EndPoints(Tubes)$ **do**
11: $\quad$ **if** $isNearby(pair)$ **and** $areDifferentTubes(pair)$ **then**
12: $\quad\quad EndPointPairs \leftarrow EndPointPairs \cup pair$
13: $EndPointPairs \leftarrow sortByDistance(EndPointPairs)$
14: **for each** $(endPoint1, endPoint2) \in EndPointPairs$ **do**
15: $\quad$ **if** $areCompatible(endPoint1, endPoint2)$ **then**
16: $\quad\quad (tube1, tube2) \leftarrow (tubeOf(endPoint1), tubeOf(endPoint2))$
17: $\quad\quad Tubes \leftarrow Tubes \cup unite(tube1, tube2) \setminus \{tube1, tube2\}$

Filtering The first phase (line 1 in algorithm 1) filters the point cloud received from the sensor so that only points corresponding to the tubes remain.

Filtering begins with a pass-through filter to remove points whose depth lies outside a reasonable range. A random sampler reduces the number of points to a fixed amount by removing points with a uniform probability. A radius outlier filter erase points which do not have enough neighbors within a sphere of a given radius. These filters are efficient at removing noise since the regions of interest tend to form denser regions on the point cloud, comparatively to points due to noise.

This first set of three filters is intended to reduce the number of points where the surface normals will be computed. The normals are estimated using a least-square method to fit a tangent plane to each point using the covariance matrix formed by the neighboring points from the raw point cloud [8]. The curvature of each point is estimated by using the eigenvalues of this matrix. The unfiltered point cloud was used for the search surface so that there are

more points available around each neighborhood, thus increasing this operation's accuracy. Noisy points are not considered to have a significant impact in these estimates as they are greatly outnumbered by the relevant data.

The points from the plane belonging to the bin's bottom are then removed with RANSAC, an iterative, non-deterministic method that determines a model's coefficients by finding a sufficiently large subset of points (inliers) which are within a given distance from it. The distances from the points to the plane take into account both the point's coordinates and its normal vector. The minimum distance for a point to be considered as a inlier must be carefully chosen since if it is too small, the filter will not be able to remove points from the bin's bottom, and if it is too large, points belonging to the tubes may also be removed (the points whose normals are facing upwards are quite vulnerable to this problem), which will degrade the performance of the following steps of the algorithm. The advantages of using RANSAC include its simplicity and its robustness to outliers, even when in high proportion. The biggest drawback is that, due to its stochastic nature, it has no upper bound for the number of iterations needed to find a model that fits well the data. This leads to a trade-off between the execution time and the probability of computing an accurate model.

A second random sampler and radius outlier filter are applied to reduce the size of the resulting cloud to make the following processing steps more computationally efficient and remove points that may have remained from the bin's bottom.

The filtering process ends with a statistical outlier filter. This filter begins by computing the average μ and standard deviation σ of the distances of all the points to their k nearest neighbors [9]. The cloud's points are then scanned a second time and a point is considered as an outlier (and thus removed) if the average distance $\overline{d}$ follows inequality (1), where $mult$ is a multiplier that helps to regulate how restrictive the filter is. In this case, k and $mult$ were set to 100 and 3.0, respectively.

$$\overline{d} > \mu + mult * \sigma \tag{1}$$

Segmentation The second phase (line 2 in algorithm 1) aims to divide the filtered point cloud (which should only contain points from the tubes) into regions where the surface normal's direction varies smoothly. Each region corresponds to a continuous and non-occluded portion of a tube. Therefore, the points should be clustered so that each visible or partially visible tube is associated with at least one segment, and each segment pertains to exactly one tube.

To perform this segmentation, a region-growing algorithm is used where each region starts with a seed point assigned to it and is progressively expanded by adding nearby points for which the angle between their normal and the one for the seed point is below the smoothness threshold [10]. Additional points may be added as seeds for the same region if their curvature is sufficiently low. When there are no more seed points to grow a given region, the algorithm creates a new cluster and sets as the initial seed the unlabeled point with the lowest curvature.

The thresholds used by this algorithm must be properly defined to achieve a balance between over and under-segmentation.

Cylinder Fitting The third phase (lines 3-8 in algorithm 1) processes each segment independently. For each segment, the RANSAC algorithm repeatedly constructs cylinders, with the outliers of each cylinder serving as input for the next instance of RANSAC, until the number of remaining points is too low (below 100) for a new cylinder to be fit. Each call to the RANSAC algorithm runs up to 1000 iterations and attempts to find a cylinder with a radius between 0.7 cm and 1.5 cm. Associating multiple cylinders to each segment allows the algorithm to model curved tube sections.

The seven parameters for the cylinders' model are the 3D coordinates of a point and the three components of a unit vector to describe the axis, and its radius. The cylinder's length is not used as a parameter for RANSAC to reduce the number of iterations needed to produce good estimates for the other parameters. Instead, the length is computed after each run of the RANSAC algorithm by applying a rotation to the model's inliers so that the cylinder's axis is aligned with the z axis, and finding the inliers with minimum and maximum z coordinate. This method also allows the computation of both of the cylinder's endpoints, at center of the bases, which are used on the tube joining phase.

In order to avoid the overlapping of cylinders created from the same segment, after each run of RANSAC, the outliers within a slightly wider and longer cylinder with the same axis and center as the newly-created cylinder are removed. This bigger cylinder was set to be extended by 1 cm in length at both bases and to have a radius of 2 cm, which will always be larger than any reasonable cylinder fitted by RANSAC (since the tubes have a radius of 1.25 cm). The overlapping degrades significantly the performance of the tube joining phase since the algorithm may be unable to recombine both tubes into one.

Tube Joining The fourth and final phase (lines 9-17 in algorithm 1) consists in combining the cylinders created in the previous phase to form complete tubes. Each tube is modeled as a linked list of cylinders.

Initially, it is assumed that each cylinder corresponds to one and exactly one tube, even for cylinders that were generated by the same segment. One advantage of not considering all of the cylinders of the same segment to belong to the same tube is that the algorithm becomes more robust to occurrences of under-segmentation during the second phase.

This phase starts by considering all pairs of endpoints of distinct tubes and computing two distance metrics: the euclidean distance that separates both endpoints, and an angular distance, which describes how curved a junction would need to be to link both tubes. The angular distance, measured in degrees, is the sum of two angles formed by the unit vectors of the cylinder axes associated with each endpoint with a vector that links both endpoints, as depicted in Figure 2.

The endpoint pairs with an euclidean distance above 10 cm or angular distance above 90° are discarded since they are unlikely to belong to the same

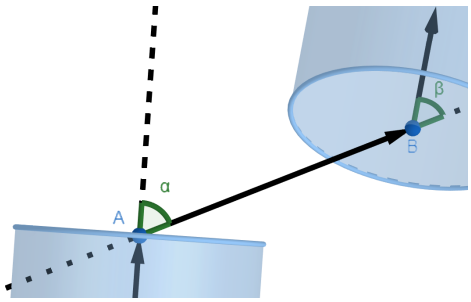


Fig. 2. Visual representation of the angular distance, sum of the angles α and β

tube. The remaining endpoint pairs are ordered using a distance metric that combines the euclidean and angular distance, as presented in equation (2). A coefficient was multiplied to the angular distance in an attempt to make both distances comparable (since one is expressed in meters and the other in degrees). The coefficient was set to 0.001 since the maximum acceptable distance for both metrics (10 cm and 90°) was assumed to be 'equivalent' (with $\frac{0.1}{90} \approx 0.001$). In the exceptional case where the euclidean distance is below 1.25 cm (the radius of a tube), it is very likely that the cylinders belong to the same tube, so the angular distance is set to 0 to increase the chances of the cylinders being combined in case there are not perfectly aligned (due to RANSAC's stochastic factors).

$$dist(A, B) = euclideanDist(A, B) + 0.001 * angularDist(A, B) \tag{2}$$

By adopting a greedy approach, the remaining endpoint pairs are processed in ascending order of distance, based on the rationale that the closer the cylinders are, the more likely they belong to the same tube. This is akin to the 'Joint Reconstruction' stage proposed by Qiu et al. [6], which also joins tubes based on their euclidean distance (which they call 'gap distance') and relative angles.

As each pair is processed, the endpoints' respective tubes are combined into a single tube that is modeled by the union of cylinders that belonged to both of the original cylinders. The resulting tube's length is estimated as the combined length of its cylinders lengths in addition to the euclidean distance between the endpoint connections. It should be noted that since it is common for the junctions between cylinders to be curved, this estimate is slightly lower than the actual length. If the cylinder overlapping problem was not solved in the previous phase, then these estimates would overshoot the actual length.

When processing the pairs, three constraints are applied to reduce the probability of a wrong junction of tubes. Firstly, the endpoints must belong to different tubes. Secondly, a 'length constraint' imposes that two tubes can only be joined if their combined length does not exceed 60 cm (a margin of error of 10 cm was added to the 50 cm tube length). This heuristic can be applied for other sorts of tubes if an upper bound for the length is known. Finally, a 'visibility constraint' prevents the union of two endpoints when there is an empty gap between them.

This was implemented by projecting both endpoints onto a 2D range image of the point cloud that resulted from the filtering phase from the sensor's point of view. Afterwards, the midpoint between both endpoint projections is computed and it is determined whether any pixel in a small neighborhood around this midpoint has less depth (i.e. closer to the sensor) than the maximum depth among both endpoints. If there is no such pixel, then the tubes cannot be combined. To the best of the author's knowledge, this is the first publication that presents these last two constraints for the problem of tube reconstruction.

3.3 Results

A dataset[3] with 50 point clouds of the bin with different amounts of tubes in various arrangements was constructed to evaluate the performance of the solution. There are five different test cases for each value for the number of tubes, ranging from 1 to 10. The tubes, bin and sensor properties are the same as those described in section 3.1. Tables 1 and 2 present the average value of different performance metrics with respect to the number of tubes.

The results in table 1 were obtained with an Intel Core i7-8750H processor, with 2.20 GHz. Each test case was evaluated five times to ensure more reliable execution times. The phase that takes the longest is the filtering, where the slowest operation was the plane fitting, as a large number of iterations was used for RANSAC. The increase of the filtering time with the number of tubes is likely due to a shift between the proportions of the points belonging to the bin's bottom and to the tubes: as there are less points on the bin's plane, RANSAC needs more iterations to converge to an acceptable model. It can be speculated that this increase in execution time should not increase indefinitely with the number of tubes and will stabilize once the amount of tubes is high enough for the bin's bottom to not be visible. The tube joining phase has a remarkably low execution time, with an order of magnitude of 1 ms.

Overall, the execution time of the perception algorithm has an order of magnitude of 1 s, which is rather reasonable for industrial applications.

To evaluate the performance of the segmentation phase, two annotators counted the number of visible continuous tube sections, using color images captured by the Zivid sensor for the 50 test cases. Ideally, there would be a one-to-one mapping between clusters and tube sections. The 'segmentation error' metric in table 2 is the relative error between the number of clusters produced by the segmentation phase and the number of tube sections. This error is the greatest for cases with one tube. This is likely caused by a moderate amount of leftover noise, due to imperfect removal of the bin's bottom plane. The increase of this error in cases with more tubes is due to some tube sections being too small (as a result of a growing number of occlusions), and thus not having enough points for the region growing algorithm to be able to create clusters for them.

[3] The 'Entangled Tubes Bin Picking' dataset is available at https://github.com/GoncaloLeao/Entangled-Tubes-Bin-Picking-Dataset.

Table 1. Average execution time for the algorithm's four phases

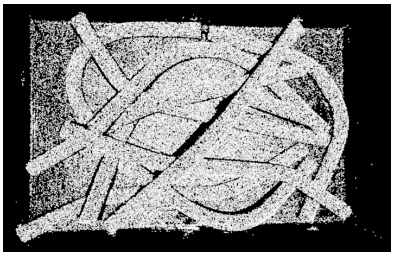
No. of tubes	Filtering time (s)	Segmentation time (s)	Fitting time (s)	Joining time (s)	Total time (s)
1	0.49	0.08	0.13	0.0036	0.70
2	0.60	0.14	0.16	0.0043	0.90
3	0.60	0.14	0.17	0.0050	0.92
4	0.60	0.14	0.21	0.0056	0.95
5	0.63	0.15	0.19	0.0069	0.97
6	0.62	0.14	0.20	0.0083	0.97
7	0.64	0.14	0.19	0.0089	0.98
8	0.69	0.14	0.19	0.0094	1.03
9	0.83	0.14	0.19	0.0095	1.17
10	0.94	0.14	0.20	0.0098	1.28

One metric used to assess the performance of the tube joining phase in a given test case are the average and standard deviation for the lengths of the tubes. Ideally, the average length should be 50 cm and the standard deviation should be minimal. According to table 2, the tube joining phase appears to have a good performance overall. As the number of tubes increases, it is natural for these metrics to worsen since the visible surface area of the tubes decrease, giving less information about each individual tube for the algorithm to work with.

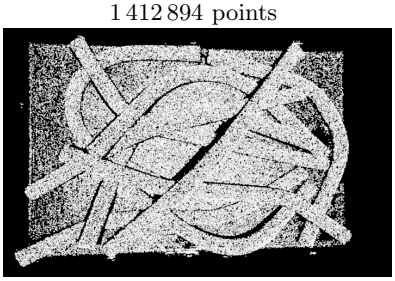
The tube length metrics are not sufficient to assess with great confidence the performance of the perception algorithm since the lengths are estimates and do not account for incorrect matchings between tube sections of similar length. Ergo, along with the number of tubes produced by the solution ('Number of joint tubes'), the annotators counted how many tubes were correctly and partially correctly modeled. A tube (in real-life) is considered to be 'correct' if it is associated with one and exactly one virtual tube which has a 'sufficient' amount of cylinders to cover its visible sections and does not have cylinders in sections of the bin where the tube is not present. A 'partially correct' tube only differs in the fact that multiple virtual tubes can be assigned to it. As seen in table 2, the algorithm performs well even in cases where the bin is fuller. It should be noted that, in a bin picking system, the bin is scanned after removing each tube, so it is acceptable if there are few partially correct tubes, as long as at least a few tubes are correctly modeled. As more tubes are removed, the remaining ones have more chances of having a correct model.

Figure 3 illustrates the full process for one of the test cases with ten tubes (test case '10_bin_picking2' from Figure 1c). The number of remaining points after each filter is also shown. In this case, the algorithm performed quite well since nine of the tubes are correctly modeled and the remaining one is partially correct (the two tube models corresponding to the partially correct one are marked with a red ellipse in Figure 3j). It is interesting to notice that in Figure 3h there was an occurrence of under-segmentation (marked with a red ellipse) that the cylinder fitting phase was able to recover from.

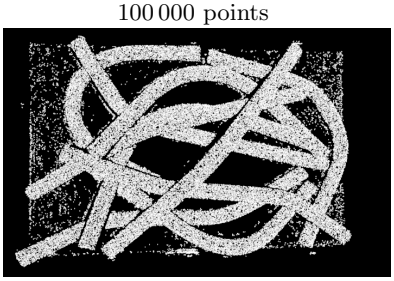
(a) Raw point cloud
1 412 894 points



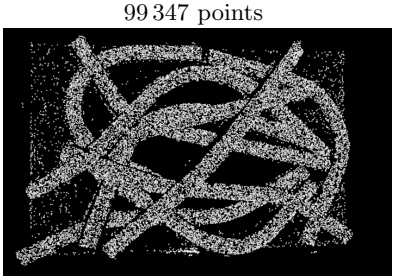
(b) After the first random sampler
100 000 points



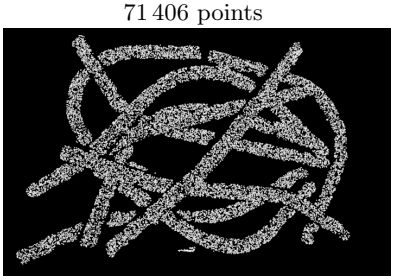
(c) After the first radius filter
99 347 points



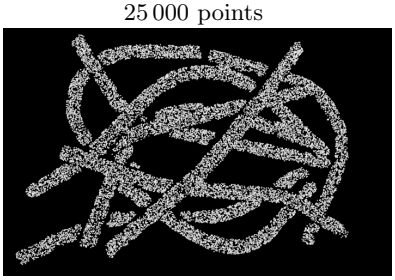
(d) After the plane removal filter
71 406 points



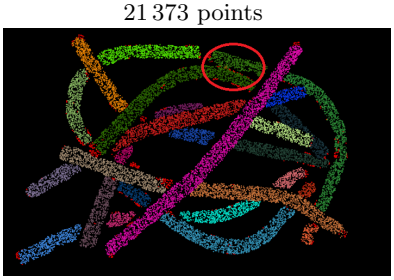
(e) After the second random sampler
25 000 points



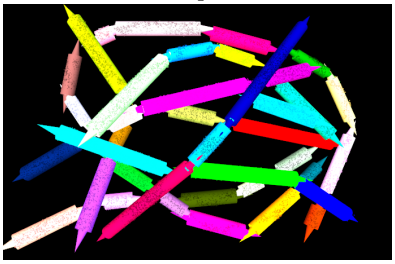
(f) After the second radius filter
21 373 points



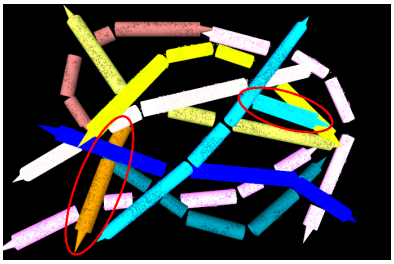
(g) After the statistical outlier filter
21 151 points



(h) After segmentation



(i) After fitting cylinders



(j) After joining tubes

Fig. 3. Results of the algorithm's steps for the case shown in Figure 1c

Table 2. Accuracy metrics for the segmentation, fitting and joining phases

No. of tubes	Segmentation error	Avg. length (m)	Std. deviation length (m)	No. of joint tubes	No. of correct tubes	No. of partially correct tubes
1	2.00	0.439	0.00105	1.2	0.8	0.2
2	0.83	0.439	0.08911	2.4	2	0
3	0.20	0.502	0.00824	3	3	0
4	0.00	0.468	0.03406	4.4	3.6	0.4
5	0.04	0.480	0.03661	5.2	4.8	0.2
6	0.00	0.481	0.03135	6.2	5.6	0.4
7	0.06	0.454	0.08529	7.6	6.4	0.6
8	0.06	0.414	0.12530	9.2	7	1
9	0.13	0.386	0.13101	10.6	6.6	2.2
10	0.12	0.347	0.15357	12.6	7	3

4 Conclusions and Future Work

The algorithm presented in this paper processes a point cloud from a depth sensor and provides a model for a set of tubes of equal radius but varying curvatures in a bin where they are arranged randomly. Using a primitive fitting approach for pose estimation, rather than starting from an initial 3D model of the tubes, allows the solution to handle intra-class variability. Some distance-based heuristics presented by Qiu et al. [6] alongside some novel constraints, such as checking for a gap between the cylinders, also enable it to deal with occlusions and entanglement. This is proven by the experimental results, where the solution was able to accurately describe the shape of most tubes in bins with up to ten tubes.

Using this solution's output, many heuristics can be used to select which tube to pick up next, such as choosing the one with the least occlusions. This solution can thus be integrated into bin picking systems of a vast variety of industries. Despite the tubes that were used in the experiments being rigid, this method has the potential of being compatible with flexible tubes, for which there are very few bin picking solutions.

Another relevant contribution is the 'Entangled Tubes Bin Picking' dataset, which can be used by the robotics community to benchmark other solutions to this challenging problem.

This work opens several lines for future research. Firstly, the solution's performance can be measured using different kinds of tubes, with other lengths and radii, possibly made of a more flexible material. Tests can also be conducted using other sensors, ideally those with less precision. These experiments can lead to an enrichment of the dataset. Secondly, the performance of the algorithm's fitting phase (execution time and accuracy) can be compared to an alternative where a spline is fitted to each segment, as suggested by Bauer and Polthier [7]. Little to no modifications will need to be done to the other three phases of the solution. Lastly, to decrease the algorithm's execution time, a decision procedure can be devised to decide if the plane filtering should be applied to the cloud.

Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 - The EU Framework Programme for Research and Innovation 2014-2020, under grant agreement No. 723658.

References

[1] Alonso M, Izaguirre A, Graña M (2019) Current Research Trends in Robot Grasping and Bin Picking. In: International Joint Conference SOCO'18-CISIS'18-ICEUTE'18, Springer Verlag, San Sebastian, Spain, vol 771, pp 367–376, DOI 10.1007/978-3-319-94120-2_35

[2] Bolles RC, Horaud RP (1986) 3DPO: A Three Dimensional Part Orientation System. The International Journal of Robotics Research 5(3):3–26, DOI 10.1177/027836498600500301

[3] Zhang H, Long P, Zhou D, Qian Z, Wang Z, Wan W, Manocha D, Park C, Hu T, Cao C, Chen Y, Chow M, Pan J (2016) DoraPicker: An autonomous picking system for general objects. In: IEEE International Conference on Automation Science and Engineering, IEEE, Fort Worth, TX, USA, pp 721–726, DOI 10.1109/COASE.2016.7743473

[4] Kita Y, Kawai Y (2015) Localization of freely curved pipes for bin picking. In: IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, IEEE, Luxembourg, Luxembourg, pp 1–8, DOI 10.1109/ETFA.2015.7301507

[5] Taylor G, Kleeman L (2003) Robust Range Data Segmentation Using Geometric Primitives for Robotic Applications. In: Proceedings of the Fifth IASTED International Conference on Signal and Image Processing, Acta Press, Honolulu, HI, USA, pp 467–472

[6] Qiu R, Zhou QY, Neumann U (2014) Pipe-run extraction and reconstruction from point clouds. In: Computer Vision - ECCV 2014. 13th European Conference. Proceedings: LNCS 8691, Springer International Publishing, Zurich, Switzerland, vol 8691, pp 17–30, DOI 10.1007/978-3-319-10578-9_2

[7] Bauer U, Polthier K (2007) Parametric reconstruction of bent tube surfaces. In: Proceedings - 2007 International Conference on Cyberworlds, CW'07, IEEE, Hannover, Germany, pp 465–474, DOI 10.1109/CW.2007.59

[8] Rusu RB (2010) Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. KI - Künstliche Intelligenz 24(4):345–348, DOI 10.1007/s13218-010-0059-6

[9] Rusu RB, Blodow N, Marton Z, Soos A, Beetz M (2007) Towards 3D object maps for autonomous household robots. In: IEEE International Conference on Intelligent Robots and Systems, IEEE, San Diego, CA, USA, pp 3191–3198, DOI 10.1109/IROS.2007.4399309

[10] Rabbani T, van den Heuvel F, Vosselmann G (2006) Segmentation of point clouds using smoothness constraint. In: Maas HGR, Schneider D (eds) ISPRS 2006 : Proceedings of the ISPRS commission V symposium, International Society for Photogrammetry and Remote Sensing (ISPRS), Dresden, Germany, vol 35, pp 248–253