FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Robotic Bin Picking of Entangled Tubes

Gonçalo da Mota Laranjeira Torres Leão

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Armando Jorge Miranda de Sousa (Ph.D.)

Co-Supervisor: Carlos Miguel Correia da Costa (M.Sc.)

Co-Supervisor: Germano Manuel Correia dos Santos Veiga (Ph.D.)

April 3, 2020

© Gonçalo da Mota Laranjeira Torres Leão, 2020

Robotic Bin Picking of Entangled Tubes

Gonçalo da Mota Laranjeira Torres Leão

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor Luís Paulo Gonçalves dos Reis

External Examiner: Doctor Pedro Mariano Simões Neto

Supervisor: Doctor Armando Jorge Miranda de Sousa

Co-Supervisor: Carlos Miguel Correia da Costa

Co-Supervisor: Doctor Germano Manuel Correia dos Santos Veiga

Gonçalo Leão

April 3, 2020

Abstract

Bin picking is a problem from Computer Vision and Robotics that involves taking an object from an unorganized heap using a robotic manipulator. This problem is present in many industrial processes, so its automation will enable an increase in productivity in many manufacturing systems, such as in the automotive and aircraft industries. In spite of the vast and mature body of research on bin picking, there are very few works that focus on entangled objects, where items are not easily separable, that is, cases where it is difficult for a robotic manipulator to pick a single item at a time.

The work associated with this dissertation focused on curved tubes, which are highly prone to occlusions and entanglement. This case study is highly relevant in manufacturing and production industries. The goal was to develop a system that enabled a robot arm (Yaskawa) with a parallel jaw gripper (Robotiq) to perform the picking operation in a heap of tubes, for which no predefined model was provided. The proposed solution began by constructing a model of the tubes by processing a point cloud acquired by a 3D scanner (Zivid or Photoneo). This representation of the heap of tubes allowed the detection of occlusions, and thus the identification of more easily pickable tubes. Grasp and motion planning were then performed to determine suitable grasping poses for the gripper and post-grasp trajectories that allowed a single object to be successfully picked. In cases where all the tubes have occlusions, the Gazebo simulator and ODE physics engine were used to evaluate the candidate trajectories and choose the one with the most potential for disentanglement. A force/torque sensor (ATI) was used to confirm whether a single tube was picked. If multiple (entangled) tubes were picked, then the manipulator performed a rotation based on the torque data in an attempt to make the inadvertently picked tubes drop back into the bin.

Results showed that the modeling algorithm was able to accurately describe the shape of most tubes in bins with up to ten items. Regarding the overall picking solution, real-life experiments with sets of bent PVC pipes and rubber radiator hoses showed that the robot was able pick a single tube on the first try with success rates of 99% and 93%, respectively. These experiments indicate that trajectory evaluation via simulation is very promising for picking operations as the physics engine was used on 17% of the picking attempts with the set of rubber hoses, with a success rate of 68%. There is also evidence that the entanglement resolution rotation based on the torque data is promising, with a usage rate of 8% and a success rate of 27% with the rubber hoses.

Computing suitable grasping poses and post-grasp trajectories for entangled tubes has the potential of increasing the variety of objects that can be handled efficiently by bin picking systems. This work thus serves as a stepping stone to the development of cost-effective, scalable robotic systems for handling entangled objects.

Keywords: bin picking; force sensing; grasp planning; industrial robots; linear objects; motion planning; pose and shape estimation; simulation; 3D perception

Resumo

Bin picking é um problema de Visão por Computador e Robótica que consiste em pegar num objeto de uma pilha desorganizada com um manipulador robótico. Este problema está presente em muitos processos industriais, pelo que a sua automatização conduzirá a um aumento de produtividade em muitos sistemas de fabrico, tais como nas indústrias automóvel e aeronáutica. Apesar do tópico de bin picking já possuir um corpo de pesquisa bastante alargado e maduro, existe muito pouco trabalho realizado no que toca a objetos emaranhados, que não são facilmente separáveis, ou seja, é difícil para o manipulador robótico pegar num objeto de cada vez.

O trabalho associado a esta dissertação focou-se em tubos com curvas, que são altamente propícios a oclusões e a se emaranharem uns nos outros. Este caso de estudo é de alta relevância para as indústrias de produção e fabrico. O objetivo passou por desenvolver um sistema que permitiu a um braço robótico (Yaskawa) com uma garra em forma de pinça (Robotiq) efetuar a operação de picking com um monte desordenado de tubos, para os quais não foi fornecido um modelo predefinido. A solução proposta começou por construir um modelo dos tubos através do processamento de uma nuvem de pontos adquirida por um scanner 3D (Zivid ou Photoneo). Esta representação do monte de tubos permitiu a deteção de oclusões, e, consequentemente, a identificação de tubos que são mais fáceis de pegar. As etapas de grasp e motion planning foram depois efetuadas para determinar poses de pega para a garra e movimentos após agarrar um tubo que permitiam que se pegasse com sucesso num único objeto. Nos casos onde todos os tubos tinham oclusões, o simulador Gazebo e o motor de física ODE foram usados para avaliar trajetórias candidatas e escolher aquela que tinha mais potencial para desemaranhar os tubos. Um sensor de força e torque (ATI) foi usado para confirmar se se pegou em apenas um tubo. Se tivesse pegado em vários tubos, o manipulador efetuava uma rotação baseada nos dados de torque numa tentativa de fazer com que os tubos que tinham sido inadvertidamente levantados caíssem de volta dentro da caixa.

Os resultados mostraram que o algoritmo de modelação foi capaz de descrever com precisão a forma da maioria dos tubos presente em caixas com até dez tubos. Relativamente à solução geral de picking, experiências na vida real com conjuntos de tubos dobrados feitos de PVC e de borracha mostraram que o robô foi capaz de pegar num único objeto à primeira tentativa com taxas de sucesso de 99% e 93%, respetivamente. Estas experiências indicam que a avaliação de trajetórias através da simulação é muito promissora para operações de picking já que o motor de física foi usado em 17% das tentativas de pega de tubos de borracha, com uma taxa de sucesso de 68%. Há também evidência que a rotação baseada em dados de torque para resolver casos de tubos que estão emaranhados é promissora, com uma taxa de utilização de 8% e de sucesso de 27% para os tubos de borracha.

O cálculo de boas poses de pega e de trajetórias após pegar num tubo tem o potencial de aumentar a variedade de objetos manipuláveis por sistemas de bin picking. Assim, este trabalho constitui uma contribuição relevante para o desenvolvimento de sistemas robóticos eficientes em termos de custo e escaláveis para o manuseamento de objetos emaranhados.

Acknowledgements

First of all, I would like to thank my supervisor, Professor Armando Sousa, for his constant feedback and helpful advice throughout my dissertation. His optimistic and enthusiastic personality motivated me to always go a step further.

I would also like to express my gratitude to my second supervisor Carlos Costa, for all his patience and assistance. He was always there to help me learn how to work with the robot's software and hardware.

I am also very grateful to my second supervisor Professor Germano Veiga, who was the first to inform me that robotic manipulation of entangled objects is currently a very promising research area.

I would like to thank all the staff at the Industry and Innovation Lab (iiLab) of INESC TEC, who were very kind to give me a place to work and to allow me to use their hardware (robot arms, grippers and sensors) for my experiments.

Last but certainly not least, I am very grateful to my family and friends for their constant support.

Gonçalo da Mota Laranjeira Torres Leão

"We can only see a short distance ahead,
but we can see plenty there that needs to be done."

Alan Turing

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

API	Application Programming Interface
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
DART	Dynamic Animation and Robotics Toolkit
EtherNet/IP	EtherNet/Industrial Protocol
GQ-CNN	Grasp Quality Convolutional Neural Network
GUI	Graphical User Interface
ICP	Iterative Closest Point
IFR	International Federation of Robotics
ODE	Open Dynamics Engine
OpenGL	Open Graphics Library
PCA	Principal Component Analysis
PCL	Point Cloud Library
PI^2	Policy Improvement with Path Integrals
POMDP	Partially Observable Markov Decision Process
PRM	Probabilistic Roadmap Method
PVC	Polyvinyl Chloride
RANSAC	Random Sample Consensus
RGB-D	Red Green Blue - Depth
ROS	Robot Operating System
RRT	Rapidly-exploring Random Trees
SIFT	Scale-Invariant Feature Transform
SME	Small and Medium-sized Enterprises
SVM	Support Vector Machine
TOF	Time-Of-Flight

Chapter 1

Introduction

This chapter presents the dissertation's context, as well as its objectives and motivation.

1.1 Context

The term *robot* was coined by Czech playwright Karel Čapek in the 1920 play entitled *R.U.R.* (*Rosumovi Umflí Roboti* or *Rossum's Artificial Robots*), which describes a factory that manufactures synthetic humanoids [Rob06]. It originates from the Czech word *robota*, which translates to *forced labor* or *compulsory service*[1]. While this expression portrays robots rather negatively, it conveys the important notion that they can perform tasks that are useful to Mankind. A more modern and accurate definition of a robot is that of an autonomous machine that can sense its environment, make decisions and perform actions in the real world[2].

Bin-picking corresponds to the problem of taking an object from a randomly stacked pile with a robotic gripper, typically mounted with vision sensors. This classical problem, that involves both Computer Vision and Robotics, is challenging due to the overlapping of objects that lead to occlusions. Other difficulties arise from the industrial settings where this problem is tackled. These include the complex geometry of industrial objects and the lighting reflections induced by metallic surfaces [RK96].

Despite the broad and mature body of research for this problem, the vast majority of the work has focused on non-entangled objects. Objects with certain shapes, namely long curved tubes, are particularly prone to entanglement, which is a major issue for picking systems, as it leads to undesirable situations where multiple objects are picked at once. This can cause disruptions in industrial lines, such as items falling out of working areas or more items than expected being received in a destination area. Moreover, objects with complex shapes are more likely to produce occlusions, which gives vision-based robots incomplete information about the environment. As

[1] https://www.etymonline.com/search?q=robot

[2] https://robots.ieee.org/learn

a result, many of the techniques used to address bin-picking on simpler cases cannot be directly applied to objects with more complex shapes.

1.2 Objectives

This dissertation focused on bin-picking for tube-shaped objects that have curves throughout their length and whose shape is not known beforehand (i.e. there is no predefined model for the object). For this work, it was assumed that the tubes' radius and mass were known beforehand, which are reasonable assumptions for most picking systems, as all the items in a bin tend to have very similar, if not the same, properties. The tubes in this study were pipes and hoses, as opposed to wires and strings, which have a considerably smaller radius. In addition, this work focused on objects that are rigid or semi-rigid.

The practical goal consisted in developing a bin picking system capable of handling entangled tube-shaped objects. This system used a robot arm with a depth and force/torque sensor, alongside a parallel jaw gripper. The solution used data collected from a 3D scanner to construct a representation of the tubes present in the bin. This model was used to determine the best pose (position and orientation) and movements for a robotic gripper to pick up, one by one, all the objects. A pairing of a *good* grasping pose and post-grasp trajectory was defined as one where closing the gripper's fingers and following the trajectory led to a single tube being removed from the bin. Bad grasping poses or trajectories were due to the entanglement between the various tubes in the bin, which caused multiple tubes to be picked up at once. Due to the complex geometry of the tube' entanglement, the system made use of a simulator and physics engine to predict how the objects moved when a certain tube was grasped and moved. This simulation aided in the selection of the best trajectory for more complex scenarios. In addition, the system included a failure detection mechanism to detect cases in which multiple tubes were picked or no tube was successfully grasped, by using data acquired by the force/torque sensor. This data was also used to compute a rotation for the robot arm to be able to solve cases where multiple tubes were picked by making the extra tubes drop back into the bin.

1.3 Motivation

Robot systems have the potential of increasing the efficiency, consistency and level of quality of a wide array of tasks in the manufacturing world. This increase in productivity may, in turn, lead to several positive effects for the society and economy, such as more company competitiveness, namely for Small and Medium-sized Enterprises (SME). According to a report from the International Federation of Robotics (IFR), it is estimated that the number of robots used in the industry will reach 3 million by 2020, which represents an average annual growth rate of 14% between 2017 and 2020 [Int17].

Tube-shaped objects are very common in manufacturing and production systems. Examples include refrigeration tubes in the automotive and aircraft industries, water pipes for plumbing installations and heat pipes in civil engineering. In many of these cases, the tubes contain multiple curvatures along their length. Bin picking is a problem shared by these industries since, when buying in bulk, it is not cost-effective for many of these components to be neatly packed into separate units, often due to their complex geometry. Thus, automating the bin-picking task has the potential of increasing significantly the productivity of many existing production and manufacturing systems by boosting the speed that they are supplied with materials.

Despite the significant advancements of the automation in the industrial environment, the manufacturing and assembly of products with entangled parts is still labor intensive. This is due to high dexterity and cognitive capabilities being required. Current solutions are both expensive and very context-specific, which severely limits their scalability to other manufacturing lines. This has safety and health consequences, as people perform potentially unhealthy or repetitive tasks, which may lead to an increase in job-related injuries or a decrease of the workers' satisfaction.

Therefore, the work done in this dissertation serves as a stepping stone to developing cost-effective, scalable industrial systems capable of dealing with entangled objects.

1.4 Research questions

The following questions were addressed by this dissertation:

RQ1 *How can the pose and shape of tubular objects be estimated?* Estimating both the pose and the shape can provide valuable clues for an adequate grasping point.

RQ2 *What are the criteria for good grasping configurations and trajectories for picking entangled objects?* Knowing the geometric properties of an adequate grasping pose and post-grasp trajectory is critical for the objects to be picked up with fewer difficulties.

RQ3 *How can existing bin picking solutions be applied for entangled tubes?* There might be algorithms designed for non-entangled objects that can be adapted to the problem at hand. At the very least, the vast body of knowledge on this subject provides ideas for new algorithms.

1.5 Document structure

The rest of this dissertation report is divided into 5 parts. Chapter 2 presents the state-of-the-art regarding bin picking, with a focus on tube-shaped objects and entanglement. Chapter 3 describes the most useful software elements for the work. The following two chapters present the full solution for bin picking of entangled tubes. Chapter 4 focuses on the initial steps of the algorithm, which create a model for the tubes from a point cloud. Chapter 5 presents the solution as a whole and details the following steps. Each of these two chapters include the description and results of experiments to evaluate the solution's performance and accuracy. The work devoted to these chapters led to the publication of two scientific articles [LCSV19, LCSV20]. Finally, Chapter 6 presents the conclusions of this dissertation, along with several lines for future research.

Chapter 2

Fundamentals on bin picking

The chapter presents the state-of-the-art regarding the bin picking task.

2.1 Bin picking process

Alonso et al. [AIG19] defined bin picking as "the action of taking objects from a bin (a box with its top side open) for subsequent manipulation or positioning in a different place and pose". In most cases, the objects are arranged randomly in the bin (*random bin picking*). Despite the existence of solutions with high commercial impact that are capable of solving particular cases, currently, there is no general solution to this engaging problem, according to Buchholz [Buc15].

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 [Buc15]. The most relevant sub-tasks include data acquisition, filtering, segmentation, pose estimation, grasp planning, motion planning and motion execution. These systems might perform additional tasks such as error detection and recovery [ZLZ+16], to deal with cases where the robot fails to grasp an item or accidentally drops it.

One very important requirement for bin picking solutions is the cycle time. In most industrial scenarios, the system should not take more than 10 seconds on average per item [DLB+08].

Each of the following subsections presents, in order, the most relevant sub-tasks of bin picking that were addressed by this dissertation's work. Figure 2.1 provides an overview of the upcoming subsections, by presenting the sequence of sub-tasks and their methods.

2.1.1 3D data acquisition

The first step of any bin picking system is to obtain information about the scene.

Nowadays, most bin picking systems rely on vision sensors to acquire three-dimensional information about the bin and its contents. This data can be represented as a point cloud, a set of points defined by their 3D position, or a depth map, a 2D image of the scene where each 'pixel'

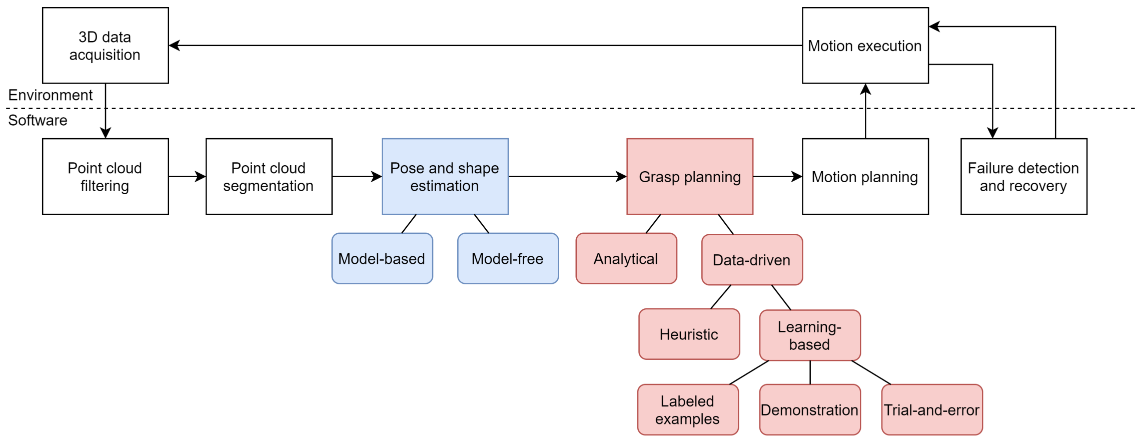


Figure 2.1: Overview of the bin picking process cycle and its methods

is encoded with the depth information from a particular viewpoint. It should be noted that it is possible to convert between both of these 3D data representations [CBR16], so the remainder of this dissertation will assume the depth information is represented using a point cloud, unless stated otherwise.

There are several types of 3D imaging techniques [STD09]. The most common ones include:

- stereo vision, where depth is perceived by comparing the displacement of corresponding points in a pair of images.

- structured light, where a pattern is projected to the scene and the 3D shapes are extracted by analyzing the pattern's deformation.

- Time-Of-Flight (TOF), where the amount of time between the emission and reception of a wave is used to compute the distances from the sensor to the objects. The wave that is emitted can be of various types, such as electromagnetic or mechanic.

Many 3D scanners used in bin picking systems also capture the scene's color, which can be used as a complement to depth on the subsequent sub-tasks. These sensors are commonly referred to as Red Green Blue - Depth (RGB-D) sensors [AIG19].

Alongside the 3D position of a point, the normal vector to the object's surface is typically used in bin-picking systems. Some sensor, such as Photoneo's scanners[1], are capable of performing this computation directly. They can also be estimated a posteriori, namely via a least-square method that fits a tangent plane to each point and uses the covariance matrix of its neighboring points from the cloud [Rus10].

An alternative to vision-based systems is to use tactile information. However, these *blind* bin-picking solutions are not robust and efficient [Buc15], and will thus not be explored.

[1]https://www.oem.ee/news/machine-vision/photoneophoxi3d

2.1.2 Point cloud filtering

The previous section stated that most bin picking systems use point clouds as input. These clouds are typically quite dense. Thus, to reduce the computational time of the next sub-tasks, it is important to reduce its size, namely by removing noise and information that is not relevant.

One common downsampling technique involves using a voxel grid filter, which divides the 3D space into a disjoint set of tiny cubes (voxels) of equal side length [SWJ17, ZLZ$^+$16]. Then, all the points within each voxel are replaced by their centroid.

Noise reduction due to sensor inaccuracy may be performed by removing points whose positions lie outside a reasonable range. Another frequently-used filter is the statistical outlier removal filter [GMLB12, MK15, ZLZ$^+$16], which analyses the distances between points and removes those that do not follow a certain global distance distribution (which is usually assumed to be Gaussian) [Rus10].

The points that belong to the planes of the bin's sides and bottom can be removed by using Random Sample Consensus (RANSAC) to fit planes to the cloud [MK15]. RANSAC is 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 [FB81]. The distances from the points to the plane take into account both the point's coordinates and its normal vector. The resulting inliers correspond to the points that should be removed.

2.1.3 Point cloud segmentation

Segmentation refers to the process of partitioning a point cloud into disjoint regions that are homogeneous and different from each other according to certain criteria [GD12]. In the specific context of bin picking, segmentation can be applied to divide the cloud into a set of smaller clouds, that each represent a visible portion of an object.

One of the most common and simplest segmentation algorithms is region growing, where neighboring points are iteratively added to a region if their similarity is sufficient high. Regions start off as small clusters of *seed points*. As the algorithm is executed, new seed points may be generated to create new regions. The similarity criteria may be based on the points' position, color and surface normals [AIG19]. One notable example of segmentation applied to bin picking is presented by Kita and Kawaii [KK15], who aimed to estimate the pose of rigid curved pipes. They resorted to region growing based on the surface normals, so that each segment corresponded to a smooth surface.

It should be noted that the segmentation of the point cloud may be interspersed with the pose estimation phase [TK03, QZN14].

2.1.4 Pose and shape estimation

The aim of this task is to estimate the position, orientation and shape of the objects in the bin [AIG19].

Methods for pose and shape estimation can be divided into two main families: the *model-based* approaches assume there is a well-known model of the object, while the *model-free* approaches do not make this assumption. The latter group is much more relevant to this dissertation since they also estimate each object's shape.

Despite the model-based methods being either inadequate or very complex to be used in this dissertation, they will be briefly presented.

2.1.4.1 Model-based methods

For this category of methods, all that is needed is to estimate the value of six parameters for each object, corresponding to their 3D position and orientation. The shape is known beforehand.

Bolles and Horaud [BH86] 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.

The most common technique for pose estimation is the well-known Iterative Closest Point (ICP) algorithm, which aims to represent the point cloud in the same reference frame as the model [CM91, BM92, Zha92]. In other words, it aims to find the optimal geometric transformation (translation and rotation) that aligns both shapes whilst minimizing the distance between them. This optimization is performed by iteratively refining the transformation.

The ICP algorithm requires an initial estimate of the pose. The initial pose can be obtained via template-matching methods, such as LINE-MOD [HHC$^+$11], which have a database of the objects in different poses and select the entry that minimizes a certain error function with respect to the perceived object's point cloud [BO08]. An alternative is to exploit directly the information provided by the depth image. For instance, Kita and Kawai [KK15] determined the tubes' principal axis (and thus the objects' orientation) by examining cross-sections of the depth image for each cluster produced by a region growing-based segmentation.

2.1.4.2 Model-free methods

Model-free approaches have the advantage of handling intra-class shape variations, such as tubes with varying curvatures.

One possibility is to apply machine learning techniques. Li et al. [LCA14] proposed building a training set composed of depth images of the target object with different poses, which were described as set of points on the object surface. Feature extraction was performed on each depth image using Dense Scale-Invariant Feature Transform (SIFT) descriptors, and a codebook was built using sparse coding. Finally, classification was performed using Support Vector Machines (SVM). Their approach is particularly interesting as it deals with deformable objects. One important disadvantage of using machine learning methods is that they require a large quantity of data for training, which may not be applicable in industrial settings if there is a large variety of items to be picked. A possibility to mitigate this issue is also proposed by Li et al. [LCA14], who acquired the training data via simulation, rather than using real sensors.

One very common method used to obtain the pose of a bin's objects without having a model apriori is by fitting mathematical models (usually shape primitives). This approach is interesting for bin picking scenarios, since no training is needed.

Cylinders are sometimes employed for the pose estimation of tubes. Taylor and Kleeman [TK03] described a split-and-merge segmentation algorithm that began by identifying smoothly connected regions with the surface normals. Surface elements, such as ridges and valleys, were then found to check if a given region's shape was consistent with a cylinder. If so, fitting was performed with a least squares solver, and regions were merged iteratively when the model for the combined region had a lower residue. If the region was not consistent, it was split into smaller parts and the algorithm was applied recursively.

Qiu et al. [QZN14] focused on reconstructing industrial site pipe-runs and performed a statistical analysis of the surface normals for global similarity acquisition, which they claimed it was more robust to noise than local features. This analysis involved using RANSAC to obtain the cylinders' principal direction. Cylinder positions were then extracted via mean-shift clustering to detect circle centers on the orthogonal plane that contained the projection of the points belonging to cylinders with a given axis direction. Finally, pipe sections were joined using several criteria such as the distance and skew between cylinders.

An alternative to using cylinders is to fit splines to model curved tubes. Bauer and Polthier [BP07] used a moving least squares technique to compute a spine curve, using a partial 3D view of the tube. The spine was then approximated to an arc-line spline using heuristics.

A third option is to model the objects using superquadrics [BZK+03], which are between cylinders and splines in terms of complexity. These primitives are a generalization of the quadrics, that can model many geometric shapes, namely cylinders with curvatures [Bar81]. These models are attractive since they can model a very large variety of shapes with a small amount of parameters. In particular, Wu and Levine [WL97] stated that the bending feature could be described using a single parameter (curvature). Figure 2.2 illustrates different shapes that can be modeled using superquadrics.

2.1.5 Grasp planning

The aim of this phase of the bin picking process is finding a collision-free configuration for robot gripper that allows it to pick up an object. Grasp planning can be divided into two sub-tasks: grasp synthesis, where the goal is to determine grasps that follow certain criteria, and grasp analysis, which evaluates the quality of the grasp [HSSR05]. In some of the publications, there is no clear separation between the grasp synthesis and analysis stages, which are performed together.

Leon et al. [LMSB14] defined a *grasp* as a "set of contacts on the surface of the object, which purpose is to constrain the potential movements of the object in the event of external disturbances".

Roa and Suárez published an extensive survey about grasp quality metrics [RS15].

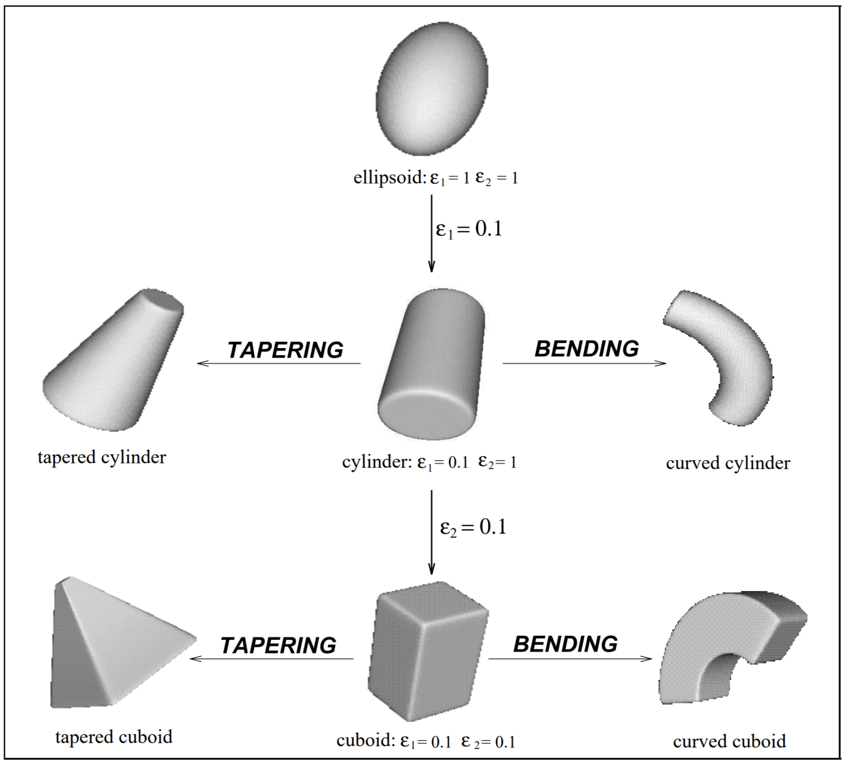

ellipsoid: $\varepsilon_1 = 1$ $\varepsilon_2 = 1$
$\varepsilon_1 = 0.1$
TAPERING
BENDING
tapered cylinder
cylinder: $\varepsilon_1 = 0.1$ $\varepsilon_2 = 1$
curved cylinder
$\varepsilon_2 = 0.1$
TAPERING
BENDING
tapered cuboid
cuboid: $\varepsilon_1 = 0.1$ $\varepsilon_2 = 0.1$
curved cuboid

Figure 2.2: Superquadric shapes obtained by varying some parameters [WL97]

According to Dupuis et al. [DLB+08], while there is much work in the area of robot vision that is specific to bin picking, the variety of approaches for grasp planning is more reduced. Thus, much of the research cited in this subsection is not focused on bin picking.

Common parameters for the grasp configuration [BMAK13] include:

- The grasping point on the object, that should be aligned with the gripper's tool center point.
- The approach vector, that indicates the 3D angle for the gripper as it closes in on the grasping point.
- The wrist orientation for the gripper.
- An initial finger configuration.

It should be noted that the grasp configuration information returned by these methods is not universal. Some methods defined the grasp configuration using the contact points only, which has the disadvantage of not describing how the gripper should be oriented or how wide its should be opened [CRC18].

The grasp synthesis methods are divided into two main categories: analytical and data-driven methods [BMAK13, SEKB12].

2.1.5.1 Analytical methods

Analytical methods aim to find grasps that follow certain physical properties. The grasp synthesis problem is typically formulated as an optimization problem with a series of constraints over several grasping properties. The grasp is commonly defined as a grasp map that converts the wrenches exerted by the gripper fingertips on the contact points to their impact on the object's center of mass [MLS94]. The notion of *wrench* is formally presented in definition 2.1.1.

Definition 2.1.1. A **wrench** W is a generalized force consisting of a linear and angular component that act on a point. It is represented as a vector in $\mathbb{R}^6$ as follows:

$$W = \begin{bmatrix} f \\ \tau \end{bmatrix}$$

where $f \in \mathbb{R}^3$ and $\tau \in \mathbb{R}^3$ are the linear and angular components, respectively [FC92].

Some of the most common grasp properties [Shi96, RS15] include:

- Force-closure: the grasp is resilient to any external wrench that may act upon the object.
- Dexterity: the gripper can move the object in a way that is compatible with the task being performed.
- Equilibrium: the combined wrench applied to the object (by the fingertips and external forces) is null.
- Stability: errors on the object's position caused by disturbances vanish after a certain amount of time when the disturbances disappear.

Several earlier works focused on grasping polyhedral objects [Ngu87, DLW00], while more recent publications dealt with objects of any shape, either specified by a triangular mesh [LLC03] or directly by a point cloud [LLD04, EKS09].

There is a substantial body of research in finding optimal force-closure grasps, where the goal consists in minimizing an objective function that reflects how far the grasp is from losing the closure property. Several heuristic optimization techniques were experimented, namely gradient descent [ZD04, ZW03].

To perform grasp analysis, Miller at et. used the GraspIt! simulator [MA04], which was developed by Columbia University Robotics Group. This grasping simulator was used to analyze, visualize and plan grasps using a variety of different hand models (including two-fingered grippers). One of its quality measures reflected the largest worst-case disturbance wrench that a unit-strength grasp can resist, as proposed by Ferrari and Canny [FC92].

A limited number of analytical approaches take into account the task the robot is trying to fulfill. Borst et al. [BFH04] defined a task wrench space, which contained the wrenches required to accomplish a given goal.

To sum up, one of the biggest advantages of the analytical methods is that they guarantee certain desirable grasping properties. However, most of them require a precise geometric model of the object. Another recurrent problem is the high computing time required for finding suitable candidates. Moreover, few of these approaches are task oriented, and the few that are suffer from the additional computational complexity of modeling the task [SEKB12].

2.1.5.2 Data-driven methods

Data-driven methods (sometimes referred to as *empirical* [SEKB12]) give more importance to the representation of the scene to determine the grasp candidates [BMAK13]. Unlike the analytical methods, they usually provide a more general description of the grasp (for instance, rather than specifying the each of the fingertips' positions, they can use a single grasping point). These methods started to become increasingly popular at the turn of the millennium, partially due to the availability of GraspIt! to the robotics community in 2004.

Bohg et al. [BMAK13] divided these approaches into two families: the heuristic methods use directly the sensory data, while the learning-based methods rely on previously acquired knowledge to find suitable grasping configurations. The learning-based methods can be categorized according to the source of knowledge: labeled data, human demonstration or trial and error.

The advantages of using these methods is that most of them tend to work well using sparse, incomplete and noisy sensor measurements.

Heuristic methods

One of the earliest relevant heuristic approaches was presented by Miller et al. in 2003 [MKCA03]. They modeled objects as a set of primitives, including boxes and cylinders (fig. 2.3) and then used several rules to define a set of grasping configurations. For instance, for cylinders, the object could be grasped by wrapping the gripper's fingers around the side surface or by grabbing one of the bases (fig. 2.4). One important aspect of their work is that they took into account the geometry of the robot gripper and arm for grasp planning: rather than just specifying the position of the contact points of the fingers with the object, they also indicated the 3D position of the palm, its approach vector and the fingers' orientations. Moreover, they filtered grasp configurations where there was a collision between the arm and the world model. They then tested each of the pre-defined grasps using GraspIt!. One weakness of their approach is that they required a complete model of the object to be picked, represented by a triangular mesh[2].

Richtsfeld and Vincze [RV08] found the top surface of a set of objects in a table by using a depth threshold on a triangular mesh. They then found their center of mass, which was inside the objects' surface, since they assumed the objects were convex. The first grasp point was the closest point on the surface's rim line to the center of mass. The second point intersected the rim with the line between the first grasp point and the center. Both points were shifted down to reduce the chances of slipping and it was checked if at least one of the grasp points still lied on a visible surface. The approach vector angle was the average of the minimum and maximum angles where there was no collision between the hand and the objects.

[2]https://graspit-simulator.github.io/build/html/data_files_bodies.html

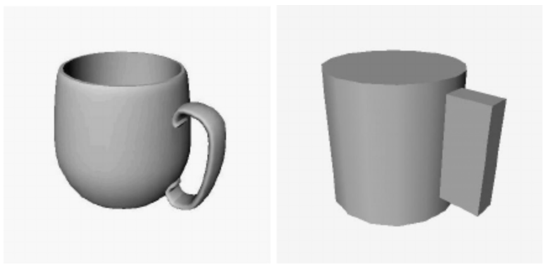


Figure 2.3: Approximation of a mug to a cylinder and a box [MKCA03]

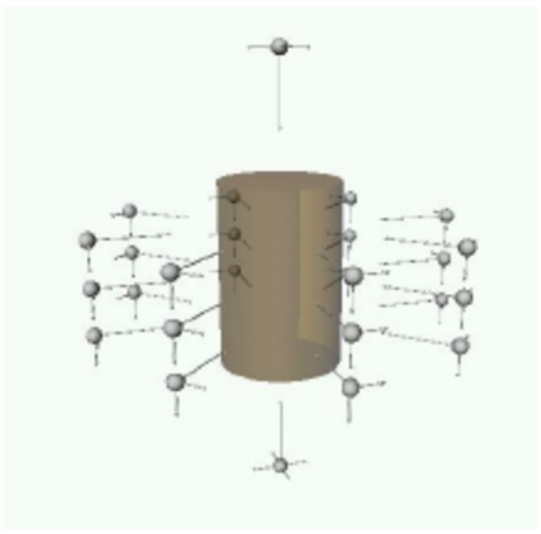


Figure 2.4: Pre-defined grasps for a cylinder: balls represent the position of the palm center, long arrows show the approach vector and short arrows show the thumb direction [MKCA03]

Rao et al. [RLP+10] adopted a similar approach, and also used a triangular mesh constructed from a 3D point cloud to sample initial grasp points on the object surface. By assuming the object was symmetric along its vertical axis, they computed the second point as the intersection between the surface normal line and the other side of the object. Finally, they computed the approach vector by discretizing its direction and performing collision detection between a plane that was perpendicular to this vector and the surface. The best approach angle was the one that maximized the distance between the intersection and the line formed by the grasping points.

Bohg et al. [BJRL+11] generated grasp candidates using OpenGRASP [LUD+10], another grasping simulator, which could detect collisions between the gripper and objects, and could also test for grasp closure. They claimed that generating and testing all grasps would take at least several minutes, so they only generated some candidates using several heuristics, such as the one proposed by Richtsfeld and Vincze [RV08].

Pas and Platt [tPP15] found grasping candidates by randomly sampling a set of points of the object's cloud and performed a grid-search using a discrete set of positions and orientations relative to a Darboux frame, which was aligned with point's surface normal and principal directions.

One approach to significantly reduce the search space for grasp candidates is to choose *which* item to pick before generating grasps along its surface. Common approaches include choosing the topmost object of the pile [YK86] or the one with the most free space (largest distance to adjacent objects) [WLCL09].

Simple heuristics can also be used for the grasp analysis, as a more straightforward alternative to using grasping simulators. Wong et al. [WLCL09] defined a quality metric that accounts for the proximity to major and minor pivot points, and to the nearest object.

One of the most attractive features of heuristic methods is their simplicity, compared to other grasp planning alternatives. One of the disadvantages is that these methods may contain several parameters that need to be adjusted to the type of object being picked [MHDW18]. This drawback is not very relevant for this dissertation since the scope is limited to tube-like objects.

Learning from labeled examples

Supervised learning methods aim to teach the robot to generate and evaluate grasp configurations by providing them examples of configurations where the grasp quality was computed beforehand.

Some approaches aim to train a binary classifier to determine if a configuration is adequate or not for a given scenario [SDN08, RLP+10, FV12, MHDW18], whereas others train a regressor, which outputs a metric that reflects the quality of the grasp [PMAJ04, PBC13] or its success rate [MLN+17].

Common learning techniques include SVM [PMAJ04, RLP+10, FV12, RAMNT12], logistic regression [SDN08] and artificial neural networks [PBC13]. These methods receive as input the grasp configuration parameters and attributes that describe the objects, such as the parameters of superquadrics [PMAJ04], or a region of the image, such as color and depth variance [RLP+10] or a bag of features descriptor [RAMNT12].

As an alternative, a depth image of the scene can be directly given to the learning model, to avoid pose estimation and feature engineering. In this case, the learning model is typically a Convolutional Neural Network (CNN) [MLN+17, MHDW18].

One drawback of supervised learning approaches is that they often require a large amount of training data, which can be rather slow and burdensome to perform when collecting data using real sensors [SDN08]. Moreover, if the labeling is performed manually, it is prone to human errors. Solutions to these issues include generating examples in simulation with physics engines [MHDW18] or using an existing dataset [MLN+17]. One notable dataset is Dex-Net[3] (Dexterity Network), which is maintained by the Berkeley AUTOLAB. Dex-Net 2.0 contains 6.7 million test cases of synthetic point clouds with parallel-jaw grasps. Some of the point clouds correspond to deformable objects, such as washcloths.

Figures 2.5 illustrates the approach followed by Mahler et al. [MLN+17], where grasp candidates are generated and then ranked with their Grasp Quality Convolutional Neural Network (GQ-CNN).

Matsumura et al. published a more recent article in November 2019 (in the midst of this dissertation's work) where they trained a CNN using simulated examples to determine if grasping a given candidate and applying a lifting motion led to entanglement [MDWH19]. To the best of the author's knowledge, this is the only paper that explicitly deals with random bin picking of entangled objects. For a given test case, they first detected grasp candidates by using their *graspability index*, which was computed by using binarized depth images of the top view of the bin with thresholds at different heights and two mask (binary) images. One mask described a

[3] https://berkeleyautomation.github.io/dex-net/

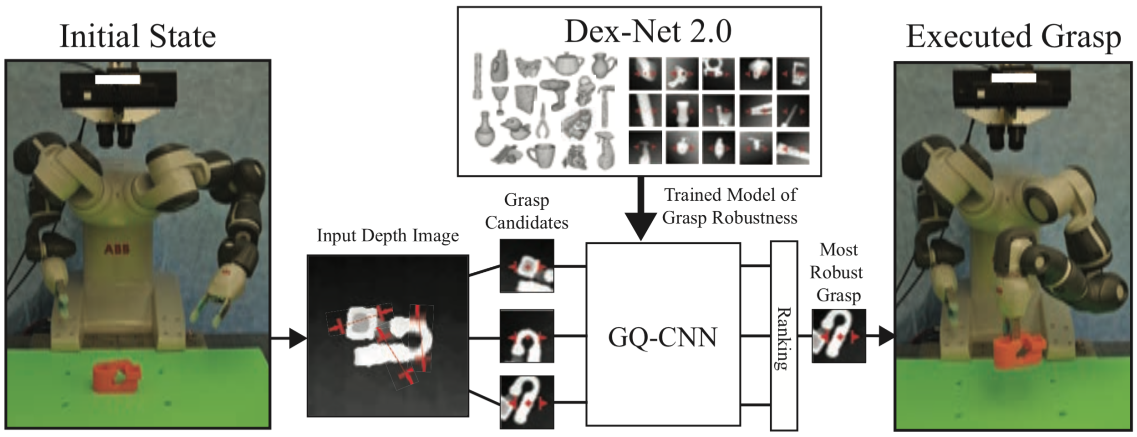


Figure 2.5: Grasp planning and execution with the GQ-CNN [MLN+17]

region of the image where the item should be present so that the grasp is stable, while the other one described the areas where the gripper was present during the grasping motion and which should not include any object to avoid collisions. After applying a series of operations with the images, including convolution and Gaussian bluring, they obtained a *graspability map*, where pixels with higher intensity values corresponded to more stable grasps. Then, they fed their CNN with a depth image that was cropped and rotated according to the grasp's configurations. This neural network would then output either *Tangled* or *Not tangled* to indicate if the grasp was expected to lead to multiple items being picked. Figures 2.6 and 2.7 provide an overview of the computation of the graspability index and the grasp candidate classification by the CNN, respectively. In one of their experiments, they trained the CNN with thousands of depth images of bins filled with 15 identical planar tubes with two right angles (*Object D*). They then experimented picking in real-life a tube from the bin, and were able to successfully pick a single item 17 out of 20 times. The time needed to determine a grasp was 1.9 s.

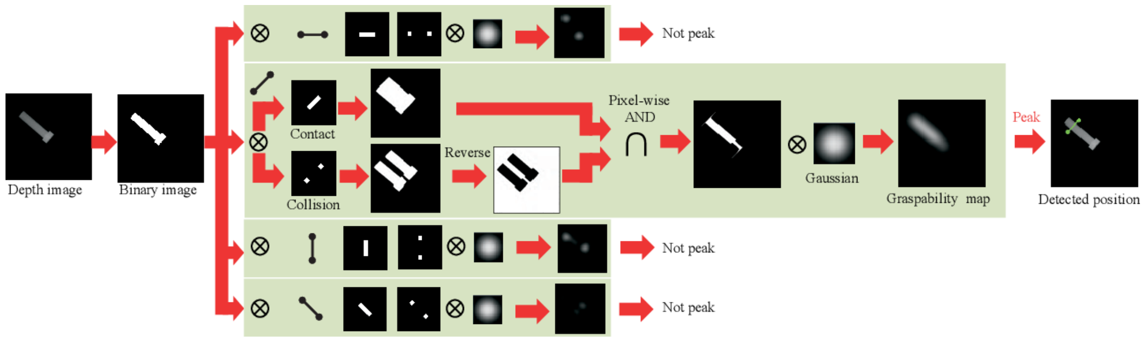


Figure 2.6: Graspability index computation [MDWH19]

Learning from demonstration

Learning from demonstration methods aim to learn a policy (i.e. a mapping that allows a robot to select an action based on the world's state) from examples provided by a teacher, which is usually

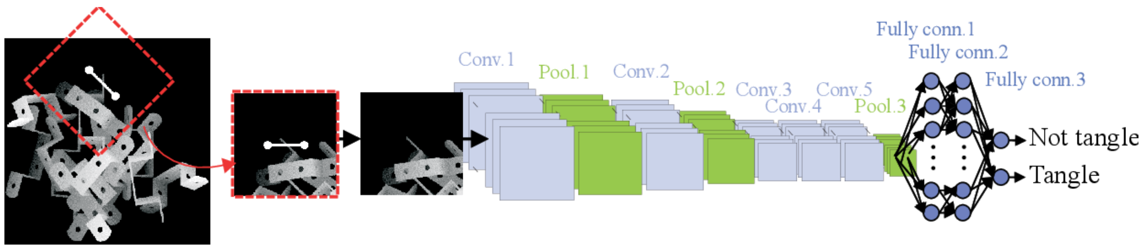


Figure 2.7: Grasp candidate classification (entanglement detection) with a CNN for a bin picking scenario [MDWH19]

human (but can also be another robot) [ACVB09].

Capturing the input grasping pose from the teacher can be accomplished with data gloves with magnetic [EK07] or bend sensors [DEKSVB15], as seen in figure 2.8. Another possibility is using a single image [RKK09]. While using datagloves has the advantage of providing exact values for the hand joint configurations, it may also prevent the user from performing a natural grasp.

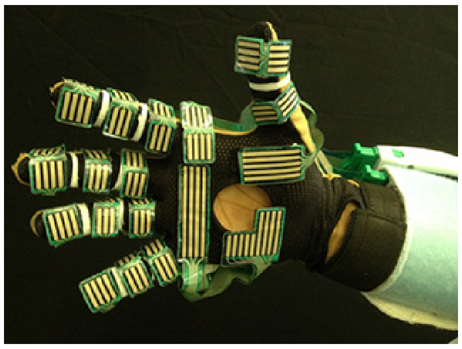


Figure 2.8: Dataglove with bend sensors [DEKSVB15]

When collecting data from the demonstrator, the physical and mechanical difference between the ‘teacher’ and ‘student’ hands must be taken into account, since they affect the quality of the learning process [DBP$^+$09]. In order for a parallel jaw gripper to learn from a human, multiple real fingers must be combined into virtual fingers [DEKSVB15].

Learning from trial-and-error

Trial-and-error methods aim to iteratively improve grasping poses. The initial grasp configurations can be obtained via human demonstration or one of the previously stated heuristic methods, for instance.

Reinforcement learning can be used when adopting trial-and-error approach, where a upper-level learner decides the grasp configuration and feeds it to a lower-level controller, which is responsible for executing it [KDPP10]. The controller then provides feedback to the learner about the quality of the grasp it performed, so that the latter may improve its grasp synthesis process. To assure the learning process converges quickly, there must be a balance between exploiting good grasping poses and exploring new ones. Examples of reinforcement learning algorithms include Q-learning [ATGD99] and the Policy Improvement with Path Integrals (PI^2) algorithm [STK+11].

One advantage of using reinforcement learning is that it does not need neither much prior knowledge, nor assumptions, akin to how babies learn how to grasp objects [KDPP10]. However, when using reinforcement learning, it is important that the state-action space has a low dimensionality, so that the learning process converges in a reasonable amount of time.

2.1.6 Motion planning

Motion planning is used to determine a trajectory for the robot gripper and arm, namely after grasping an object (i.e. a post-grasp trajectory). Most of the research focuses on the particular case of finding a trajectory that avoids the gripper and the grasped object from colliding with the environment [EP13, IE17, VVS17]. Elbanhawi et al. distinguished path planning from kinodynamic planning: while the former is only a geometric problem that focuses on finding a collision-free trajectory, the latter also takes into consideration the robot's kinematics and dynamics [ES14]. In addition, they pointed out that most techniques assume that the environment is static, though methods that handle moving obstacles already exist.

Motion planning approaches can be divided into two broad categories: local optimization techniques iteratively recompute a trajectory so that it moves away from collisions, while global search techniques use heuristics to explore the robot configuration space by focusing on its more promising sub-subsets or projections [IE17, KSLB19]. Local optimization algorithms include artificial potential fields, which define a set of repulsive forces on the obstacles and attractive forces on the goal so that the robot is progressively oriented to a suitable motion. Another class of local optimization algorithms are trajectory optimization methods such as TrajOpt [SDH+14], which use gradient descent to minimize an objective function subject to equality and inequality constraints (such as keeping the gripper horizontal or preventing each of the robot arm's joints from exceeding a certain velocity). The most common global search methods are sampling-based algorithms such as Rapidly-exploring Random Trees (RRT) [LaV98] or the Probabilistic Roadmap Method (PRM) [KSLO96], which incrementally explore the search space via random sampling. These methods have the advantage of being able to explore a high-dimensional search base rather efficiently, even though the resulting trajectory is often sub-optimal.

In the problem at hand, due to tube entanglement, in many cases, collision is unavoidable and the objects' pose may change as a certain tube is grasped, which must be taken into consideration when determining if a trajectory leads to multiple tubes being picked. Thus, this dissertation work proposed an alternative solution that involved evaluating trajectories using the Gazebo physics engine [KH04].

2.1.7 Failure detection and recovery

Failure detection is crucial in bin picking operations. If an item falls outside a robot's working area, it may compromise the following steps in a assembly line. For this reason, companies spend an expressive amount of money on regular quality checks for their products and manufacturing pipelines.

Force/torque sensors have been used several times to detect if pick-and-place operations were successful [YK91, MRP+16, CYK19]. Moreira et al. [MRP+16] focused on picking a set of entangled flexible tubes from a vehicle's air conditioning system. They assumed that grasp and motion planning routines already existed and that the mass of the tubes was known. By using the data from a 6 axis force/torque sensor coupled with machine learning algorithms, they determined if the tube slipped after being grasped. Kanoulas et al. [KLCT18] proposed a key strategy in their algorithm which is also used in this work: if the force readings lied outside the expected range, the robot released the object being grasped. Costanzo et al. [CDN19] presented the concept of manipulating a grasped object using force feedback. In their study, they used force/tactile sensors to adjust a parallel-jaw gripper's force so that it could rotate around its axis whilst maintaining the object's orientation. In this work, rather than adjusting the closing force, torque feedback was used to determine the gripper's axis of rotation so that non-grasped tubes that were inadvertently lifted were dropped back in the bin.

2.2 Bin picking solutions and projects

The section presents relevant commercial solutions and research projects related to bin picking.

Since bin picking is widespread in many industries, many commercial solutions are currently available. Software solutions include:

- AccuPick 3D[4] from Solomon.
- AUTOMAPPPS[5] from Convergent IT.
- CapSen PiC[6] from CapSen Robotics. In late 2019, they presented a solution for bin picking of entangled hooks[7]. Their approach to disentangle the took seems to involve performing a fixed trajectory with a robot arm that rotates the gripper in various directions. This movement is performed even when the robot picks a single hook.
- FlexiPick[8] from Bluewrist.
- InPicker[9] from Infaimon.
- MVTec HALCON[10] from MVTec Software GmbH.
- PickWorker[11] from Mujin.

[4] https://www.solomon-3d.com/product/bin-picking
[5] http://convergent-it.com/products/robot-bin-picking/
[6] http://www.capsenrobotics.com/capsen-pic.html
[7] https://www.youtube.com/watch?v=fcvulzMQ1kg
[8] http://bluewrist.com/test/robot-bin-picking
[9] https://inpicker.com
[10] https://www.mvtec.com/company/reference/detail/bin-picking-application-with-mvtec-halcon-boosts-productivity
[11] https://www.mujin.co.jp/en/solution/fa/picking.html

- SuperPick[12] from Soft Robotics.
- Various solutions[13] from Photoneo. Their AnyPick solution does not require the user to provide a model of the object.
- Various solutions[14] from Pick-it. Their solutions have an interface with ROS.

Some of these vendors (such as Solomon and Infaimon) also sell some of the robot hardware required to perform the bin picking task, namely 3D scanners.

Several research groups are also working on projects to develop more advanced bin picking systems. Some relevant projects are presented in table 2.1. The 'Participants' column includes the country of the company/organization's headquarters.

Table 2.1: Bin picking research projects

Project name	**Participants**	**Duration**	**Notes**
Auto3D[15]	• SINTEF (Norway) • Tordivel (Norway)	01/01/2010 - 01/12/2010	• Resulted in the development of the 3DMaMa algorithm, for 3D object localization and pose estimation, which requires a Computer-Aided Design (CAD) model of the object as input.

[12] https://www.softroboticsinc.com/superpick
[13] https://www.photoneo.com/bin-picking/
[14] https://www.pickit3d.com/product/pick-it-software
[15] https://www.sintef.no/en/projects/auto3d-random-bin-picking/

STAMINA[16,17] (Sustainable and Reliable Robotics for Part Handling in Manufacturing Automation)	• Aalborg University (Denmark) • Albert Ludwig University of Freiburg (Germany) • BA Healthcare (France) • BA Systèmes (France) • Heriot-Watt University (United Kingdom) • INESC TEC (Portugal) • PSA Automobiles (France) • University of Bonn (Germany) • University of Edinburgh (United Kingdom)	01/10/2013 - 31/03/2017	• They aimed to develop various industrial robots to perform several logistic and handling tasks, namely bin picking. • They developed a bin picking system to work with similar engine tube connectors to the ones used in this dissertation [GAB16]. Their approach involved fitting cylinders to the tubes and grasping the one with the highest centroid. The grasp pose takes into account the cylinder's axis orientation. • They achieved a 100% success rate in their experiments, with an average cycle time of 27s, which was mostly spent on motion planning and execution. • The tubes used in these experiments are the same as those in the article by Moreira et al. [MRP+16].

16 https://cordis.europa.eu/project/rcn/110654/factsheet/en
17 http://criis.inesctec.pt/index.php/criis-projects/stamina/

BINPICKING 3D[18] (Generalized bin picking system for automatic handling of unsorted parts in industrial applications)	• Infaimon (Spain)	01/07/2015 - 31/12/2015	• One of their planned innovative aspects was to develop a solution that was agnostic to the objects to be picked and the robot's hardware, namely by locating the sensor in the gripper. • Resulted in the development of the InPicker solution. • Results showed that the solution did not allow proper recognition of deformable objects.
PICKPLACE[19] (Flexible, safe and dependable robotic part handling in industrial environments)	• Fraunhofer Society (Germany) • Mondragon Assembly (Spain) • National Research Council (Italy) • Tekniker (Spain) • Tofaş (Turkey) • ULMA Handling Systems (Spain)	01/01/2018 - 31/12/2020	• One of their planned innovative aspects is to develop a pick-and-place solution that can handle parts with high variability.

[18] https://cordis.europa.eu/project/rcn/197393/factsheet/en
[19] https://cordis.europa.eu/project/rcn/213143/factsheet/en

SoftHandler 3D[20] (Commercial feasibility of an integrated soft robotic system for industrial handling)	• Italian Institute of Technology (Italy) • qbrobotics (Italy)	01/03/2019 - 31/08/2020	• One of their planned innovative aspects is to develop a pick-and-place solution that can handle parts with high variability, using the SoftHands[21] gripper, which aims to mimic the human hand more closely.

2.3 Bin picking of entangled objects

To assess if the problem of bin picking of entangled objects has already been tackled, or if its resolution would constitute a relevant contribution to the state-of-the-art, a query was made in four research databases on March 11, 2020. The search query was performed on the title and abstract fields of the papers. It was composed of four sub-queries combined with a logical *AND* that aimed to retrieve:

- Publications dealing with robotics.
- Publications where something is grasped, handled or manipulated.
- Publications where objects are entangled or twisted.
- Publications where objects are *tube-like*.

Table 2.2 presents the results. A paper was considered *relevant* if it presented a grasp planning approach to deal with a set of entangled *tube-like* objects (not necessarily for the purpose of bin picking). Most of the papers were not interesting as they instead focused on twisting motions or on robots with cylindrical components.

Only two papers passed this *relevancy test*, and they were both present in all four databases. One of them deals with untangling a single rope with two grippers [LS13], while the other focuses on bin picking of rigid twisted tubular objects and does not present a detailed approach for grasp planning in scenarios of entanglement [WCK+94].

A second, more broad research was performed (also on March 11, 2020) to find publications that dealt with picking or manipulating objects that are not necessarily shaped like tubes. The query was composed of three sub-queries combined with a logical *AND* that aimed to retrieve:

- Publications dealing with robotics.

[20] https://cordis.europa.eu/project/rcn/220708/factsheet/en
[21] https://www.softhands.eu/

Table 2.2: Entangled tube grasping query

Research database	**Search command**	**No. results**	**No. relevant results**
Engineering Village	(((((robot*) WN KY) AND ((grasp* OR handl* OR manip*) WN KY)) AND ((*tangle* OR twist*) WN KY)) AND ((cylind* OR hose* OR pipe* OR rope* OR tub*) WN KY))	36	2 [LS13] [WCK+94]
Scopus	(TITLE-ABS-KEY (robot*) AND TITLE-ABS-KEY (grasp* OR handl* OR manip*) AND TITLE-ABS-KEY (*tangle* OR twist*) AND TITLE-ABS-KEY (cylind* OR hose* OR pipe* OR rope* OR tub*))	44	2 [LS13] [WCK+94]
Web of Science	TOPIC: (robot*) AND TOPIC: (grasp* OR handl* OR manip*) AND TOPIC: (*tangle* OR twist*) AND TOPIC: (cylind* OR hose* OR pipe* OR rope* OR tub*)	35	2 [LS13] [WCK+94]
IEEE Xplore	(robot*) AND (grasp* OR handl* OR manip*) AND (*tangle* OR twist*) AND (cylind* OR hose* OR pipe* OR rope* OR tub*)	33	2 [LS13] [WCK+94]

- Publications where something is grasped, handled, manipulated or picked.
- Publications where objects are entangled (*twisted* was not added to this query, as it resulted in over 400 results being retrieved in each of the databases, with too many of them being irrelevant to this work).

Table 2.3 presents the results. In this case, a paper was considered to be *relevant* if it dealt with picking or handling objects (of any sort) in a pile and if it was explicitly stated in the abstract that the objects were entangled. Most of the article that came up in the results were not relevant since they had the word *rectangle* in their abstract (rather than *tangled*, *entangled* ...).

Four different articles were considered as relevant. One of the articles was the one by Matsumura et al. [MDWH19] presented in Section 2.1.5.2. The other three articles were found with this search:

- Yoshida et al. [YHH+09] used a force sensor to determine how many clothes were picked from a pile, similarly to what was done by Moreira et al. [MRP+16] and described in Section 2.1.7. The difference between their approach is that Yoshida et al. identified cases of entanglement by detecting abrupt changes on the force data.
- Monsó et al. [MAT12] presented an approach to pick a single piece of clothing from a disorganized pile, where the items were subject to both occlusions and entanglement. They

Table 2.3: Entangled object grasping query

Research database	**Search command**	**No. results**	**No. relevant results**
Engineering Village	(((((robot*) WN KY) AND ((grasp* OR handl* OR manip* OR pick*) WN KY)) AND ((*tangle*) WN KY)))	120	3 [YHH+09] [MAT12] [MDWH19]
Scopus	(TITLE-ABS-KEY (robot*) AND TITLE-ABS-KEY (grasp* OR handl* OR manip* OR pick*) AND TITLE-ABS-KEY (*tangle*))	142	3 [YHH+09] [MAT12] [KSKG16]
Web of Science	TOPIC: (robot*) AND TOPIC: (grasp* OR handl* OR manip* OR pick*) AND TOPIC: (*tangle*)	83	3 [YHH+09] [MAT12] [KSKG16]
IEEE Xplore	(robot*) AND (grasp* OR handl* OR manip* OR pick*) AND (*tangle*)	89	3 [YHH+09] [MAT12] [MDWH19]

predefined a set of 20 post-grasp motions for a three-fingered gripper that included lifting a cloth or moving it from one side of the working area to the other. For the grasping point, they chose the one that was the highest on the point cloud or the most *wrinkled* one. The motion was chosen by Partially Observable Markov Decision Process (POMDP) planning, a *Learning from trial-and-error* technique, based on the computed grasp's configuration.

- Kaipa et al. [KSKG16] worked with CAD models of the objects and estimated the relative posture of an object to be picked and an occluding object. These relative postures reflected the convexity of their regions of contact. Based on these postures, they defined a set of conservative, pessimist rules to determine if picking a given object with a lifting motion led to multiple objects being picked. They defined as future work generalizing their approach for interactions with multiple occluding objects.

Based on the previous discussion, to the best of the author's knowledge, the scientific article by Matsumura et al. [MDWH19] is the only one that presents a complete approach for picking of entangled objects, which they mention themselves in the manuscript. A very reduced number of papers explicitly deal with entangled objects, but they only focus on some of the sub-tasks for bin picking. Therefore, it can be concluded that this dissertation work is innovative and represents an important contribution to the state-of-the-art.

2.4 Summary

This chapter started by presenting in detail the bin picking process, giving more emphasis to the phases that are more relevant to this dissertation's problem. Several commercial solutions and research projects related to bin picking were also introduced. This review of the state-of-the-art was completed with a thorough survey which established that there is very few little work on bin picking systems capable of handling entangled objects, thus highlighting the innovative aspects of this work.

Chapter 3

Relevant tools

This chapter presents the most relevant software tools for the dissertation' work. The most important properties of each component are detailed, as well as a justification for why they were useful.

3.1 ROS

Robot Operating System (ROS[1]) is an open-source meta-operating system with the purpose of facilitating the development of software for robots. It is referred to as a *meta-operating system* since, despite providing the typical services of an operating system (such as hardware abstraction and message-passing between processes), it is intended to be run on another operating system.

A very useful book entitled *A Gentle Introduction to ROS* [O'K14] provides a rather complete and user-friendly tutorial to ROS.

ROS has different distributions[2], which correspond to versioned sets of packages (similar to Linux's distributions). All of the work for this dissertation was done on the Melodic Morenia distribution for being the most recent one. As a result, the host operating system that was chosen was Ubuntu 18.04 (Bionic), the primary target for the Melodic distribution.

ROS deals with four fundamental concepts: nodes, messages, topics and services. Nodes correspond to processes and are responsible for performing computations. Messages are a data structure used by nodes to communicate amongst themselves. Messages sent by nodes are published in topics. In order for a node to be able to receive certain kinds of messages, it must subscribe to the corresponding topic. Finally, unlike topics, which are used for asynchronous communication and use a broadcasting scheme, services are used for synchronous communication between nodes using a request–response scheme [QGC+09].

[1] `http://wiki.ros.org/ROS/Introduction`
[2] `http://wiki.ros.org/Distributions`

Software in ROS is organized into packages, which can include nodes and ROS-independent pieces of software. This organization allows modules to be easily reused in other projects.

The system developed for this dissertation used ROS due to its architecture, which enables nodes to be developed in a modular fashion (i.e. without being aware of the existence of other nodes). In addition, due to its widespread usage by the robotics community, there were several publicly available packages that were used in this work, such as RViz[3] for data visualization or rosbag[4] for recording and replaying messages. ROS also provides interoperability between many other useful frameworks such as OpenCV. Finally, being open-source allows ROS to be installed for free and for its code to be reviewed if needed.

3.2 PCL

Point Cloud Library (PCL[5]) is a large-scale and open source project with algorithms for 3D point cloud processing. It is written in C++. These algorithms include registration, filtering, segmentation and matching [RC11]. Figure 3.1 provides an overview of the code libraries available.

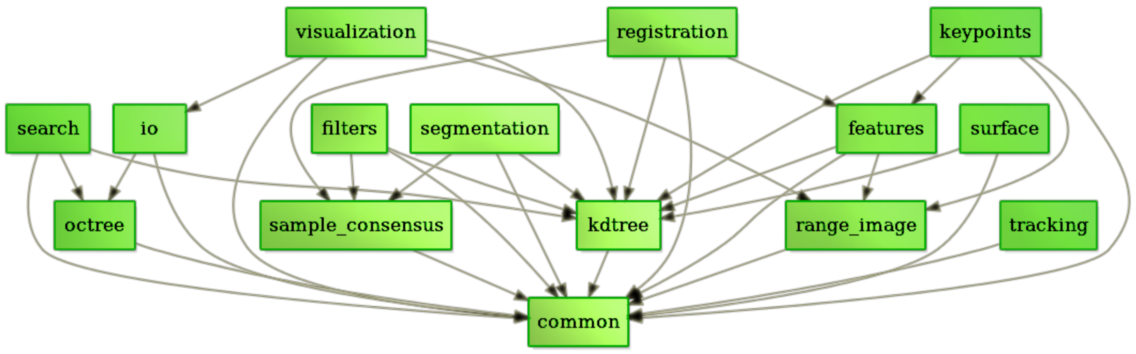


Figure 3.1: Main modules of PCL[6]

PCL includes a visualization library that allows users to view the results of the algorithms.

This framework was chosen for the dissertation due to the efficiency of the algorithms' implementation and for being fully integrated with ROS.

3.3 Physics engines and robot simulators

A physics engine is a software responsible for the simulation of a physical system. Table 3.1 presents some of the most relevant physics engines. This table also specifies whether the engines support the simulation of deformable objects in case the bin picking system is expanded to deal with this kind of objects in the future. Some of the elements in the table are actually more complete

[3] http://wiki.ros.org/rviz
[4] http://wiki.ros.org/rosbag
[5] http://pointclouds.org
[6] http://pointclouds.org/about

software platforms (such as Blender, which is a 3D computer graphics software toolset). For these cases, the table is referring to their underlying physics engines.

Table 3.1: Overview of 3D physics engines

Engine	**Open-source**	**Supports soft-body dynamics**
Blender[7]	Yes	Yes
Bullet[8]	Yes	Yes
CryEngine[9]	No (but most of the code is available on Github[10])	Yes
Dynamic Animation and Robotics Toolkit (DART)[11]	Yes	Yes
Newton Game Dynamics[12]	Yes	No
Open Dynamics Engine (ODE)[13]	Yes	No
PhysX[14]	Yes	Yes
Simbody[15]	Yes	No
Unreal Engine[16]	No	Yes
Vortex Studio[17]	No	No

The use of a simulator in this dissertation is relevant as it allows for algorithms to be tested with more ease than using physical hardware, thus accelerating the development process. Table 3.2 presents the most used robot simulators.

Table 3.2: Overview of robot simulator

Simulator	**Open-source**	**Supported engines**
CoppeliaSim[18]	Yes	ODE, Bullet, Vortex, Newton
Gazebo[19]	Yes	ODE, Bullet, Simbody, DART
Webots[20]	Yes	ODE (forked version)

[7] https://www.blender.org/
[8] https://pybullet.org/wordpress/
[9] https://www.cryengine.com/
[10] https://github.com/CRYTEK/CRYENGINE
[11] https://dartsim.github.io/
[12] http://newtondynamics.com/forum/newton.php
[13] http://www.ode.org/
[14] https://developer.nvidia.com/gameworks-physx-overview
[15] https://simtk.org/projects/simbody/
[16] https://www.unrealengine.com
[17] https://www.cm-labs.com/vortex-studio/
[18] https://www.coppeliarobotics.com
[19] http://gazebosim.org
[20] http://www.cyberbotics.com

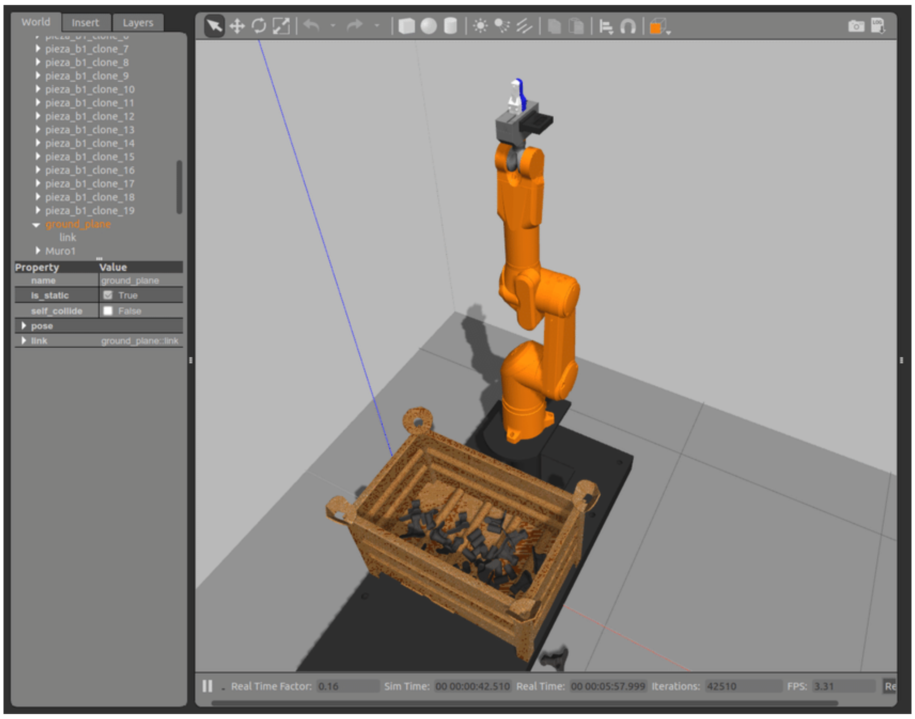
Out of these three simulators, the one with the largest active community is Gazebo. Moreover, this simulator supports four physics engines. Thus, this simulator was chosen for this dissertation's work as there was more support from the community and more flexibility to change the physics engine if needed.

Gazebo can emulate with high accuracy the robots' physical interactions with the environment, as well as the feedback from their sensors (namely, depth cameras). This software provides an Application Programming Interface (API) so that it can be easily integrated with large-scale projects. In addition, the system makes use of Open Graphics Library (OpenGL) to render realistic scenes in real-time [KH04].

The simulator supports plugins[21], which consist of code fragments in C++ that are compiled as shared libraries and ran alongside the simulation. They allow developers to control aspects of the simulation including the physics engine's parameters or the models present in the world. They can also be connected with ROS. Some plugins (namely those that affect physics properties) are run by *gzserver*, the executable for the server, while others are run by *gzclient*, the executable for the simulator's graphical client[22].

One key advantage of using Gazebo is that its interface with ROS is the same as with the robot's hardware. Thus, the same implementation code can be used for both cases, without any modification.

Bin picking can be performed in this simulator, as seen in figure 3.2.



Property	Value
name	ground_plane
is_static	True
self_collide	False
pose	
link	ground_plane::link

Real Time Factor: 0.16 Sim Time: 00 00:00:42.510 Real Time: 00 00:05:57.999 Iterations: 42510 FPS: 3.31

Figure 3.2: Bin picking simulation in Gazebo [BADA18]

[21] http://gazebosim.org/tutorials/?tut=plugins_hello_world
[22] http://gazebosim.org/tutorials?tut=components

Bullet and DART both support soft body dynamics. However, according to some comments[23][24] on Gazebo's forum, this simulation tool is very likely not exposing this functionality of the underlying engines yet.

The ODE physics engine was chosen, since it has the highest amount of configurable features in the *.world* file, out of the four engines supported by Gazebo.

3.4 Summary

This chapter presented important software tools for the dissertation work, including a baseline system for robot software development, a cloud processing library, some physics engines that can simulate the dynamics of tube-shaped objects and robot simulation tools.

[23]http://answers.gazebosim.org/question/5129
[24]http://answers.gazebosim.org/question/17456

Relevant tools

Chapter 4

Proposed solution for modeling of entangled tubes

In order to perform both grasp and motion planning, firstly, the robot must acquire a useful representation of the tubes within the bin. This chapter presents the dissertation's solution to build a model of the tubes using point clouds captured by a 3D scanner.

The work presented in this chapter resulted in an article, entitled *Perception of Entangled Tubes for Automated Bin Picking*, which was accepted and presented in the ROBOT'2019 conference[1] on November 2019 [LCSV19].

Sections 4.3 and 4.4 are heavily based on Sections 3.2 and 3.3 of the conference article, respectively.

4.1 Overview

The modeling system was implemented as a ROS package. The system receives as input a point cloud, which can either be read from a *.ply*[2] file or received as a ROS message with the type *sensor_msgs::PointCloud2*. In the latter case, the name of the topic to which the ROS subscriber listens must also be specified.

The system outputs various information in multiple formats. It publishes markers[3] (of the type *visualization_msgs::MarkerArray*) of the outcomes of each step in multiple topics, so that each of the system's intermediate results can be inspected in RViz. Examples of the visualization of tubes in RViz can be found in Figure 5.2. It also publishes various point clouds (such as the result of the segmentation step) and enables the user to specify a folder where *.ply* files of these clouds are automatically created and stored. Another notable output is a *.csv* file which contains many

[1] https://web.fe.up.pt/~robot2019/
[2] https://web.archive.org/web/20161204152348/http://www.dcs.ed.ac.uk/teaching/cs4/www/graphics/Web/ply.html
[3] http://wiki.ros.org/rviz/DisplayTypes/Marker

numerical results (such as the number of segments or tubes), and, most notably, the execution time of each step.

This modular system is fully configurable via a *.yaml*[4] file, where all of the system's parameters can be freely edited without the need to recompile the whole package. In particular, the pipeline of filters in the first step can be freely customized by adding, removing, editing and changing the order of multiple filters.

This package was developed in C++ as it is the language used for PCL (which contains a wide array of useful functions and classes for point cloud processing) and one of the languages supported by ROS. Moreover, it allows for object-oriented programming and the generation of efficient executable code.

4.2 Key definitions

Before presenting the main steps for the modeling algorithm, it is important to define some keywords that will be used in the rest of this dissertation's work.

Definition 4.2.1. A **real-life tube** corresponds to a tube that is actually present in the bin.

Definition 4.2.2. A **virtual tube** corresponds to a tube that was created by the modeling algorithm. It consists of an ordered list of cylinders and joints.

If the modeling is performed correctly, then there is a one-to-one mapping between the real-life and virtual tubes.

Definition 4.2.3. An **endpoint of a cylinder** is a three-dimensional point at an intersection between the cylinder's central axis and one of its bases. Each cylinder of a virtual tube has two endpoints.

Definition 4.2.4. A **joint** is an entity that connects two consecutive cylinders of a virtual tube by their closest endpoints.

Definition 4.2.5. An **endpoint of a virtual tube** is an endpoint of one of its cylinders that is not connected to any other endpoint by a joint. Each virtual tube has two endpoints.

Definition 4.2.6. The **length of a virtual tube** is the sum of the combined length of its cylinders and joints (whose length is the distance between its endpoints).

4.3 Algorithm description

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

[4] https://yaml.org/

Algorithm 1 Overview of the tube modeling 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:     $tube, inliers, outliers \leftarrow fitCylinder(segment)$
6:     while $size(inliers) >= inliers_{min}$ do
7:         $Tubes \leftarrow Tubes \cup tube$
8:         $tube, inliers, outliers \leftarrow fitCylinder(outliers)$
9: $EndPointPairs \leftarrow \emptyset$
10: for each $pair \in EndPoints(Tubes) \times EndPoints(Tubes)$ do
11:     if $isNearby(pair)$ & $areDifferentTubes(pair)$ then
12:         $EndPointPairs \leftarrow EndPointPairs \cup pair$
13: $EndPointPairs \leftarrow sortByDistance(EndPointPairs)$
14: for each $(endPoint1, endPoint2) \in EndPointPairs$ do
15:     if $areCompatible(endPoint1, endPoint2)$ then
16:         $(tube1, tube2) \leftarrow (tubeOf(endPoint1), tubeOf(endPoint2))$
17:         $Tubes \leftarrow Tubes \cup unite(tube1, tube2) \setminus \{tube1, tube2\}$
```

The following subsections present each step in detail.

4.3.1 Filtering

The first step (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 removal filter erases 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 [Rus10]. 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, as explained in section 2.1.2. 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 removal 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 removal 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 [RBM+07]. 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 (4.1), where *mult* is a multiplier that helps to regulate how restrictive the filter is.

$$\overline{d} > \mu + mult * \sigma \tag{4.1}$$

4.3.2 Segmentation

The second step (line 2 in algorithm 1) aims to divide the filtered point cloud into disjoint piecewise smooth regions.

Definition 4.3.1. A **segment** is a subset of the filtered point cloud where the surface normal varies continuously. It corresponds to one of the clusters given by the segmentation step.

Each segment 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/segment 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 [RvdHV06]. 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.

4.3.3 Cylinder Fitting

The third step (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 below a minimum threshold. 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, 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 in the tube joining step.

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 is set to be extended by a length ε_{Δ_l} at both bases and to have a radius ε_r. The overlapping degrades significantly the performance of the tube joining step since the algorithm may be unable to recombine both tubes into one.

4.3.4 Tube Joining

The fourth and final step (lines 9-17 in algorithm 1) consists in combining the cylinders created in the previous step to form complete tubes.

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 step.

This step 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 4.1.

The endpoint pairs with an euclidean or angular distance above predefined thresholds are discarded since they are unlikely to belong to the same tube. The remaining endpoint pairs are ordered using a distance metric that combines the euclidean and angular distance, as presented in equation (4.2). A coefficient $c_{\Delta\theta}$ 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). In the exceptional case where the euclidean distance is below the radius of a tube, it is very likely that the cylinders belong

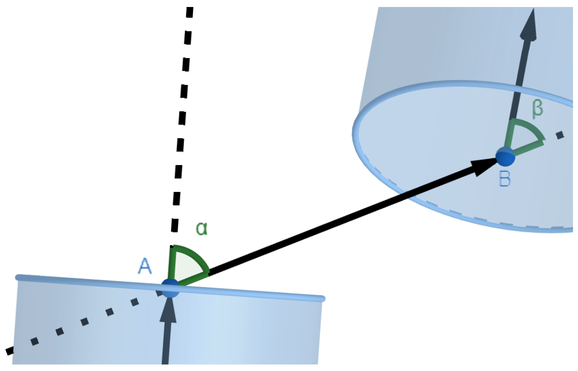


Figure 4.1: Visual representation of the angular distance, sum of the angles α and β

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).

$$dist(A,B) = euclideanDist(A,B) + c_{\Delta\theta} * angularDist(A,B) \tag{4.2}$$

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. [QZN14], which also joined 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 step, 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 a maximum predefined value. 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 allows the distinction between gaps that are due to the close proximity between two tube's ends and those that are due to an occluding tube. The constraint was implemented by projecting both endpoints onto a 2D range image of the point cloud that resulted from the filtering step 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 work that presents these last two constraints for the problem of tube reconstruction.

4.4 Experiments

4.4.1 Hardware

A Zivid One Plus L scanner[5] (Figure 4.2b) 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.

4.4.2 Setup

The 3D scanner 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 4.2a). 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 4.2c). The bin's dimensions were 55 cm (length) x 37 cm (width) x 20 cm (height).

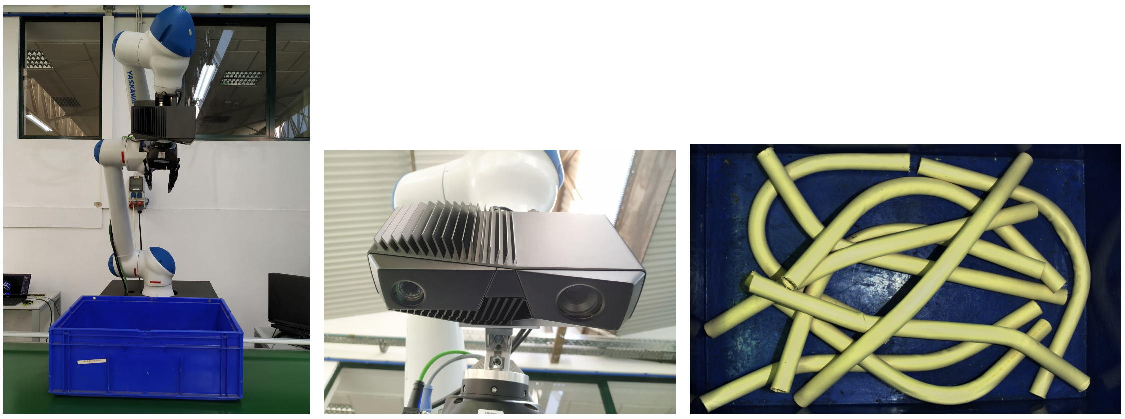

(a) Overview (b) Zivid sensor (c) Bin with bent PVC tubes

Figure 4.2: Experimental setup

4.4.3 Parameter values

Table 4.1 presents the PCL classes that were used in each of the modeling algorithm's steps and sub-steps (for the filtering step).

[5]https://www.zivid.com/zivid-one-plus-large-3d-camera

Table 4.1: PCL classes used in each step and sub-step

Step/sub-step	**PCL class**
Pass-through filters	*pcl::PassThrough*[6]
Random samplers	*pcl::RandomSample*[7]
Radius outlier removal filters	*pcl::RadiusOutlierRemoval*[8]
Normal estimation	*pcl::NormalEstimationOMP*[9]
Plane filter	*pcl::SACSegmentationFromNormals*[10]
Statistical outlier removal filter	*pcl::StatisticalOutlierRemoval*[11]
Segmentation	*pcl::RegionGrowing*[12]
Cylinder fitting	*pcl::SACSegmentationFromNormals*

Tables 4.2 and 4.3 indicate the parameter values for each filter of the first step and for the three following steps, respectively. Parameters marked with an asterisk (*) are not detailed in this chapter, so it is advised for the reader to consult the appropriate PCL 1.9.1 API pages for the classes in Table 4.1. If a certain parameter is not specified, then the PCL class' default value for that attribute was used.

In Table 4.3, for the tube joining step, the coefficient for the angular distance $c_{\Delta\theta}$ was set to 0.001 since the maximum acceptable distance for both metrics (10 cm and 90°) were assumed to be 'equivalent' (with $\frac{0.1}{90} \approx 0.001$). The maximum tube length for the 'length constraint' was set to 60 cm, to add a margin of error of 10 cm to the 50 cm tube length.

4.4.4 Results

A dataset[13] 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.

Tables 4.4 and 4.5 present the average value of different performance metrics with respect to the number of tubes.

The results in table 4.4 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 step 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

[7]http://docs.pointclouds.org/1.9.1/classpcl_1_1_pass_through_3_01pcl_1_1_p_c_l_point_cloud2_01_4.html

[8]http://docs.pointclouds.org/1.9.1/classpcl_1_1_random_sample_3_01pcl_1_1_p_c_l_point_cloud2_01_4.html

[9]http://docs.pointclouds.org/1.9.1/classpcl_1_1_radius_outlier_removal_3_01pcl_1_1_p_c_l_point_cloud2_01_4.html

[10]http://docs.pointclouds.org/1.9.1/classpcl_1_1_normal_estimation_o_m_p.html

[11]http://docs.pointclouds.org/1.9.1/classpcl_1_1_s_a_c_segmentation_from_normals.html

[12]http://docs.pointclouds.org/1.9.1/classpcl_1_1_statistical_outlier_removal_3_01pcl_1_1_p_c_l_point_cloud2_01_4.html

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

Table 4.2: Parameter values used in the filtering step

Filter	Parameter	Value
First pass-through filter	Minimum z	0.1 m
	Maximum z	1.0 m
First random sampler	Number of points	100000
First radius outlier removal filter	Radius	0.7 cm
	Minimum number of points	10
Normal estimation	Radius	0.5 cm
Plane filter	Maximum inlier distance	1.5 cm
	Probability*	0.2
	Normal distance weight*	0.07
	Axis*	(0,0,1)
	Maximum number of iterations	1000
	Angle epsilon threshold*	0.14 rad
Second random sampler	Number of points	25000
Second radius outlier removal filter	Radius	0.7 cm
	Minimum number of points	15
Statistical outlier removal filter	Number of neighbors	100
	mult	3.0

Table 4.3: Parameter values used in the segmentation, cylinder fitting and tube joining steps

Step	Parameter	Value
Segmentation	Minimum cluster size	100
	Number of nearest neighbors	77
	Curvature threshold*	0.2
	Residual threshold*	10.0
	Theta threshold*	5.0
Cylinder fitting	Minimum number of points to form a cylinder	100
	Minimum cylinder radius	0.7 cm
	Maximum cylinder radius	1.5 cm
	Maximum inlier distance*	2.0 cm
	Normal distance weight*	0.07
	Maximum number of iterations	1000
	ε_{Δ_l}	1.0 cm
	ε_r	2.0 cm
Tube Joining	Maximum endpoint euclidean distance	10.0 cm
	Maximum endpoint angular distance	90°
	$c_{\Delta\theta}$	0.001
	Maximum tube length	60.0 cm

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 step 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.

Table 4.4: Average execution time for the algorithm's four steps

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

To evaluate the performance of the segmentation step, two human 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 4.5 is the relative error between the number of clusters produced by the segmentation step 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.

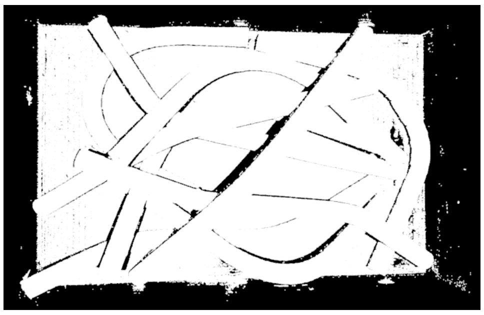
Two metrics used to assess the performance of the tube joining step in a given test case are the average and standard deviation for the lengths of the tubes. Ideally, the average length should be 50 cm and the standard deviation should be minimal. According to table 4.5, the tube joining step appears 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.

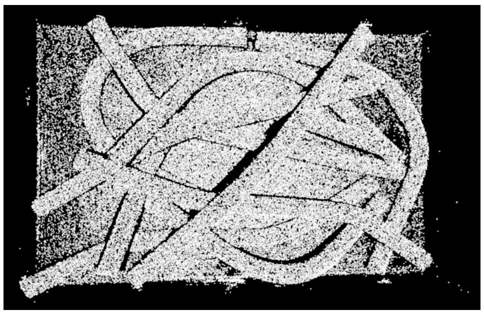
The tube length metrics are not sufficient to assess with great confidence the performance of the perception algorithm since the lengths are estimates and do not account for incorrect matches between tube sections of similar length. Ergo, along with the number of tubes produced by the solution (*Number of joint tubes*), the annotators counted how many tubes were correctly and partially correctly modeled. A real-life tube 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 4.5, 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.

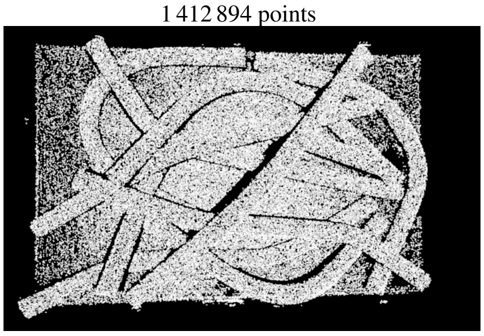
Table 4.5: Accuracy metrics for the segmentation, fitting and joining steps

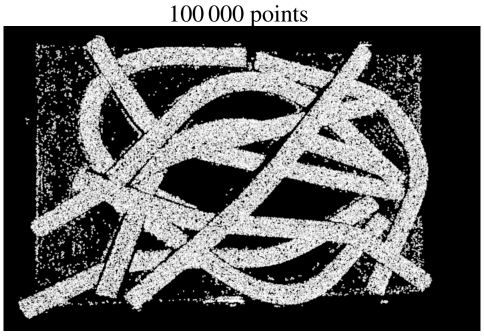
No. of tubes	**Segmentation error**	**Avg. length (m)**	**Std. deviation length (m)**	**No. of joint tubes**	**No. of correct tubes**	**No. of partially correct tubes**
1	2.00	0.439	0.00105	1.2	0.8	0.2
2	0.83	0.439	0.08911	2.4	2	0
3	0.20	0.502	0.00824	3	3	0
4	0.00	0.468	0.03406	4.4	3.6	0.4
5	0.04	0.480	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

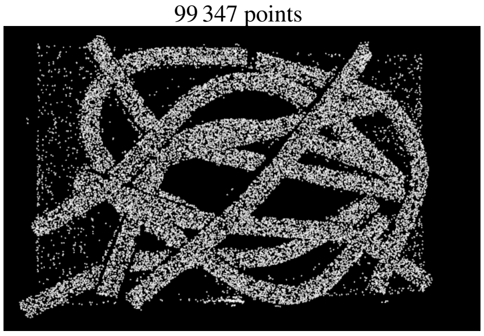
Figure 4.3 illustrates the full process for one of the test cases with ten tubes (test case *10_bin_picking2* from Figure 4.2c). The number of remaining points after each filter is also shown. In this case, the algorithm performed quite well since nine of the tubes are correctly modeled and the remaining one is partially correct (the two tube models corresponding to the partially correct one are marked with a red ellipse in Figure 4.3j). It is interesting to notice that in Figure 4.3h there was an occurrence of under-segmentation (marked with a red ellipse) that the cylinder fitting step was able to recover from.

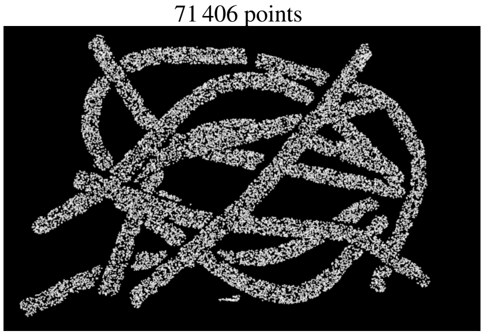

(a) Raw point cloud
1 412 894 points

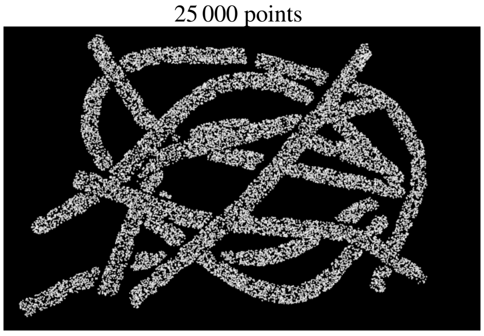

(b) After the first random sampler
100 000 points

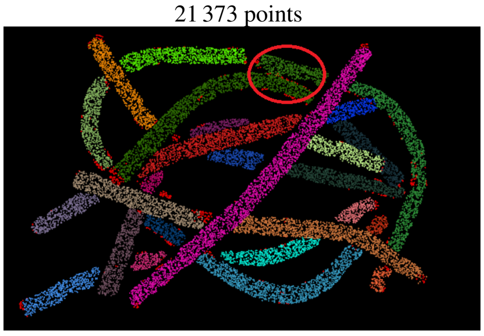

(c) After the first radius outlier removal filter
99 347 points

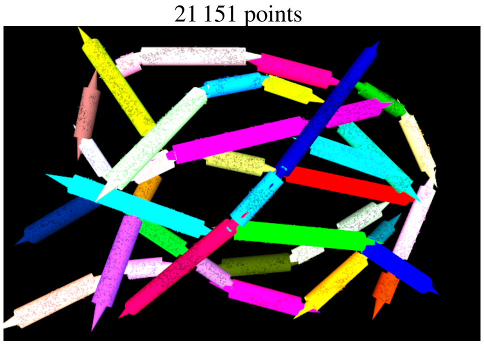

(d) After the plane removal filter
71 406 points


(e) After the second random sampler
25 000 points


(f) After the second radius outlier removal filter
21 373 points


(g) After the statistical outlier removal filter
21 151 points


(h) After segmentation


(i) After fitting cylinders

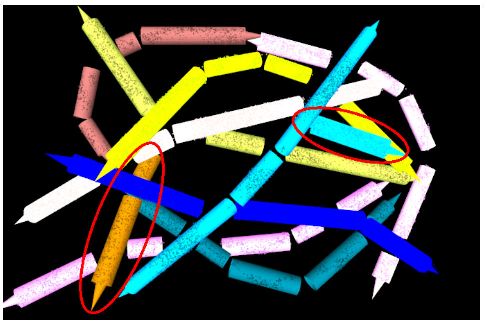

(j) After joining tubes

Figure 4.3: Results of the algorithm's steps for the case shown in Figure 4.2c

4.5 Automated tube modeling evaluation module

As explained in section 4.4.4, most of the metrics used to evaluate the perception algorithm were computed manually due to the lack of a ground truth: there was no automated way of telling to which tube each element of the point cloud belonged. Having an automatic way of evaluating the outcome of the perception algorithm would be highly advantageous since a much larger amount of test cases could be evaluated, making the results more statistically relevant.

During this dissertation's work, most of a system that is able to automatically measure the perception algorithm was developed. Section 4.5.1 details all of the work that was done, while Section 4.5.2 proposes the next steps.

4.5.1 Description

A plugin for Gazebo was created that randomly generates a set of tubes and then runs the modeling algorithm using a point cloud of the scene. The simulated world contains a 3D scanner that is positioned looking downwards towards the bin. The *Openni Kinect* plugin[14] was used in the simulation so that the point clouds captured by the sensor were published in a ROS topic.

The plugin is a Gazebo World Plugin[15,16], as these give full control to the developer of when objects should be created and removed, as well as access to their properties. Similarly to most of this dissertation's work, the plugin was developed in C++.

In the simulated environment, a different color is given to each tube. This color is used as a ground truth system to indicate to which tube each point belongs. To ensure there was a one-to-one mapping between the tubes and the colors, three steps were accomplished:

- To ensure that a tube's color is constant throughout its surface, all of the simulated world's light sources were removed, with the exception of an ambient light.
- The anti-aliasing feature of Gazebo's rendering engine was disabled, as this caused certain points of a given object to have slightly different colors when in close proximity to other objects. Since disabling this feature was not available in Gazebo's API, a fork[17] of the Gazebo open-source project was created, and, after careful examination of the code, a single line[18] in the *gazebo/rendering/Camera.cc* file was changed to produce the desired effect. As a consequence, to use this evaluation module, this modified version of Gazebo must be used.
- When acquiring a point cloud of the scene, the physics engine was disabled to ensure that all the tubes are immobile. Due to an issue with the *Openni Kinect* plugin where the depth and color information are not synchronized, if any of the tubes is moving, there are great odds that the colors of some points are incorrect in the point cloud.

[14] `http://gazebosim.org/tutorials/?tut=ros_depth_camera`
[15] `http://osrf-distributions.s3.amazonaws.com/gazebo/api/9.0.0/classgazebo_1_1WorldPlugin.html`
[16] `http://gazebosim.org/tutorials?tut=plugins_world`
[17] `https://bitbucket.org/GoncaloLeao/gazebo/src/default`
[18] `https://bitbucket.org/GoncaloLeao/gazebo/commits/5182739f317c46184652e9cf62480bdf1c33d164`

Figure 4.4 depicts an example of a simulated scene in Gazebo with a bin filled with 10 randomly generated tubes. Due to the absence of shadows and light sources, it is difficult to perceive depth in the image. The scanner is located in the top-left corner of this figure and is colored in a slightly darker gray tone than the bin. Figure 4.5 shows the point cloud representation of the scene. There are 11 different colors: one for each tube and one for the bin.

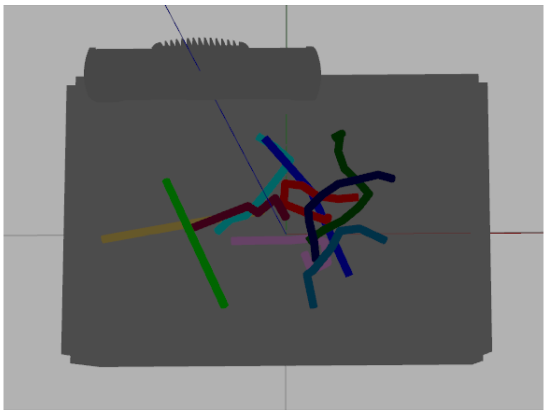


Figure 4.4: Example of a randomly generated set of tubes in Gazebo

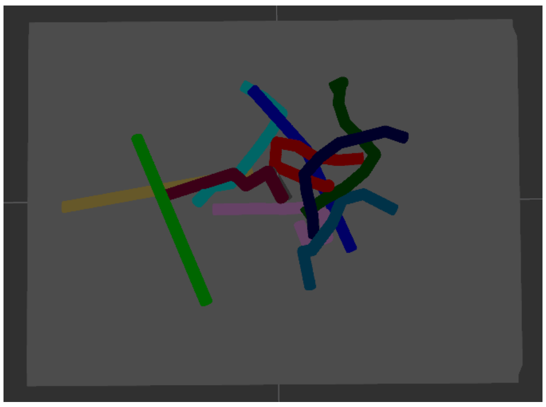


Figure 4.5: Point cloud of the scene in figure 4.4 in RViz

A simulated tube is composed of a linked list of cylinders which are connected by their endpoints. Using multiple cylinders allows for the emulation of curved tubes. In this module, the cylinders have a constant radius. To ensure the tube's surface is smooth, a sphere with the same radius as the cylinders is created for each endpoint that connects two cylinders.

The main steps of the evaluation procedure of a single test case are as follows:

- A set of tubes is randomly generated using Algorithm 2.
- Each tube is spawned in the world at a certain height from the bin's bottom and at a position that enables it to fall inside the bin. After each spawn, the plugin waits for all of the tubes present in the world to stop moving by checking that the linear and angular velocity of their center of mass is below a certain threshold.
- A point cloud of the scene is captured by the scanner and the ROS node in which Gazebo is running sends a request to the node running the tube modeling node with the point cloud. The Gazebo then awaits a response from the server.
- The response from the tube modeling node is processed. Due to the time restrictions of this dissertation's work, this last step was left for future work.

Algorithm 2 Random tube generation algorithm

```
Input:
    $N$ - Number of tubes to generate
    $R$ - Cylinder radius
    $L$ - Cylinder length
    $CL_{min}$ - Minimum cylinder length
    $C_{min}, C_{max}$ - Minimum/maximum amount of cylinders
    $\theta_{min}, \theta_{max}$ - Minimum/maximum angle between two consecutive cylinders
Output:
    $Tubes$ - Set of generated tubes
 1: for $n \in \{0, \dots, N-1\}$ do
 2:     $tube \leftarrow \emptyset$
 3:     $c \leftarrow randomIntegerInRange(C_{min}, C_{max})$
 4:     $lengths \leftarrow []$
 5:     $remainingLength \leftarrow L$
 6:     for $i \in \{0, \dots, c-2\}$ do
 7:         $length \leftarrow randomIntegerInRange(CL_{min}, remainingLength - (c-1-i) * CL_{min})$
 8:         $lengths.push(length)$
 9:         $remainingLength \leftarrow remainingLength - length$
10:     $lengths.push(remainingLength)$
11:     $previousEndpoint \leftarrow (0,0,0)$
12:     $endpoints \leftarrow [previousEndpoint]$
13:     for $i \in \{0, \dots, c-1\}$ do
14:         $\phi \leftarrow randomFloatInRange(0, 2\pi)$
15:         if $i = 0$ then
16:             $\theta \leftarrow randomFloatInRange(0, \pi)$
17:             $unitDirection \leftarrow (sin(\theta) * cos(\phi), sin(\theta) * sin(\phi), cos(\theta))$
18:         else
19:             $\theta \leftarrow randomFloatInRange(\theta_{min}, \theta_{max})$
20:             $unitDirection \leftarrow (sin(\theta) * cos(\phi), sin(\theta) * sin(\phi), cos(\theta))$
21:             $unitDirection \leftarrow rotateVectorAroundAxisWithAngle(unitDirection, (0,0,1) \times previousDirection, (0,0,1) \cdot previousDirection)$
22:         $nextEndpoint \leftarrow previousEndpoint + lengths[i] * unitDirection$
23:         $previousDirection \leftarrow nextEndpoint - previousEndpoint$
24:         $previousEndpoint \leftarrow nextEndpoint$
25:         $endpoints.push(nextEndpoint)$
26:     $endpoints \leftarrow applyRandomTranslation(endpoints)$
27:     for $i \in \{0, \dots, c-1\}$ do
28:         $tube \leftarrow tube \cup createCylinderWithRadiusAndEndpoints(R, endpoints[i], endpoints[i+1])$
29:         if $i > 0$ then
30:             $tube \leftarrow tube \cup createCylinderWithRadiusAndCenter(R, endpoints[i])$
31:     $Tubes \leftarrow Tubes \cup tube$
```

The ODE physics engine is used to compute the pose of the tubes at each time step. Making the tubes drop into the bin rather than simply spawning them without gravity provides much more realism to the tube configurations.

The bin's dimensions as well as all of the parameters in Algorithm 2 are can be freely modified by the user by editing a *.yaml* file. In addition, the sensor's pose can be altered by editing the simulation's *.world*[19] file, that describes several parameters of the simulated scene.

4.5.2 Next steps

As explained in the previous section, the algorithm to process the response from the tube modeling node was left for future work. This response contains two point clouds: the cloud resulting from the filtering step and a cloud from the joining step, which only contains points that were fitted to any of the cylinders. The first cloud has the same colors as those of the input cloud, while the second one colors each point according to which virtual tube it belongs.

To evaluate the accuracy of the modeling technique, first of all, since the second cloud is a subset of the first one (as some points were not used by any of the calls to RANSAC to form a cylinder), the points of the first cloud that are not in the second one should be labeled with a color from a virtual tube. To label the points present in the joints, a simple algorithm could be used where all the unlabeled points inside a cylinder representing one of the joints of a tube with color c are labeled as having color c. The remaining unlabeled points could be set to have a *null color* $c_\emptyset$, which is different from those of the virtual tubes. The result of this phase is a point cloud with the same points as the first cloud but with two different colors: a *ground truth color* (originally from the first cloud) and a *virtual tube color* (obtained from the second cloud or the aforementioned labeling technique).

Secondly, a mapping must be established between the ground truth colors and the virtual tube colors. This corresponds to an unbalanced assignment problem in a weighted bipartite graph, where an edge connecting a ground truth color C to a virtual tube color c has a weight equal to the number of points which do not have both of these colors. The problem could then be solved by using one algorithm that solves the minimum cost flow problem or the Hungarian method [RT12], for instance.

Afterwards, a confusion matrix (similar to those used for classification in machine learning) can be constructed to properly evaluate the tube modeling algorithm. In Table 4.6, let C_i be a ground truth color and c_i its matched virtual tube color. The number of points with colors c_i and C_j is given by $n_{i,j}$. In this matrix, various performance measures could be computed, such as the average accuracy or the error rate [SL09].

After developing the whole evaluation system, a scientific article can be published where several aspects of the modeling algorithm are explored, including:

- Finding an automated way of adjusting all of the parameters presented in section 4.4.3 in function of the tube's properties, namely their radius and length.

[19] `http://sdformat.org/spec?ver=1.7&elem=world`

Table 4.6: Confusion matrix template for the evaluation of the tube modeling algorithm

Virtual tube color	Ground truth color C_1	...	C_j	...	C_n
c_1	$n_{1,1}$	...	$n_{1,j}$	...	$n_{1,n}$
...	...	...	...	...	...
c_i	$n_{i,1}$	...	$n_{i,j}$	...	$n_{i,n}$
...	...	...	...	...	...
c_m	$n_{m,1}$	...	$n_{m,j}$	...	$n_{m,n}$
$c_\emptyset$	$n_{\emptyset,1}$	...	$n_{\emptyset,j}$	...	$n_{\emptyset,n}$

- Finding the ranges for the tube radius and length, outside of which the modeling algorithm has a poor accuracy, no matter the values chosen for its parameters.
- Finding an optimal size for the first random filter that balances the whole algorithm's accuracy and execution time.

To conclude this section, it is interesting to note that the random tube generation algorithm could also be used to automatically generate labeled examples to train a machine learning-based classifier to determine how many tubes are present in a bin and correctly label each element of a point cloud.

4.6 Main challenges

This section describes some of the most interesting/complex challenges faced during the work required to produce this chapter's results.

4.6.1 3D scanner accuracy

During initial work to develop the modeling algorithm, an Intel RealSense D415[20] sensor was used. Tests of the region-growing segmentation using filtered point cloud captured by this sensor yielded poor results and they were highly sensitive to the algorithms' parameters, due to the sensor's low accuracy. Both of these problems were easily solved when switching to commercial high-end sensors. These experiments clearly illustrated why a sensor's accuracy and resolution are critical for vision-based algorithms and why the price between sensor models varies so much.

4.6.2 Cylinder overlapping

During the model fitting step, cylinders created from the same segment had a high tendency to overlap with each other, which decreased significantly the solution's accuracy by causing issues during the tube joining step where cylinders belonging to the same real-life tube were not combined due to the maximum angular distance being exceeded. This was solved by removing the

[20] https://www.intelrealsense.com/depth-camera-d415

outlier points in a *safety zone* around each cylinder, after each call to RANSAC using the ε_{Δ_l} and ε_r parameters presented in subsection 4.3.3.

4.7 Summary

This chapter presented this dissertation's work solution for modeling sets of freely curved tubes with occlusions and entanglement using as input a point cloud captured by a 3D scanner. This modeling system is a key element of the bin picking solution presented in this work.

This chapter began by introducing some key definitions to formalize the problem. Then, the modeling algorithm was presented in detail. It is composed of four steps: filtering, segmentation, cylinder fitting and tube joining. It was then described how the system was evaluated both in terms of execution time and how accurately the tube models described their actual pose. This chapter ended with a detailed proposal of a solution that could be used to evaluate this modeling system automatically.

Chapter 5

Proposed solution for bin picking of entangled tubes

In the previous chapter, a method to obtain a representation of the tubes from a point cloud was described. This chapter builds upon this method by presenting the dissertation's solution to perform the bin picking task with tube-shaped items.

This chapter's work resulted in an article, entitled *Detecting and Solving Tube Entanglement in Bin Picking Operations*, which was published in the Applied Sciences journal[1] on March 2019 [LCSV20].

Sections 5.2 and 5.3 are heavily based on Sections 3 and 4 of the journal article, respectively.

5.1 Overview

The ROS package presented in Section 4.1 was expanded to also perform grasp and motion planning. This package will be referred to as the *Tube Pick Planner*. After planning, it broadcasts the coordinates frames[2] that define the gripper's pose when grasping the tube and also when placing it in the destination area. In addition, the waypoints of the trajectory the gripper should follow after grasping the tube are added to the ROS parameter server[3].

A plugin (of the *WorldPlugin* type) for Gazebo was developed to evaluate the different trajectories to remove the grasped tube from the bin. It creates a ROS *action server*[4] that awaits for the node running the code on *Tube Pick Planner* to send a request for trajectory simulation. This request is a custom ROS message that contains the information about the tubes' shape and trajectories (with the identifier of the tube to be moved and the waypoints). The server then replies

[1]https://doi.org/10.3390/app10072264
[2]http://wiki.ros.org/tf2/
[3]http://wiki.ros.org/Parameter%20Server
[4]http://wiki.ros.org/actionlib

with the cost associated with each trajectory (using another custom message type), as detailed in Section 5.2.3.2.

Another ROS package, the *Force/Torque Analysis* package, was developed to process the force/torque data, which is received as a standard *geometry_msgs::WrenchStamped*[5] message. This package is responsible for determining how many tubes were picked by the sensor and publishing the coordinate frames that define a pose for the gripper before and after performing a rotation that attempts to solve a case of entanglement, as explained in Sections 5.2.5 and 5.2.6

Finally, a package was created to control the whole bin picking operation. It builds upon the *task_manager* package which was developed and kindly provided by the research team of the Industry and Innovation Lab (iiLab) of INESC-TEC. The task manager is able to read *.scxml*[6] files that represent the robot's behavior as a state machine and control the robot accordingly. Each state, with the exception of the initial and final states, consists of an action that the robotic system must accomplish. Most of the elementary *skills* were also provided by the team at iiLab. In this dissertation's work two new skills were created to accomplish the goals of the *Tube Pick Planner* and *Force/Torque Analysis* packages. The list of skills used in this work is summarized in table 5.1.

Table 5.1: Robot skills used for the bin picking operation

Skill name	**Description**
ForceTorqueAnalysisSkill	Used to determine how many tubes were picked and the paramters of a rotation for disentanglement if needed.
MoveArmSkill	Used to make the robot arm perform movements that are specified in configurable coordinate frames. It is able to read waypoints published in the parameter server to execute a set of relative movements.
PhotoneoSkill	Used to make the Photoneo sensor scan the scene.
RobotiqGripperSkill	Used to make the Robotiq gripper open or close.
TubePickPlannerSkill	Used to determine the gripper poses and trajectories to perform the pick-and-place operation.
WaitSkill	Used to make the robot wait for a certain amount of time. This is used to ensure that the 3D scanner or the gripper is immobile when scanning bin or grasping a tube, respectively.

When executing the bin picking operation, each of the different skills specified in the *.scxml* file interacts with the task manager using action servers. When a certain state is reached, the task manager sends a request to the appropriate skill and waits for the server to indicate whether the action was successful or not. This outcome determines the next action in the state machine.

During this dissertation's work, several coordinate frames were manually defined. First of all, a frame was defined for the bin's center with its axes perpendicular to the bin's bottom and walls. This frame was defined with respect to the robot arm's base. Frames were also created for the scanner's pose when scanning the bin and for the area where the tubes should be placed (the

[5] http://docs.ros.org/melodic/api/geometry_msgs/html/msg/WrenchStamped.html
[6] https://www.w3.org/TR/scxml/

destination area). Special care was required when defining these frames so that neither the robot arm nor any of its attached tools collided with the environment when transitioning between each pose.

5.2 Algorithm description

Figure 5.1 provides an overview of the different phases of the algorithm for picking entangled tubes. The ultimate goal of this solution is to allow a robotic arm to carry, one-by-one, each tube from the bin to the destination area, until the bin is empty.

The following subsections detail each phase of the algorithm.

5.2.1 Tube modeling

The algorithm used for obtaining a model of the tubes is very similar to the one described in Section 4.3. The only difference is on the filtering step, where rather than applying various computationally expensive and parameter-heavy filters to remove points that do not belong to the tubes, a crop-box filter (using the *pcl::CropBox*[7] class) is used. This filter is equivalent to six pass-through filters (two for each dimension) that remove points that are outside predefined bounds in the x, y and z axes. It is possible to use this filter since the point cloud is on the bin's coordinate frame.

During this phase, only three operations are performed: the surface normals and curvature are estimated, the box filter is applied, and finally a random filter downsamples cloud.

Figure 5.2 illustrates the tube modeling phase for the two sets of tubes presented in Figures 5.14a and 5.14b, which were used in this chapter's experiences. In the images for the tube joining step, the cylinders of each tube have different colors and the joints are represented as colored lines connecting the cylinders. The green and red lines represent occluded and non-occluded joints, respectively. Each cone represents an endpoint.

[7] `http://docs.pointclouds.org/1.9.1/classpcl_1_1_crop_box_3_01pcl_1_1_p_c_l_point_cloud2_01_4.html`

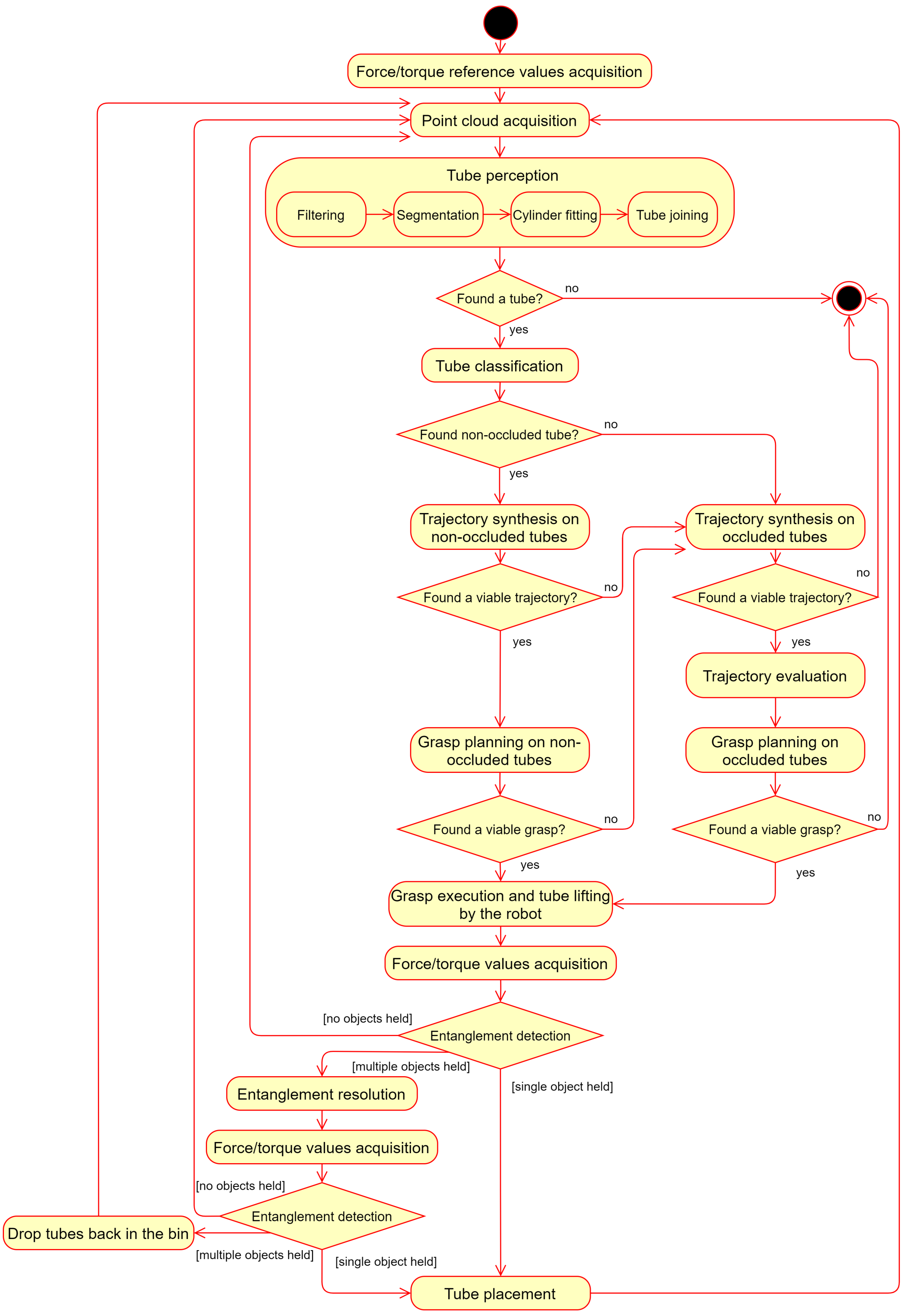


Figure 5.1: UML activity diagram of the bin picking algorithm

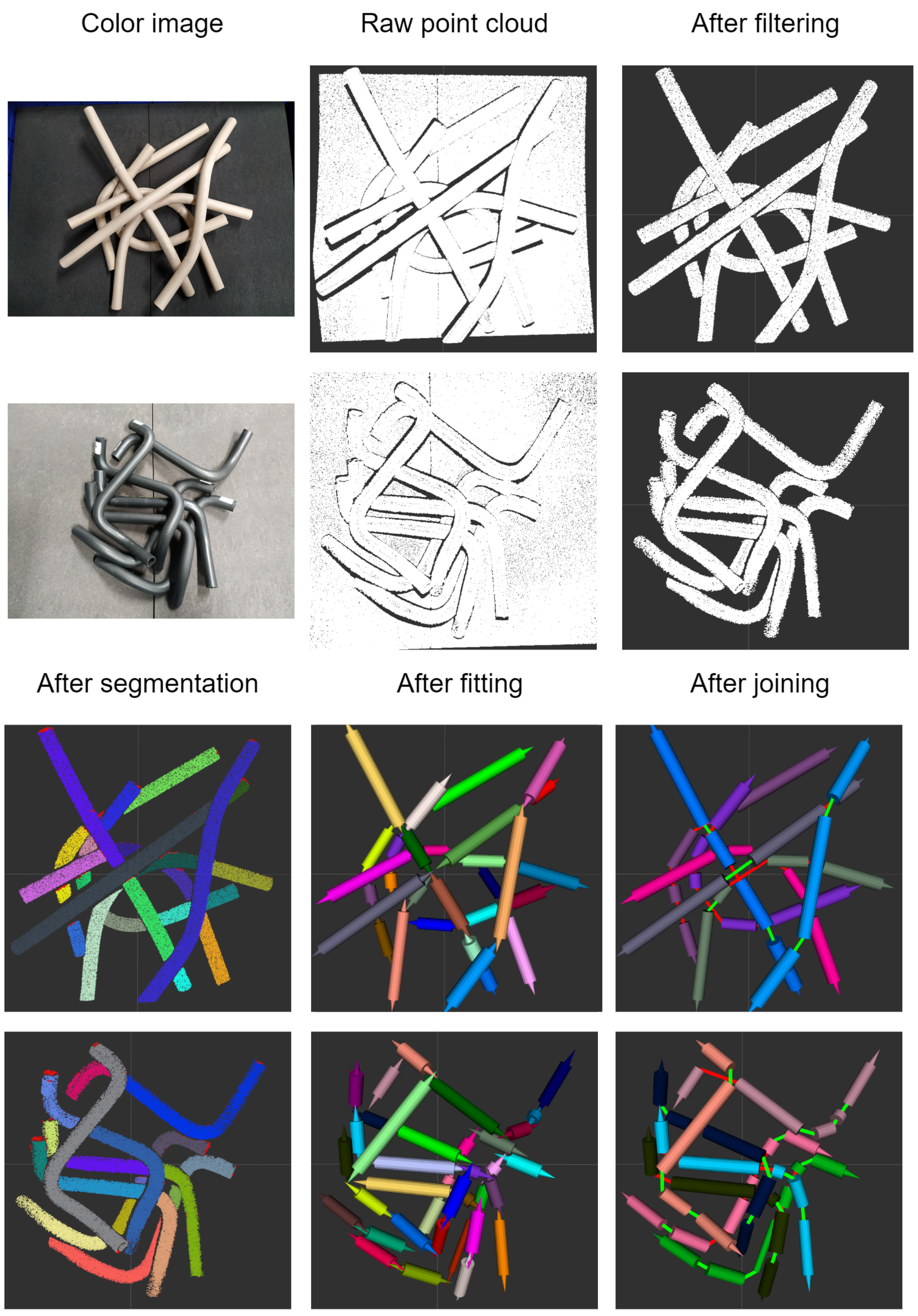


Figure 5.2: Two examples of each step of the tube modeling phase

5.2.2 Tube classification

This phase aims to select the tubes that have the highest potential of being picked without facing any issues of entanglement, where other tubes are also brought up by the robot arm when it performs the lifting motion.

Firstly, if a value is provided for the tubes' length, then all the virtual tubes whose length is below a certain minimum threshold (which subtracts from the expected length a tolerance value) are immediately discarded since they correspond to incomplete models of real-life tubes. The value of a virtual tube's length is typically smaller than the corresponding real-life tube's length (due to the approximation of the curve as a set of segments). This was taken into account when defining the minimum acceptability threshold.

Secondly, the number of occlusions for each tube is counted from the sensor's point of view. Since the sensor's line of sight is vertical, this provides a computationally efficient manner of detecting overlaps between tubes. Occlusions can only occur in joints since cylinders correspond to approximation of continuous, visible portions of a tube.

Definition 5.2.1. An **occluded joint** is a joint for which the segments used to create both of its connected cylinders are different.

At the end of this phase, three disjoint sets of tubes are created: the non-occluded tubes, the weakly-occluded tubes and the strongly-occluded tubes. In this work, only the non-occluded and weakly-occluded tubes are considered as candidates to be picked.

Definition 5.2.2. A **non-occluded tube** is a virtual tube with no occluded joints.

Definition 5.2.3. A **weakly-occluded tube** is a virtual tube with a single occluded joint. The linked list of cylinders in an occluded tube is partitioned into two **sections** by the occluded joint.

Definition 5.2.4. A **strongly-occluded tube** is a virtual tube with two or more occluded joints.

5.2.3 Motion planning

The goal of this phase is to generate and evaluate a set of trajectories for the robotic gripper to follow after grasping a tube. A cost is computed for each trajectory which reflects an estimate of how likely executing it will lead to a case of entanglement. This cost is computed via simulation.

Definition 5.2.5. A **trajectory** of a virtual tube is defined by an ordered list of poses, known as **waypoints**, that represent translations and rotations of the tube relative to its initial position in the bin. The first waypoint of any trajectory is always a relative pose with no translation or rotation.

In this work, the waypoints are simplified to only contain a relative position, since all trajectories are a set of translations.

5.2.3.1 Trajectory synthesis

Two sorts of trajectories are considered: an *upwards trajectory* (where only the z component of the gripper's position is affected) and an *escape trajectory*. This second trajectory begins by performing a small upwards motion, followed by a movement along a line on the x and y axes to try to move the tube so that it is no longer occluded, and ends with another larger upwards trajectory.

If the set of non-occluded tubes is not empty, then an upwards trajectory is created for each of them with a cost of 0, and the algorithm moves on to the grasp planning phase directly. Therefore, the simulator is not used, as it is assumed that since these tubes do not have any other object on top of them, as long as a suitable grasping point is chosen, lifting them should not lead to a situation of entanglement. Avoiding using the simulator for these simpler cases has the advantage of considerably shortening the algorithm's processing time.

If no suitable grasps are found for any of the non-occluded tubes, then the weakly-occluded tubes are considered. For each of these tubes, three trajectories are generated: the upwards trajectory and two escape trajectories that try to free the tube from the occluding object by performing a lateral movement in both directions of the escape line, defined below.

Definition 5.2.6. The **escape line** of an occluded tube is the orthogonal projection of the line that passes through both of the occluded joint's endpoints on the xOy plane.

The distance d_{escape} along the escape line that the gripper moves in one of the directions is given by Equation 5.1, which is a sum of a variable term, d_{var}, and a constant term, d_{const}.

$$d_{escape} = d_{var} + d_{const} \tag{5.1}$$

Let $occl$ be one of the two endpoints of the occluded joint, $EP(occl)$ the set of cylinder endpoints on the same section as $occl$ and $proj(P)$ the projection of point P on the escape line. The variable term (Equation 5.2) is the distance between the occlusion and the farthest endpoint projected on the escape line. It varies according to which direction the tube is moved.

$$d_{var} = max(\{d \mid d = euclidean\_distance(proj(occl), proj(ep)), ep \in EP(occl)\}) \tag{5.2}$$

The constant term is a predefined distance that significantly increases the chances of solving the entanglement by accounting for possible inaccuracies during the tube modeling. It should be at least as large as three times the tube's radius. This is due to the possibility of the length of an occlusion being as large as the occluding tube's diameter. In addition, the farthest endpoint may not be the farthest point of the section to the occlusion, which occurs namely when the axis of the cylinder containing the farthest endpoint is not parallel to the escape line.

Figure 5.3 provides a visual aid for the escape line and distances.

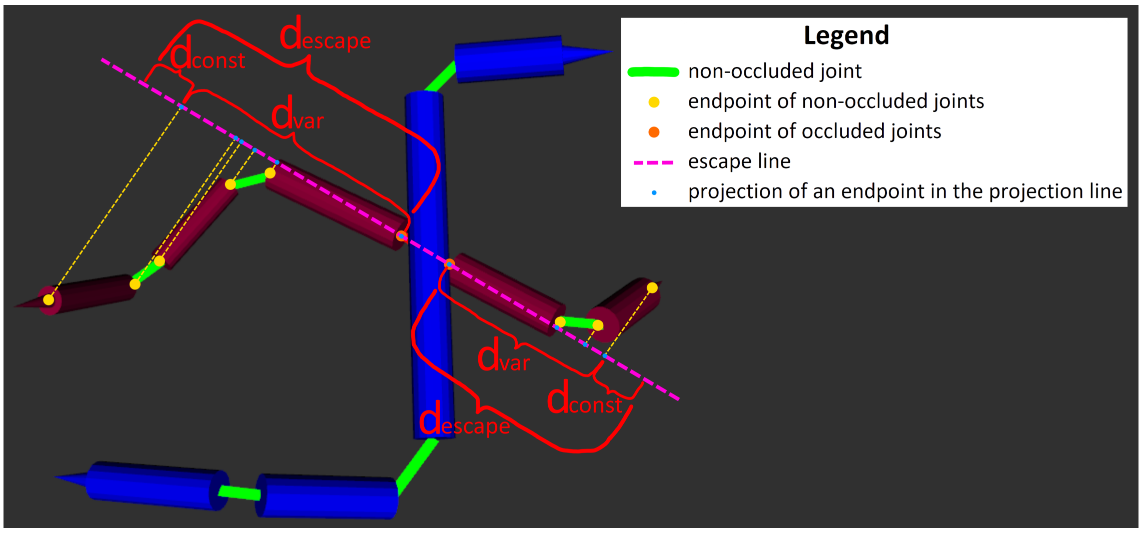

Legend
non-occluded joint
endpoint of non-occluded joints
endpoint of occluded joints
escape line
projection of an endpoint in the projection line
d_{escape}
d_{const}
d_{var}
d_{var}
d_{const}
d_{escape}

Figure 5.3: Visual representation of the escape line and distances for an occluded tube

To prevent the tube from colliding with the bin's walls, it is checked if the bounding box of the virtual tube (in the x and y axes) after applying the horizontal translation of the escape trajectory intersects a predefined rectangular safety zone, which is smaller than the bin's boundaries. If this intersection occurs, then the escape distance is shortened to be the largest distance along the escape line for which such a collision does not occur.

5.2.3.2 Trajectory evaluation

The synthesized trajectories alongside the tubes' geometric models are then sent to Gazebo. Each trajectory contains an identifier of the tube to be moved and an ordered list of waypoints.

In the simulated environment, the tubes are also modeled as a set of cylinders with the same lengths, radii and poses as those computed during the tube modeling phase. The only differences are that cylinders are created for each joint to connect each pair of endpoints and a sphere with a radius equal to the tube's radius is added to each endpoint involved in a joint. These two changes prevent the tubes from having gaps. To reduce the number of objects in the simulated environment, and thus decrease the computational effort, none of the models for the robot's parts are added to the simulated environment. Figure 5.4 presents some examples of simulated tubes in Gazebo.

Before spawning all of the tubes in the simulation world, a small amount of time is given to allow all objects to reach a *stable state*, where there are no longer moving due to gravity and contact forces between them. Before testing each trajectory, the world is reset back to this same stable state.

For each trajectory, the associated tube is moved at a constant velocity along each segment between consecutive waypoints until the last endpoint is reached. The velocity in the simulation environment is the same as the gripper's during the motion execution. The poses of the other tubes are computed at each time step using the physics engine. After each trajectory is concluded, the displacement of each tube is measured as the distance between the positions of its center of mass

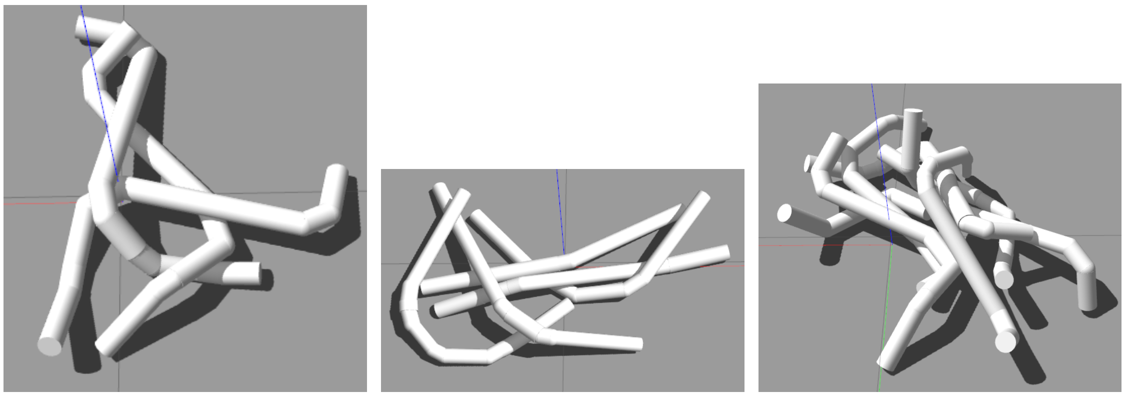

(a) Example with 3 tubes (b) Example with 5 tubes (c) Example with 7 tubes

Figure 5.4: Examples of sets of simulated tubes in Gazebo

after following the trajectory and on the stable state. The cost for each trajectory corresponds to the sum of the displacements of all tubes, with the exception the one that was intentionally moved. Unlike the entanglement metric proposed by Matsumura et al. which only accounts for parts that were lifted [MDWH19], this cost function not only enables the detection of cases where other tubes were unintentionally lifted, but also it is sensitive to trajectories that cause too much disruption in the bin.

5.2.4 Grasp planning

During the grasp planning phase, a set of grasps is created for each trajectory. This phase occurs after the trajectory planning phase since it is assumed that the tubes are rigid enough so their shape is not affected by the location of the grasping point. If the tubes were more flexible, then the trajectory planning should take into account the grasp's configuration.

Definition 5.2.7. A **grasp** for a virtual tube is defined by the 3D point $G_{point} = [G_x, G_y, G_z]$, known as the **grasping point**, which is the gripper's tool center point (TCP) position when it grasps the object. In addition, a grasp is associated with an orientation for the gripper, the fingertip positions, an approximation vector it follows before grasping the tube and a trajectory it follows after grasping.

5.2.4.1 Grasp synthesis

The grasping point is always defined to be along one of tube's cylinder axis. The gripper is always set to be completely vertical, and thus only the yaw is adjusted so that the planes defined by the contact points between the fingertips and the tube's surface are always normal to the surface's normal vector. The approach vector is also always set to be vertical.

For each trajectory, the grasping points are created by traversing the tube's cylinders and creating points at a constant distance between them and at a minimum distance from the cylinders' bases. Figure 5.5 shows examples of the grasping points generated for two sets of tubes. The grasping points are represented by a white sphere. In Figure 5.5b, the points are only computed for the non-occluded tubes, which are colored in green. The yellow tubes are weakly-occluded, while the red ones are strongly-occluded.

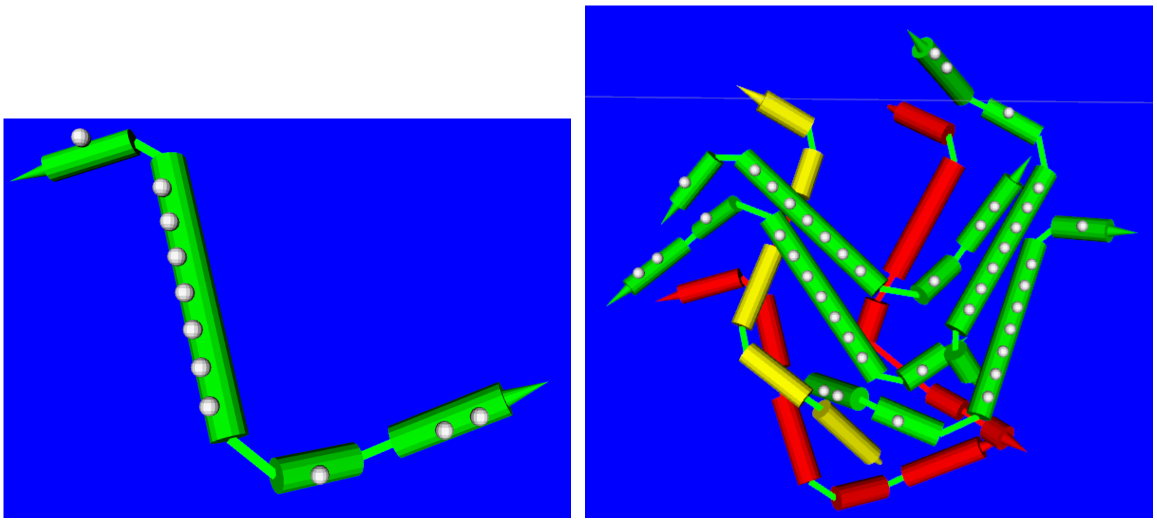

(a) Example with a single tube (b) Example with multiple tubes

Figure 5.5: Examples of grasping points on tubes

Three tests are performed to verify if a grasp is viable. Non-viable grasps are rejected, and not taken into account for the grasp evaluation.

Firstly, to determine if the gripper has a risk of colliding with either another tube or the bin's boundaries when executing a grasp, a simplified model of the parallel jaw gripper as two oriented bounding boxes with a gap between them slightly larger than a tube's radius is used. The bounding boxes are positioned and oriented according to the gripper's pose as it is about to grasp the tube. The raw point cloud (i.e. without any filters are applied to it) is used and the grasp is rejected if the number of points within the combined volume of both bounding boxes surpasses a certain threshold.

Secondly, to prevent the gripper from overstepping its working area, it is checked if, for each waypoint of the trajectory, the point obtained by adding the waypoint's translation to the grasping point lies inside a predefined bounding volume. If any of these points is outside of this zone, then the grasp is rejected.

Finally, grasps associated with an escape trajectory (and thus a weakly-occluded tube) are rejected if the grasping point lies in the section of the virtual tube that is expected to pass under the occluding tube. This prevents the gripper from colliding and dragging the occluding tube when following the trajectory. In Figure 5.3, escape trajectories that move the occluded tube to the upper-left corner can only have grasps whose points belong to the three cylinders on the left section.

5.2.4.2 Grasp evaluation

Each grasp is evaluated by associating it a cost G_{cost} using Equation 5.3, which reflects how likely the grasp will lead to an entanglement, where multiple tubes are picked at once. This cost is normalized to scale the $[0,1]$ range and consists of a linear combination of three normalized costs: the height cost, H_{cost}, the center cost, C_{cost}, and the trajectory cost, T_{cost}. The combined sum of the weights of these costs (H_{cost_weight}, C_{cost_weight} and T_{cost_weight}) is equal to 1.

$$G_{cost} = H_{cost} \cdot H_{cost\_weight} + C_{cost} \cdot C_{cost\_weight} + T_{cost} \cdot T_{cost\_weight} \tag{5.3}$$

The height cost H_{cost} (Equation 5.4) corresponds to the height of the grasping point relative to the heights of the tubes. The variable max_z is the z coordinate of the highest point of the filtered point cloud, which is obtained during the first step of the modeling phase and only contains points belonging to the tubes. It is assumed that the equation for the bin's bottom plane is $z = 0$.

$$H_{cost} = \frac{G_z}{max_z} \tag{5.4}$$

The center cost C_{cost} (Equation 5.5) reflects how far the grasping point is from the center of the tube $T_{center} = [T_x, T_y, T_z]$ along its curve length L. The rationale of this cost is that the farther the tube is grasped from its center, the higher the torque that will be generated, which can lead to more unstable grasps.

$$C_{cost} = \frac{dist(G_{point}, T_{center})}{L/2} \tag{5.5}$$

The trajectory cost T_{cost} (Equation 5.6) corresponds to an estimate provided by Gazebo during the trajectory analysis of how much the other tubes are moved when a given tube is grasped. This cost has the same value for all the grasps on the same tube, and is equal to 0.0 if the tube is non-occluded (as their trajectories do not require simulation). The displacement $disp$ is normalized using a maximum displacement value $disp_{max}$, above which it is considered that any trajectory has the same (high) odds of leading to a case of entanglement.

$$T_{cost} = \frac{min(disp, disp_{max})}{disp_{max}} \tag{5.6}$$

5.2.5 Entanglement detection

Before beginning the whole pick and place process, it is stored the sensor measurement for the force on the z+ axis (pointing upwards) being exerted on the flange when no object is being held. This measurement is referred to as the reference value for the force, F_{z_ref}.

After performing the grasping and lifting actions, the current force value on the z axis, F_z, is measured to determine how many tubes were picked. This requires prior knowledge of the tubes' mass and assumes all tubes have approximately the same mass. There are three possible cases: no object is being held, a single object is held and multiple objects are being held. These three cases are distinguished by computing the expected additional force exerted on the flange when a single tube is being held (which corresponds to the tube's weight), as presented in algorithm 3.

Algorithm 3 Entanglement detection algorithm

```
Input:
    $F_{z\_ref}$ - Reference z force (N)
    $F_z$ - Actual z force (N)
    $Mass$ - Tube mass (kg)
    $F_{tol}$ - Tolerance value for the force (N)
Output:
    $Outcome$ - Either $no\_objects\_held$, $single\_object\_held$ or $multiple\_objects\_held$
1: $Weight \leftarrow Mass * Gravitational\_acceleration$
2: $Expected\_force \leftarrow F_{z\_ref} - Weight$
3: if $F_z < Expected\_force - F_{tol}$ then
4:     $Outcome \leftarrow multiple\_objects\_held$
5: else if $F_z > Expected\_force + F_{tol}$ then
6:     $Outcome \leftarrow no\_objects\_held$
7: else
8:     $Outcome \leftarrow single\_object\_held$
```

When a single object is detected, the robot follows the default routine and proceeds to placing the tube on the destination area. If no object is detected, the robot restarts the bin picking process, and rescans the scene. Finally, if multiple objects are detected, a resolution routine is performed in an attempt to solve the entanglement problem, as explained in section 5.2.6.

5.2.6 Entanglement resolution

Similarly to the reference value for the z force, before scanning the bin for the first time, the sensor measurements for the torques exerted on the x and y axes are stored (M_{x_ref} and M_{y_ref} respectively).

When the z force is too low, which corresponds to a case of entanglement (multiple tubes were picked), the torque sensor measures the current values for the torque in the x and y axes, M_x and M_y, and the torque difference vector $\overrightarrow{\Delta M}$ is computed according to Equation 5.7.

$$\overrightarrow{\Delta M} = [M_x - M_{x\_ref}, M_y - M_{y\_ref}, 0]^T \tag{5.7}$$

This torque vector defines the axis around which the gripper and its payload would rotate if it was not attached to the robot arm, as seen in Figure 5.6.

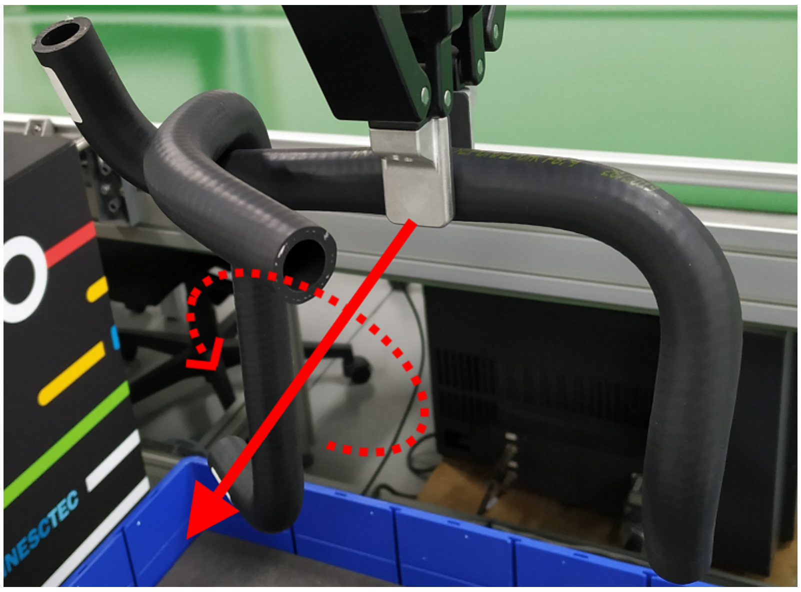


Figure 5.6: Visual representation of the torque vector

The gripper rotates around its tip in this axis and in the positive direction to try to make the undesired tubes drop back into the bin. After applying the rotation, the z force is once again measured to detect if the entanglement is still present. If so, the gripper drops the tubes back in the picking bin. If not, then the process proceeds normally and the single tube is placed on the destination.

5.2.7 Tube placement

When placing tubes on the destination area, it is important for them to have a consistent orientation so that they can be further processed more easily.

During this phase, it is assumed the gripper is holding a single tube. The points belonging to the tube are used to compute the object's principal axes using Principal Component Analysis (PCA), by performing eigendecomposition on the points' normalized covariance matrix [LR09]. The principal component with the highest eigenvalue provides a reliable estimate of the tube's orientation (i.e., if the tube is a perfect cylinder, then this principal component will correspond to the cylinder's axis). In order to ensure that the tube is placed with the correct orientation, the gripper performs a rotation to align the principal component with a predefined axis describing the desired orientation.

Optionally, to prevent the gripper from performing dangerous rotations, the principal direction vector can be projected in the xOy plane so that the alignment rotation only contains a yaw component.

Figure 5.7 presents an example of a coordinate frame that reflects the tube's orientation. The unit vector for the x-axis (in red), $\hat{x}$, is oriented according to the principal direction vector's projection on the xOy plane while the unit vector for the z-axis (in blue), $\hat{z}$, is equal to $[0,0,1]^T$. The vector for the y-axis (in green) is simply equal to $\hat{z} \times \hat{x}$. The tube is placed with a movement for the gripper that allows this coordinate frame to be aligned with a predefined frame in the destination area.

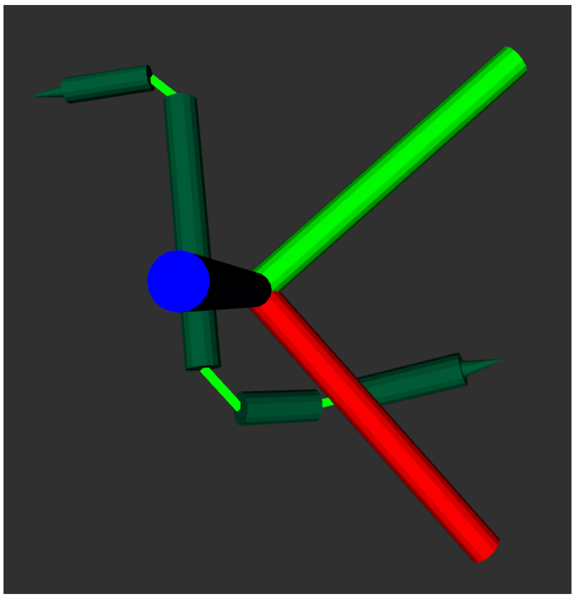


Figure 5.7: Example of the coordinate frame for a tube's orientation

5.3 Experiments

5.3.1 Hardware

This section presents the robot hardware that was used, and explains why these models were selected.

5.3.1.1 Yaskawa HC10

The model chosen for the robot arm was Yaskawa Motoman HC10[8] (Figure 5.8). As indicated by the model's name, this arm has enough power to handle payloads of up to 10 kg. It is a stationary robot arm with a maximum reach of 1200 mm that can be mounted on a horizontal base.

The manipulator is segmented into several joints and has a total of 6 axes. Each axis is equipped with a motor and multiple sensors, namely force/torque sensors.

The arm is connected to a YRC1000 controller[9], which is responsible for issuing commands to the robot and collecting information from its sensors. The controller includes a programming pendant that allows users to manually control the arm and monitor its operation. Data between the controller and the personal computer is exchanged via Ethernet/Industrial Protocol (EtherNet/IP).

This robot arm was chosen due to the fine precision of its movement and its ability to pick up multiple tubes at once, whose individual weight is at most 0.11 kg.

[8]`https://www.yaskawa.eu.com/en/products/robotics/motoman-robots/productdetail/product/hc10-1`
[9]`https://www.motoman.com/en-us/products/controllers/yrc1000`

5.3.1.2 Robotiq 2F-85

The robot arm presented in the previous subsection does not contain an end effector, the last link of the robot, which will be used to grab objects. Thus, a Robotiq 2F-85 gripper[10] (Figure 5.9) was attached to Yaskawa's endpoint.

The gripper is composed of two fingers and a single motor. It is rotational, which means that it can adapt to the shape of the object being grasped, without the need for custom fingertips. It has a stroke width (maximum distance between its fingertips when fully opened) of 85 mm.

This gripper was chosen since using two fingers is the most appropriate for picking up linear objects (by performing a pinching motion). If the gripper had three fingers, then the forces applied to the pipes would be asymmetrical, which would make the task of picking a pipe harder. In addition, its stroke width is compatible with the various types of tubes that are used, which have a maximum diameter of approximately 2.5 cm.

Figure 5.8: Yaskawa Motoman HC10[11]

Figure 5.9: Robotiq 2F-85[12]

5.3.1.3 PhoXi 3D Scanner

A Photoneo PhoXi 3D Scanner S[13] (Figure 5.10) was used to obtain a three-dimensional view of the scene. The scanner has a range from 384 to 520 mm and a scanning area of 360 x 286 mm at

[10] https://robotiq.com/products/2f85-140-adaptive-robot-gripper
[11] https://cobotsguide.com/2016/06/yaskawa-motoman-hc10
[12] https://www.thinkbotsolutions.com/shop/robotiq-2-finger-85
[13] https://www.photoneo.com/products/phoxi-scan-s

the optimal distance (442 mm). The scanning process takes at most 2250 ms and has a calibration accuracy of 0.05 mm.

The device uses structured lighting in order to build the 3D point cloud, where it emits controlled patterns of light (in this case, from the visible spectrum) onto the scene. This method allows the device to reconstruct non-textured objects. The objects' surfaces induce deformations on the light pattern, which is then acquired by a camera. These deformations allow the computation of a point cloud, a set of three-dimensional points that cover the scene's surfaces. The PhoXi 3D scanner, in particular, has a resolution of 3.2 million points.

The sensor is connected to the computer via Ethernet and can be controlled via the PhoXi Control Application. This software has a Graphical User Interface (GUI), that can be used to set up the environment and the device's settings, and an API, so that the scanner can be controlled programmatically.

The sensor can be fixed in a static position or mounted on the robotic manipulator. The second option was preferred since it allows the scanning pose of the scene to be changed with little effort.

It should be noted that the sensor does not capture the objects' color, which is not an issue since all of the tubes have little to no contrast.

5.3.1.4 ATI Gamma SI-65-5

The force-torque sensors that are embedded on the robot arm are meant to prevent the robot from performing a movement if too much force is exerted on one of its segments. As such, these sensors do not have enough precision for a specific goal of this work: determining the torque of the payload on the gripper. To achieve this purpose, an ATI Gamma SI-65-5 force-torque sensor[14] (Figure 5.11) was inserted between the robot arm's flange and the gripper.

This sensor outputs six values: the force and torque in the x, y and z directions. It is able to sense forces of up to 200 N in the z direction (which is more than enough for this dissertation's goals[15]), with a resolution of 1/40 N. It can also detect torques of up to 5 N·m in each direction, with a resolution of 10/13333 N·m.

[14] https://www.ati-ia.com/products/ft/ft_models.aspx?id=Gamma

[15] Since the heaviest tubes weight 0.11 kg, then, in the worst case scenario where 7 tubes are picked at once, the force exerted on the gripper is approximately equal to $7 \times 0.11 \times 9.81 \approx 7.6$ N, which is much less than 200 N.

Figure 5.10: Photoneo PhoXi 3D Scanner S[13]

Figure 5.11: ATI Gamma SI-65-5[14]

5.3.2 Setup

Figure 5.12 provides an overview of the arrangement used to conduct the experiments. The depth and force/torque sensors alongside the gripper described in the previous section are shown in Figure 5.13.

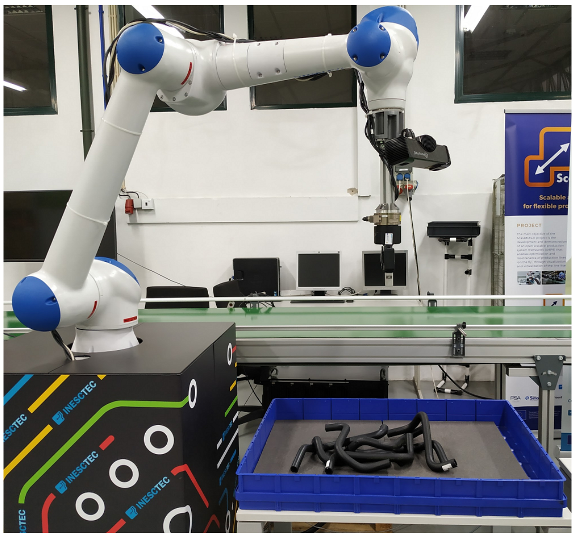


Figure 5.12: Setup overview

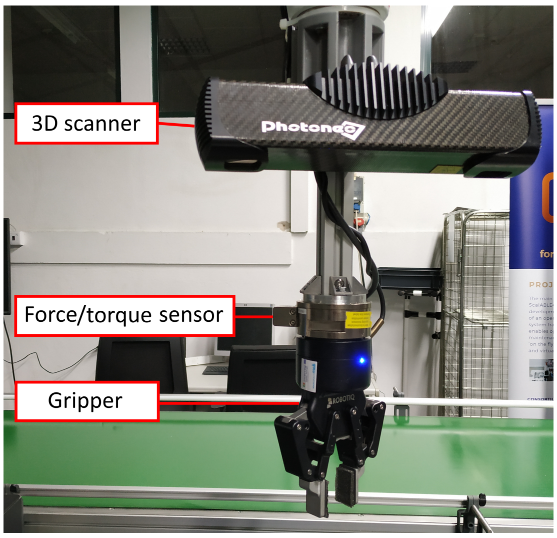


Figure 5.13: Tools attached to the flange

The tubes to be picked were placed in a bin, whose dimensions are approximately 77 cm (length) x 58 cm (width) x 9 cm (height). Two sets of 7 tubes were used (Figure 5.14). Their properties are summarized in Table 5.2.

The tubes on set A are the same as those used on the experiments of Section 4.4, dedicated to the tube modeling phase. They were straight plumbing pipes that were manually bent to have different shapes. They correspond to a simpler test case since their material has less friction, and thus the tubes that are on top of the one being picked tend to slide back into the bin.

The tubes from set B are radiator hoses from a Fiat Tipo engine. These tubes are more prone to entanglement since their curves are not planar and their rubber surface provides more friction.

Table 5.2: Main properties of the considered sets of tubes

	Set A	**Set B**
Material	PVC	Rubber
Curve length	50 cm	40 cm
Radius	1.25 cm	1.00 cm
Stiffness	Fully rigid	Semi-rigid
Weight	55 g	110 g

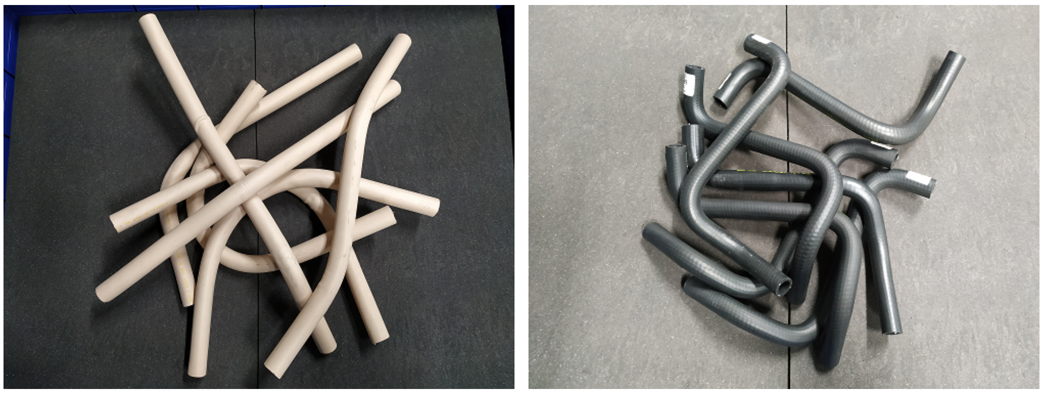

(a) Set A (b) Set B

Figure 5.14: Sets of tubes used in the experiments

For each set of tubes, 20 rounds of bin picking were performed, where a round consists of manually placing the tubes in a random arrangement in the bin and then making the robot follow the whole picking pipeline of Section 5.2 in order to empty the bin by picking the tubes one-by-one.

5.3.3 Parameter values

Table 5.3 presents the parameter values used when conducting the experiments.

Table 5.3: Parameter values used in the experiments

Phase	Parameter	Set A	Set B
Tube modeling	Cloud point size after filtering	100000	
Tube classification	Tube minimum length	35 cm	30 cm
Trajectory synthesis	Upwards trajectory movement	40 cm	
	Escape trajectory first upwards movement	2 cm	
	Escape trajectory second upwards movement	40 cm	
	d_{const}	3 cm	
Grasp synthesis	Minimum distance between grasping points and cylinder bases	2 cm	
	Maximum allowable amount of points in the gripper's bounding boxes	20	
Grasp evaluation	$disp_{max}$	1 m	
	H_{cost_weight}	0.2	
	C_{cost_weight}	0.5	
	T_{cost_weight}	0.3	
Entanglement resolution	Rotation angle	45°	

5.3.4 Results

The *Entangled Tubes Bin Picking* dataset[16] presented in Chapter 4 was expanded to include the files of all the point clouds acquired by the 3D scanner during the experiments. The dataset's repository also includes a link to raw footage of all the experiments.

Table 5.4 presents the solution's success rate for emptying the bin. The 6 unsuccessful cases were due to tubes that moved to the corners of the bin, where the scanner could not detect them, causing the round to end prematurely. The sensor could not scan the bin from a higher position since the cloud points it acquired had too much noise due to the scanning distance exceeding by a large amount the optimal range. A smaller bin was not used so that the robot could freely experiment escape trajectories without the risk of either the gripper or the grasped tube colliding with the bin's walls. As future work, multiple scans of the bin from different positions can be acquired or a sensor with a larger working area can be used.

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

Table 5.4: Percentages of rounds where the bin was successfully emptied

Set A (PVC)	90%
Set B (rubber)	80%

In all of the rounds, when all of the tubes were brought to the destination area, the robot was able to infer that the bin was empty by the latest acquired point cloud and end the picking process.

There were no cases in which multiple tubes were carried to the destination area or where a tube fell outside the bin. Therefore, every time a single tube was lifted from the bin, it always reached the destination area.

In the results and discussion that follow, let a *picked tube* be a real-life tube that was successfully placed in the destination area in a given round, possibly after multiple attempts (the last one being the *successful attempt* or *successful pick*).

Table 5.5 shows the percentage of picked tubes that were successfully placed on the first try. The high overall success rates alongside the significant amount of picked tubes indicate that the proposed algorithm had a very positive performance for both tube sets. As expected, the success rate for set B is slightly smaller than for set A since the former tubes' geometry and material made them more prone to entanglement. In set B, the lowest success rate corresponds to cases where there were 7 tubes in the bin, which is likely due to the presence of more complex tube arrangements. Conversely, in both sets, there is a 100% success rate of picking tubes at the first attempt when the bin only contains 1 or 2 objects. The results seem to indicate that the success rate does not increase at a constant rate as the number of remaining tubes decreases. It is speculated that this is due to the algorithm giving preference to picking the non-occluded tubes first, which postpones solving entanglement issues. This property may be advantageous since the contact forces exerted by the tubes that are picked first may move the other ones in such a way that it is easier to pick them later on.

Table 5.5: Percentages of picked tubes that were successfully placed on the first attempt

Number of tubes in the bin		7	6	5	4	3	2	1	Overall
Set A	Success rate	100%	95%	100%	100%	100%	100%	100%	99%
	Number of picked tubes	20	20	20	20	20	19	18	137
Set B	Success rate	80%	95%	100%	90%	90%	100%	100%	93%
	Number of picked tubes	20	20	20	20	20	19	16	135

Table 5.6 indicates how often the robot resorted to the simulator for trajectory evaluation (i.e. how often all of the tubes had occlusions) when attempting to pick a tube. It also shows how often each type of trajectory was used to move the gripper after grasping. The simulator was used much more often with set B since the tubes' shape lead to more cases of occlusion. Most of the times,

an upwards trajectory was preferred over the escape ones, that performed an additional lateral movement.

Table 5.6: Percentages of picking attempts where trajectory simulation was performed and frequency of both types of trajectories on the chosen grasp

Number of tubes in the bin		7	6	5	4	3	2	1	Overall
Set A	Usage rate (upwards trajectories)	0%	9.5%	10%	10%	5%	0%	0%	5%
	Usage rate (escape trajectories)	0%	4.8%	0%	10%	0%	0%	0%	2%
	Usage rate (total)	0%	14.3%	10%	20%	5%	0%	0%	7%
	Number of picking attempts	20	21	20	20	20	19	18	138
Set B	Usage rate (upwards trajectories)	25%	10%	5%	27%	5%	0%	0%	11%
	Usage rate (escape trajectories)	4%	0%	0%	0%	27%	11%	0%	6%
	Usage rate (total)	29%	10%	5%	27%	32%	11%	0%	17%
	Number of picking attempts	24	21	20	22	22	19	16	144

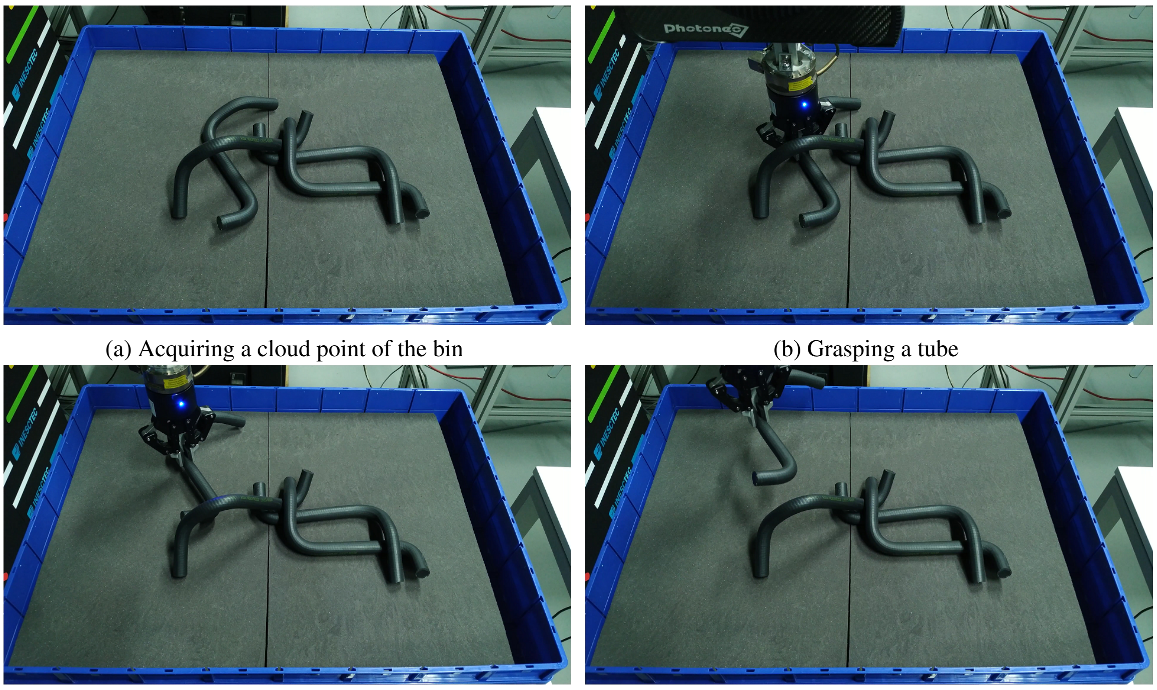
Interestingly, the trajectory simulator was only used on first attempts to pick a tube. As a result, during the second attempts, there was always at least one non-occluded tube in the pile. In addition, every single second attempt at picking a tube was successful, for both sets A and B. This evidences that simply dropping the grasped tube back into the bin when it is entangled is a good strategy for bin picking.

Table 5.7 presents the success rate of the picking attempts which used the simulator. Unlike the previous tables, these rates were not presented in function of the number of tubes in the bin, as there were not enough examples for each amount. Set A had a success rate of 100% for both kinds of trajectories, which is most likely due to the PVC tubes' surface allowing them to slip and fall back into the bin much more easily. The results from set B show a very interesting result: the success rate for the escape trajectories is significantly higher than the one for the upwards trajectories. By combining observations from tables 5.6 and 5.7, one can conclude that it is harder to execute a useful escape trajectory than to simply lift a tube, but when the former alternative is possible, it yields much better results. This conclusion suggests that having a wider variety of trajectories to choose from will increase this solution's overall performance.

Table 5.7: Percentages of picking attempts where trajectory evaluation was performed that resulted in a successful pick

Set A	Success rate (upwards trajectories)	100%
	Success rate (escape trajectories)	100%
	Success rate (total)	100%
	Number of upward trajectories	7
	Number of escape trajectories	3
	Number of uses of simulation	10
Set B	Success rate (upwards trajectories)	63%
	Success rate (escape trajectories)	78%
	Success rate (total)	68%
	Number of upward trajectories	16
	Number of escape trajectories	9
	Number of uses of simulation	25

Figure 5.15 depicts a successful pick with an escape trajectory. In this example, neither the grasped tube nor the gripper touched any of the other tubes.



(a) Acquiring a cloud point of the bin

(b) Grasping a tube

(c) Moving the tube laterally with an escape trajectory

(d) Lifting the tube

Figure 5.15: Successful escape trajectory example

Table 5.8 shows how often the entanglement resolution phase was used. Since set A had a 100% picking success rate with the simulator, this phase was never needed. For set B, the rate is also quite low since the trajectory simulator had a moderately high success. The usage rate does not seem to vary significantly with the amount of tubes, which is coherent with the previous observation that this solution solves tougher cases of entanglement as late as possible during a round.

Table 5.8: Percentages of picking attempts where the entanglement resolution phase was performed

Number of tubes in the bin	7	6	5	4	3	2	1	Overall
Set A	0%	0%	0%	0%	0%	0%	0%	0%
Set B	13%	5%	0%	14%	14%	5%	0%	8%

Table 5.9 presents the success rate of the entanglement resolution phase. The success rate for set B was rather low most likely due to one key observation that was made when the experiments were conducted: most of the times, the rotation of 45° was not enough to make the other tubes fall because the friction for the rubber material was too high. Higher angles were not attempted as they increased the chances of the tubes falling outside the bin. To increase the likelihood of solving the entanglement, a positive and negative rotation of 45° (or possibly more, with the appropriate setup) could be applied according to an axis contained in the xOy plane that is perpendicular to the torque vector $\overrightarrow{\Delta M}$. Furthermore, negative rotations could be experimented around the axis defined by this vector.

Table 5.9: Percentages of picking attempts where the entanglement resolution step was performed that resulted in a successful pick

Set A	Success rate	-
	Number of uses of entanglement resolution	0
Set B	Success rate	27%
	Number of uses of entanglement resolution	11

In none of the experiments was there a case where all of the tubes had two or more occlusions (strongly-occluded tubes). In these scenarios, one could experiment using the upwards trajectory and, as an alternative, a set of escape trajectories that take into account the properties of each occlusion. Then, the simulator could be used to evaluate the trajectories.

Table 5.10 presents the execution time of different phases of the algorithm considering all of the picking attempts (*all cases*), only the attempts that did not use the simulator (*cases without simulation*) and only the attempts that did use the simulator (*cases with simulation*). *Total processing time* is the time taken by the solution to perform all of the computations, including those of the tube modeling, motion planning and grasp planning phases, as well as other operations, including the PCA computations of the tube placement step and converting the point cloud from a *sensor_msgs::PointCloud2* to a *pcl::PointCloud<pcl::PointXYZRGBNormal>::Ptr* object. *Planning time* is the combined time taken by the motion and grasp planning phases.

As seen in Table 5.10, using the simulator has a high impact on the total processing time. For future work, this time can be significantly reduced by speeding up the simulation by increasing the simulation step size. Since the focus of this chapter's experiments was accuracy, the simulation was set to run with a maximum step size of 0.001 and an update rate of 1000 Hz, which results in the simulation time running approximately as fast as real time[17]. Moreover, unlike in the experiments of Chapter 4, the whole, unfiltered point cloud was used to estimate the surface normals and curvatures, to ensure optimal accuracy for the modeling phase. As a result, by downsampling the cloud, as presented in Section 4.4.4, it is possible to reduce the modeling time to around 1 s, which would reduce the average total processing time to around 5 s and 7 s for Set A and Set B, respectively, when all picking attempts are considered. As mentioned in Section 4.5, an interesting line for future research is to use the automated modeling evaluation system to determine an optimal downsampling filter for the point cloud that balances processing time and performance.

The total processing time for Set A is lower than for Set B when all test cases are considered. This is to be expected since the simulator is used much more often in Set B. However, both the modeling time and the total processing time without the simulator are lower in Set B than in Set A. One possible explanation is that this is due to the tubes in Set A having a larger radius (which increases the time needed to estimate the normals and curvatures since a point has more close-by neighbors) and length (which causes more grasping points to be generated since the cylinders are longer).

The large values for the standard deviation of the planning and total processing times for the cases with simulation, in comparison to those of the cases without simulation, show that there is a very high variability in the time needed for motion planning, possibly due to an equally high variability on the amount of trajectories to evaluate.

Table 5.10: Statistics for the execution times of different sets of phases

		Set A	**Set B**
Total processing time, all cases	Average	7.38 s	8.72 s
	Std. deviation	13.28 s	14.59 s
Total processing time, cases without simulation	Average	4.13 s	2.68 s
	Std. deviation	1.07 s	0.73 s
Total processing time, cases with simulation	Average	48.99 s	37.47 s
	Std. deviation	23.52 s	14.95 s
Modeling time	Average	3.15 s	2.11 s
	Std. deviation	0.81 s	0.58 s
Planning time, all cases	Average	4.06 s	6.46 s
	Std. deviation	13.15 s	14.51 s
Planning time, cases without simulation	Average	0.84 s	0.44 s
	Std. deviation	0.41 s	0.23 s
Planning time, cases with simulation	Average	45.27 s	35.11 s
	Std. deviation	23.55 s	14.84 s

[17] http://gazebosim.org/tutorials?tut=physics_params&cat=physics

5.4 Main challenges

During the work of this chapter, several difficulties were faced. These struggles required a good amount of reasoning and creativity to be overcome. This section presents the most relevant challenges.

5.4.1 Performing realistic simulations in Gazebo

One of the main difficulties was being able to simulate with an acceptable amount of realism the tubes' escape trajectories in Gazebo during the trajectory evaluation phase. Several weeks of trial and error experiments, alongside the examination of the source code, documentation and forums of both Gazebo and the underlying physics engine, ODE, were required to advance in this part of the project.

In earlier versions of the trajectory planning plugin for Gazebo, two of the most common issues with the tube whose trajectory was being simulated were it clipping occasionally through other tubes and being subject to contact forces with the other tubes. The first issue was unacceptable since it significantly reduced the cost of trajectories that should generate a high amount of tube displacement. The clipping was making tubes fall back into the bin when they should have been lifted by the tube that was being grasped. The second issue also lead to highly unrealistic scenarios since the grasped tube deviated from its planned trajectory, whereas in real-life the contact forces with the other tubes are negligible when compared to the forces exerted by the gripper. Typically, these two issues did not occur simultaneously. The first problem was present mainly when the grasped tube was set to be *static*, which prevented the physics engine from updating its pose at each time step, as opposed to the other tubes which were *dynamic*. The clipping problem seemed to occur due to the physics engine not being able to handle consistently the collision between dynamic bodies and a static object that was programmatically forced to move at each time step (to follow the planned trajectory). Thus, the logical solution was to set the grasped tube to also be dynamic, but this led to the second problem.

The complex issue was solved using a rather creative approach, which involved creating objects without mass in an out-of-bounds area (so that they did not interact with the tubes) and connecting one to each tube using an intangible fixed joint. The position of the *mass-less* object corresponding to the grasped tube was then updated at each time step to follow the planned trajectory, which allowed the tube to be dynamic without suffering practically any external force due to contact with the other tubes and with reduced occurrences of clipping. In addition, to make the simulation more stable, the step size was adjusted so that the simulation time ran approximately as fast as real time, as explained in Section 5.3.4.

5.4.2 Performing the entanglement resolution rotation

Another engaging challenge was commanding the robot arm to perform a 45° angle rotation around the axis defined by the torque vector without exceeding the arms' joint limits. Most of Yaskawa's joints, including the one that connects it to its flange (joint T), could not be freely rotated around their axes. The issue was that the angle for joint T was affected by the entanglement recovery rotation, and, depending on the direction of the torque vector, this occasionally led to this angles' limits being reached (below -180° and above +180°), which stopped the robot and halted the bin picking process.

This problem was solved by making the robot arm perform a rotation around the z axis to align the torque vector with a predetermined axis, around which the robot could comfortably rotate 45º without exceeding the joint limits. The solution also had the advantage of limiting the range of the robot arm's movements, thus allowing it to operate in a more predictable and safe manner.

5.5 Summary

This chapter detailed the full bin picking system proposed by this dissertation's work to pick a set of entangled tubes from a heap. It is built upon the modeling solution presented in the previous chapter.

The chapter began by providing an overview of how the system was implemented. Then, each phase of the solution's algorithm was detailed. Real-life experiments with sets of PVC and rubber tubes to evaluate the solution's accuracy and performance were then presented and discussed.

Chapter 6

Conclusions

As established by the thorough state-of-the-art in Chapter 2, a lot of research was devoted to the topic of robotic bin picking, yet very few of it explores and proposes solutions to the problem of entanglement, where multiple items are inadvertently picked at once. By focusing on the case of tube-shaped objects, which are very common in industrial processes, this dissertation proposed novel approaches to handle this challenge.

Driven by the research question *How can the pose and shape of tubular objects be estimated?*, this work proposed in Chapter 4 a technique that involves using a point cloud acquired by a depth sensor and applying four successive operations on it. The point cloud is first filtered to remove points that do not belong to the tubes, followed by being segmented into a set of disjoint regions that represent visible portions of a tube. Then, rather than using models with a high amount of parameters, such as splines, a set of cylinders is fitted to each segment to recreate the tube's curvatures. Finally, the cylinders are combined into complete tubes according to their euclidean and angular distances. Certain heuristics, such as preventing tubes from exceeding a maximum length, filter some incorrect pairings of cylinders. Results with sets of PVC tubes showed that the solution had an average execution time of 1.28 s. In addition, this computational effort does not seem to increase proportionally to the amount of tubes, since there can only be so many items simultaneously visible in a given capture, due to occlusions. This makes the solution attractive for industrial use, where cycle times have to be kept low and systems must be scalable. The solution exhibited a high accuracy, by being able to correctly model an average of 7 tubes in bins with 10 tubes. It is not necessary for the models of all the tubes to be accurate since only one is picked at a time, and, as the bin is progressively emptied and the number of occlusions decreases, new tubes become easier to be correctly modeled. This observation is supported by the experiments, that show the accuracy increases as the amount of tubes in the bin decreases. The modeling algorithm presented some difficulties in detecting small sections of tubes, which could be resolved by removing less points with the random filter in the filtering step.

After having a model for the set of entangled tubes, this work moved on to the question *What* *are the criteria for good grasping configurations and trajectories for picking entangled objects?* in Chapter 5. The tube models were used to determine which tubes were not occluded, in order to solve simpler cases. When all of the tubes were occluded, information about where the occlusions were located was exploited to derive the direction and distance for a set of candidate post-grasp trajectories for the gripper that moved a tube laterally before lifting it. This novel motion planning approach resorted to Gazebo and the ODE physics engine to evaluate each candidate and determine which ones were least likely to lead to cases of entanglement, by measuring the displacement of the tubes. For grasp planning, a data-driven method sampled the cylinders of the most promising tubes to locate adequate grasping poses and ranked them according to computationally efficient heuristics, such as the height of the grasping point and its distance to the tube's center. Both motion and grasp planning, also avoid collisions between the gripper or the grasped tube with the environment. The usage of heuristics during the planning phases has the advantage of allowing users to easily understand why a given decision was made. As a last resort, a force/torque sensor was used to determine how many tubes were actually lifted by the robot and act accordingly. If multiple tubes were lifted, the torque data was used to derive a rotation for the robot arm that attempts to release the tubes that were accidentally picked. Real-life experiments conducted with rubber radiator hoses showed that the system had good accuracy as it was able to successfully pick 93% of the tubes on the first try. The simulator was highly relevant to these positive results as it was used on 17% of the attempts with a success rate of 68%. Trajectories that included lateral movements were used less often than simple lifting motions, but the former yielded a higher success rate. This suggests that finding new strategies for disentanglement based on the geometric relationship between the various items is very promising. The entanglement recovery system via force/torque analysis was also shown to be useful, with a usage rate of 8% and a success rate of 27%. This rate can be improved by applying rotations with higher angles and possibly different axes, which can only be achieved safely using an appropriate setup for the robot, its attached tools and surrounding environment.

Regarding the question *How can existing bin picking solutions be applied for entangled tubes?*, the system developed during this dissertation was inspired by the work of various researchers. None of the articles from the state-of-the-art helped with the solution as a whole. Instead, a subset of the papers provided valuable clues for particular phases of the picking process. The four most important contributions were:

- Qiu et al. [QZN14], who also combined cylinders into more complete objects based on their euclidean and angular distances. This inspired the tube joining step of the modeling phase. Their work was not devoted to bin picking and was instead related to the reconstruction of pipe installations.

- Miller et al. [MKCA03], who proposed the idea of modeling objects as a set of primitives (which inspired the cylinder fitting step of the modeling phase) and sampling their surface for grasp candidates using simple rules and constraints that reduce the grasp configuration's number of dimensions (which inspired the grasp synthesis step for grasp planning). This resulted in a decrease of the execution time for grasp synthesis and an increase on the probability that a grasp led to a successful pick.

- Matsumura et al. [MDWH19], who also used simulation to predict entanglement in bin picking. The main differences are that they used the simulator to train a predictor and they only resorted to lifting movements. Due to how recent their work is, this dissertation's author idea to use simulation for motion planning was not inspired by their article. Instead, the similarities of this work with the dissertation confirmed a posteriori that using a physics engine was a pertinent approach.

- Moreira et al. [MRP$^+$16], who gave the idea of using force/torque analysis to determine how many tubes were picked, thus providing a strong failure detection system.

Due to its innovative aspects, this research and experimental results, which were shared with the scientific community with two articles, are relevant to the progress of the state-of-the-art regarding bin picking, and, at a higher level, robotic manipulation.

6.1 Main contributions

The main contributions of this dissertation are:

- A complete bin picking system for entangled tubes, which comprises:
  - A perception algorithm capable of accurately estimating the pose and shape of partially visible tubes placed in a bin in a random arrangement, without a predefined model of the objects.
  - A grasp planning algorithm that uses the information of the perception algorithm to compute grasp configurations that account for occlusion and entanglement.
  - A motion planning system that generates a set of trajectories, which include upward and lateral movements, and uses a physics engine to determine which candidates have the least chances of leading to a case of entanglement.
  - An entanglement detection and recovery system that uses force/torque data to determine how many tubes were picked and compute a rotation for the robot manipulator that has a significant chance of solving cases of entanglement.

- A publicly available dataset[1] associated with all of the experiments conducted in this work. It can be used to evaluate other entangled tube perception algorithms. The dataset also includes videos for all 40 experiments used to measure the picking system's accuracy.

- A conference article [LCSV19] that details the perception algorithm for the pose and shape estimation of entangled tubes.

- A journal article [LCSV20] that details the bin picking system as a whole.

[1] `https://github.com/up201406036/Entangled-Tubes-Bin-Picking-Dataset`

6.2 Future work

This dissertation work opens several lines for further research.

Firstly, it would be highly relevant to develop a complete module for the automated evaluation of the modeling algorithm. Section 4.5 presented in a high level of detail the work that was already done for this module and the next steps. This module could then be used to study in a more rigorous manner the limitations of the algorithm and possibly a way of automatically adjusting all of its parameters according to the tubes' properties (namely radius, length, mass) in such a way that the solution's accuracy and execution time are balanced.

Secondly, to increase the attractiveness of this solution for deployment in industrial settings, it is important to reduce the execution time, namely for the motion planning phase with the simulator. As mentioned in Section 5.3.4, experiments can be conducted to test how much the simulation can be sped up without causing too much instability, which notably leads to cases of objects clipping.

Thirdly, to increase the probability of successfully picking a single tube at a time, more types of trajectories could be generated. In particular, one could determine which tubes are occluding a given tube and compute an escape trajectory based on the formers' spatial configuration. It should be noted that the amount of trajectory candidates should be kept reasonably low to prevent the cycle time from increasing significantly due to the time needed for evaluation via simulation.

Machine learning techniques could also be explored to perform motion and grasp planning. This alternative has the potential of having high capabilities of generalization to objects with new shapes. The models could be trained to detect entanglement using this work's proposed technique for tube modeling, or by using directly a depth image as input, in a similar fashion to the recent contribution by Matsumura et al. [MDWH19]. In the former case, the module described in Section 4.5 could be used to generate labeled examples for training.

Finally, this picking system could be extended to work with a broader spectrum of shapes and with flexible objects (such as wires or cloth), for which there is very little research for pick-and-place systems. In this case, the simulation in Gazebo can be adapted so that it takes into account the material's flexibility. One interesting case study are the automobile air conditioning system tubes used by Moreira et al. [MRP+16] and on the STAMINA project (presented in Section 2.2). These tubes are flexible and contain several bifurcations and attachments.

References

[ACVB09] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, may 2009.

[AIG19] Marcos Alonso, Alberto Izaguirre, and Manuel Graña. Current Research Trends in Robot Grasping and Bin Picking. In *International Joint Conference SOCO'18-CISIS'18-ICEUTE'18*, volume 771, pages 367–376, San Sebastian, Spain, jun 2019. Springer Verlag.

[ATGD99] Alfred Anglani, Francesco Taurisano, Roberto De Giuseppe, and Cosimo Distante. Learning to Grasp by using Visual Information. In *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 7–14, Monterey, CA, USA, 1999. IEEE.

[BADA18] Pablo J. Alhama Blanco, Fares J. Abu-Dakka, and Mohamed Abderrahim. Practical Use of Robot Manipulators as Intelligent Manufacturing Systems. *Sensors*, 18(9):2877, aug 2018.

[Bar81] Alan H. Barr. Superquadrics and Angle-Preserving Transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, jan 1981.

[BFH04] Christoph Borst, Max Fischer, and Gerd Hirzinger. Grasp planning: How to choose a suitable task wrench space. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pages 319–325, New Orleans, LA, USA, 2004. IEEE.

[BH86] Robert C Bolles and Radu Patrice Horaud. 3DPO: A Three Dimensional Part Orientation System. *The International Journal of Robotics Research*, 5(3):3–26, 1986.

[BJRL$^+$11] Jeannette Bohg, Matthew Johnson-Roberson, Beatriz León, Javier Felip, Xavi Gratal, Niklas Bergström, Danica Kragic, and Antonio Morales. Mind the gap - Robotic grasping under incomplete observation. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 686–693, Shanghai, China, 2011. IEEE.

[BM92] Paul J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, feb 1992.

[BMAK13] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-Driven Grasp Synthesis - A Survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.

[BO08] Kay Boehnke and Marius Otesteanu. Progressive Mesh based Iterative Closest Points for Robotic Bin Picking. In *Proceedings of the Fifth International Conference on Informatics in Control, Automation and Robotics*, pages 121–124, Funchal, Portugal, 2008.

[BP07] Ulrich Bauer and Konrad Polthier. Parametric reconstruction of bent tube surfaces. In *Proceedings - 2007 International Conference on Cyberworlds, CW'07*, pages 465–474, Hannover, Germany, oct 2007. IEEE.

[Buc15] Dirk Buchholz. *Bin-Picking – New Approaches for a Classical Problem.* Phd thesis, Technische Universität Braunschweig, 2015.

[BZK+03] Faysal Boughorbel, Yan Zhang, Sangkyu Kang, Umayal Chidambaram, Besma Abidi, Andreas Koschan, and Mongi Abidi. Laser ranging and video imaging for bin picking. *Assembly Automation*, 23(1):53–59, mar 2003.

[CBR16] Pavel Chmelar, Ladislav Beran, and Lubos Rejfek. The Depth Map Construction from a 3D Point Cloud. In *MATEC Web of Conferences*, volume 75, pages 1–5, 2016.

[CDN19] Marco Costanzo, Giuseppe De Maria, and Ciro Natale. Two-Fingered In-Hand Object Handling Based on Force/Tactile Feedback. *IEEE Transactions on Robotics*, 36(1):157–173, oct 2019.

[CM91] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. In *1991 IEEE International Conference on Robotics and Automation*, volume 3, pages 2724–2729, Sacramento, CA, USA, 1991. IEEE Comput. Soc. Press.

[CRC18] Shehan Caldera, Alexander Rassau, and Douglas Chai. Review of Deep Learning Methods in Robotic Grasp Detection. *Multimodal Technologies and Interaction*, 2(3):1–24, sep 2018.

[CYK19] Yukyu Chan, Hungchen Yu, and Rebecca P. Khurshid. Effects of Force-Torque and Tactile Haptic Modalities on Classifying the Success of Robot Manipulation Tasks. In *2019 IEEE World Haptics Conference, WHC 2019*, pages 586–591, Tokyo, Japan, jul 2019. IEEE.

[DBP+09] Renaud Detry, Emre Baseski, Mila Popovic, Younes Touati, Norbert Kruger, Oliver Kroemer, Jan Peters, and Justus Piater. Learning Object-specific Grasp Affordance Densities. In *2009 IEEE 8th International Conference on Development and Learning*, pages 1–6, Shanghai, China, 2009. IEEE.

[DEKSVB15] Ravin De Souza, Sahar El-Khoury, José Santos-Victor, and Aude Billard. Recognizing the grasp intention from human demonstration. *Robotics and Autonomous Systems*, 74(Part A):108–121, dec 2015.

[DLB+08] Donna C Dupuis, Simon Léonard, Matthew A Baumann, Elizabeth A Croft, and James J Little. Two-Fingered Grasp Planning for Randomized Bin-Picking. In *Proceedings of the Robotics: Science & Systems 2008 Manipulation Workshop - Intelligence in Human Environments*, pages 1–6, 2008.

[DLW00] Dan Ding, Yun-Hui Liu, and Shuguo Wang. Computing 3-D optimal form-closure grasps. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, pages 3573–3578, San Francisco, CA, USA, 2000. IEEE.

[EK07] Staffan Ekvall and Danica Kragic. Learning and evaluation of the approach vector for automatic grasp generation and planning. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4715–4720, Rome, Italy, apr 2007. IEEE.

[EKS09] Sahar El-Khoury and Anis Sahbani. On computing robust n-finger force-closure grasps of 3D objects. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2480–2486, Kobe, Japan, may 2009. IEEE.

[EP13] Lars-Peter Ellekilde and Henrik Gordon Petersen. Motion planning efficient trajectories for industrial bin-picking. *International Journal of Robotics Research*, 32(9-10):991–1004, aug 2013.

[ES14] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014.

[FB81] 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.

[FC92] Carlo Ferrari and John Canny. Planning optimal grasps. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295, Nice, France, 1992. IEEE.

[FV12] David Fischinger and Markus Vincze. Empty the basket - A shape based learning approach for grasping piles of unknown objects. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2051–2057, Vilamoura, Portugal, oct 2012. IEEE.

[GAB16] Germán M García, Nikita Araslanov, and Sven Behnke. Report on Bin-picking (STAMINA). Technical report, Rheinische Friedrich-Wilhelms- Universität Bonn, 2016.

[GD12] A G Ghule and P R Deshmukh. Image Segmentation Available Techniques, Open Issues and Region Growing Algorithm. *Journal of Signal and Image Processing*, 3(1):71–75, 2012.

[GMLB12] Lucian Cosmin Goron, Zoltan-Csaba Marton, Gheorghe Lazea, and Michael Beetz. Robustly Segmenting Cylindrical and Box-like Objects in Cluttered Scenes using Depth Cameras. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6, Munich, Germany, 2012. VDE Verlag.

[HHC$^+$11] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 858–865, Barcelona, Spain, nov 2011. IEEE.

[HSSR05] Robert Haschke, Jochen Jakob Steil, I. Steuwer, and Helge Ritter. Task-Oriented Quality Measures for Dextrous Grasping. In *2005 International Symposium on Computational Intelligence in Robotics and Automation*, pages 689–694, Espoo, Finland, 2005. IEEE.

[IE17] Thomas Fridolin Iversen and Lars-Peter Ellekilde. Benchmarking motion planning algorithms for bin-picking applications. *Industrial Robot*, 44(2):189–197, 2017.

[Int17] International Federation of Robotics. The Impact of Robots on Productivity, Employment and Jobs. Technical Report April, International Federation of Robotics, Frankfurt, Germany, 2017.

[KDPP10] Oliver B Kroemer, Renaud Detry, Justus Piater, and Jan Peters. Combing Active Learning and Reactive Control for Robot Grasping. *Robotics and Autonomous Systems*, 58(9):1105–1116, 2010.

[KH04] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154, Sendai, Japan, 2004. IEEE.

[KK15] Yasuyo Kita and Yoshihiro Kawai. Localization of freely curved pipes for bin picking. In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pages 1–8, Luxembourg, Luxembourg, sep 2015. IEEE.

[KLCT18] Dimitrios Kanoulas, Jinoh Lee, Darwin G. Caldwell, and Nikos G. Tsagarakis. Center-of-mass-based grasp pose adaptation using 3d range and force/torque sensing. *International Journal of Humanoid Robotics*, 15(4), feb 2018.

[KSKG16] Krishnanand N. Kaipa, Shaurya Shriyam, Nithyananda B. Kumbla, and Satyandra K. Gupta. Resolving occlusions through simple extraction motions in robotic bin-picking. In *ASME 2016 11th International Manufacturing Science and Engineering Conference, MSEC 2016*, volume 2, Blacksburg, Virginia, USA, 2016. American Society of Mechanical Engineers.

[KSLB19] Andrew Kimmel, Rahul Shome, Zakary Littlefield, and Kostas Bekris. Fast, Anytime Motion Planning for Prehensile Manipulation in Clutter. In *IEEE-RAS International Conference on Humanoid Robots*, pages 874–880, Beijing, China, jun 2019. IEEE.

[KSLO96] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[LaV98] Steven M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. Technical report, Department of Computer Science, Iowa State University, Ames, Iowa, USA, 1998.

[LCA14] Yinxiao Li, Chih-Fan Chen, and Peter K. Allen. Recognition of deformable object category and pose. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5558–5564, Hong Kong, China, may 2014. IEEE.

[LCSV19] Gonçalo Leão, Carlos M. Costa, Armando Sousa, and Germano Veiga. Perception of Entangled Tubes for Automated Bin Picking. In *ROBOT 2019: Fourth Iberian Robotics Conference*, pages 619–631, Porto, Portugal, 2019. Springer.

[LCSV20] Gonçalo Leão, Carlos M. Costa, Armando Sousa, and Germano Veiga. Detecting and Solving Tube Entanglement in Bin Picking Operations. *Applied Sciences*, 10(7):2264, mar 2020.

[LLC03] Jia-Wei Li, Hong Liu, and He-Gao Cai. On computing three-finger force-closure grasps of 2-D and 3-D objects. *IEEE Transactions on Robotics and Automation*, 19(1):155–161, feb 2003.

[LLD04] Yun-Hui Liu, Miu-Ling Lam, and Dan Ding. A complete and efficient algorithm for searching 3-D form-closure grasps in the discrete domain. *IEEE Transactions on Robotics*, 20(5):805–816, oct 2004.

[LMSB14] Beatriz León, Antonio Morales, and Joaquín Sancho-Bru. From robot to human grasping simulation. *Cognitive Systems Monographs*, 19:1–261, 2014.

[LR09] Yu-Shen Liu and Karthik Ramani. Robust principal axes determination for point-based shapes using least median of squares. *Computer-Aided Design*, 41(4):293–305, apr 2009.

[LS13] Wen Hao Lui and Ashutosh Saxena. Tangled: Learning to untangle ropes with RGB-D perception. In *IEEE International Conference on Intelligent Robots and Systems*, pages 837–844, Tokyo, Japan, nov 2013. IEEE.

[LUD$^+$10] Beatriz León, Stefan Ulbrich, Rosen Diankov, Gustavo Puche, Markus Przybylski, Antonio Morales, Tamim Asfour, Sami Moisio, Jeannette Bohg, James Kuffner, and Rüdiger Dillmann. OpenGRASP: A Toolkit for Robot Grasping Simulation. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, volume 6472, pages 109–120. Springer, Darmstadt, Germany, 2010.

[MA04] Andrew T. Miller and Peter K. Allen. Graspit: A versatile simulator for robotic grasping. *IEEE Robotics and Automation Magazine*, 11(4):110–122, dec 2004.

[MAT12] Pol Monsó, Guillem Alenyà, and Carme Torras. POMDP approach to robotized clothes separation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1324–1329, Vilamoura, Portugal, oct 2012. IEEE.

[MDWH19] Ryo Matsumura, Yukiyasu Domae, Weiwei Wan, and Kensuke Harada. Learning Based Robotic Bin-picking for Potentially Tangled Objects. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7990–7997, Macau, China, nov 2019. IEEE.

[MHDW18] Ryo Matsumura, Kensuke Harada, Yukiyasu Domae, and Weiwei Wan. Learning based industrial bin-picking trained with approximate physics simulator. In *International Conference on Intelligent Autonomous Systems*, volume 867, pages 786–798, Singapore, Singapore, 2018.

[MK15] Chintan Mishra and Zeeshan Ahmed Khan. *Development and Evaluation of a Kinect based Bin-Picking System.* Master thesis, Mälardalen University, 2015.

[MKCA03] Andrew T. Miller, Steffen Knoop, Henrik I. Christensen, and Peter K. Allen. Automatic grasp planning using shape primitives. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, pages 1824–1829, Taipei, Taiwan, 2003. IEEE.

[MLN$^+$17] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, USA, 2017.

[MLS94] Richard Murray, Zexiang Li, and Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*, volume 29. CRC Press, 1994.

[MRP$^+$16] Eduardo Moreira, Luís Freitas Rocha, Andry Maykol Pinto, António Paulo Moreira, and Germano Veiga. Assessment of Robotic Picking Operations Using a 6 Axis Force/Torque Sensor. *IEEE Robotics and Automation Letters*, 1(2):768–775, jul 2016.

[Ngu87] Van Duc Nguyen. Constructing stable grasps in 3D, 1987.

[O'K14] Jason Matthew O'Kane. *A Gentle Introduction to ROS*. Jason Matthew O'Kane, 2014.

[PBC13] Leonardo M. Pedro, Valdinei L. Belini, and Glauco A.P. Caurin. Learning how to grasp based on neural network retraining. *Advanced Robotics*, 27(10):785–797, jul 2013.

[PMAJ04] Raphael Pelossof, Andrew Miller, Peter Allen, and Tony Jebara. An SVM learning approach to robotic grasping. In *IEEE International Conference on Robotics and Automation, 2004*, pages 3512–3518 Vol.4, New Orleans, LA, USA, 2004. IEEE.

[QGC$^+$09] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[QZN14] Rongqi Qiu, Qian-Yi Zhou, and Ulrich Neumann. Pipe-run extraction and reconstruction from point clouds. In *Computer Vision - ECCV 2014. 13th European Conference. Proceedings: LNCS 8691*, volume 8691, pages 17–30, Zurich, Switzerland, 2014. Springer International Publishing.

[RAMNT12] Arnau Ramisa, Guillem Alenyà, Francesc Moreno-Noguer, and Carme Torras. Using depth and appearance features for informed robot grasping of highly wrinkled clothes. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1703–1708, Saint Paul, MN, USA, may 2012. IEEE.

[RBM$^+$07] Radu Bogdan Rusu, Nico Blodow, Zoltan Marton, Alina Soos, and Michael Beetz. Towards 3D object maps for autonomous household robots. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3191–3198, San Diego, CA, USA, oct 2007. IEEE.

[RC11] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation*, pages 1–4, Shanghai, China, 2011. IEEE.

[RK96] Krisnawan Rahardja and Akio Kosaka. Vision-based bin-picking: recognition and localization of multiple complex objects using simple visual cues. In *1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1448–1457, Osaka, Japan, 1996. IEEE.

[RKK09] Javier Romero, Hedvig Kjellström, and Danica Kragic. Modeling and Evaluation of Human-to-Robot Mapping of Grasps. In *2009 International Conference on Advanced Robotics*, pages 1–6, Munich, Germany, 2009. IEEE.

[RLP$^+$10] Deepak Rao, Quoc V Le, Thanathorn Phoka, Morgan Quigley, Attawith Sudsang, and Andrew Y Ng. Grasping novel objects with depth segmentation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2578–2585, Taipei, Taiwan, oct 2010. IEEE.

[Rob06] Adam Roberts. *The History of Science Fiction*. Palgrave Macmillan UK, 1 edition, 2006.

[RS15] Máximo A. Roa and Raúl Suárez. Grasp quality measures: review and performance. *Autonomous Robots*, 38(1):65–88, jan 2015.

[RT12] Lyle Ramshaw and Robert E Tarjan. On Minimum-Cost Assignments in Unbalanced Bipartite Graphs. Technical report, HP Laboratories, 2012.

[Rus10] Radu Bogdan Rusu. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - Künstliche Intelligenz*, 24(4):345–348, nov 2010.

[RV08] Mario Richtsfeld and Markus Vincze. Grasping of unknown objects from a table top. *Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environmemnts*, 2008.

[RvdHV06] Tahir Rabbani, Frank van den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. In Harrie G. R. Maas and Daniel Schneider, editors, *ISPRS 2006 : Proceedings of the ISPRS commission V symposium*, volume 35, pages 248–253, Dresden, Germany, 2006. International Society for Photogrammetry and Remote Sensing (ISPRS).

[SDH$^+$14] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, aug 2014.

[SDN08] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic Grasping of Novel Objects using Vision. *International Journal of Robotics Research*, 27(2):157–173, 2008.

[SEKB12] Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. An overview of 3D object grasp synthesis algorithms. *Robotics and Autonomous Systems*, 60(3):326–336, 2012.

[Shi96] Karun B. Shimoga. Robot Grasp Synthesis Algorithms: A Survey. *International Journal of Robotics Research*, 15(3):230–266, jun 1996.

[SL09] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437, jul 2009.

[STD09] Giovanna Sansoni, Marco Trebeschi, and Franco Docchio. State-of-The-Art and Applications of 3D Imaging Sensors in Industry, Cultural Heritage, Medicine, and Criminal Investigation. *Sensors*, 9(1):568–601, jan 2009.

[STK$^+$11] Freek Stulp, Evangelos Theodorou, Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, and Stefan Schaal. Learning motion primitive goals for robust manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 325–331, San Francisco, CA, USA, sep 2011. IEEE.

[SWJ17] Kai-Tai Song, Cheng-Hei Wu, and Sin-Yi Jiang. CAD-based Pose Estimation Design for Random Bin Picking using a RGB-D Camera. *Journal of Intelligent and Robotic Systems*, 87(3-4):455–470, 2017.

[TK03] Geoffrey Taylor and Lindsay Kleeman. Robust Range Data Segmentation Using Geometric Primitives for Robotic Applications. In *Proceedings of the Fifth IASTED International Conference on Signal and Image Processing*, pages 467–472, Honolulu, HI, USA, 2003. Acta Press.

[tPP15] Andreas ten Pas and Robert Platt. Using Geometry to Detect Grasps in 3D Point Clouds. In *International Symposium on Robotics Research (ISRR)*, pages 1–16, Sestri Levante, Italy, 2015.

[VVS17] Vojtěch Vonásek, Axel Vick, and Martin Saska. Motion planning with motion primitives for industrial bin picking. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, Limassol, Cyprus, jun 2017. IEEE.

[WCK$^+$94] Sarah Wang, Robert Cromwell, Avi Kak, Ichiro Kimura, and Michiharu Osada. Model-based vision for robotic manipulation of twisted tubular parts: using affine transforms and heuristic search. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 208–215, San Diego, CA, USA, 1994. IEEE.

[WL97] Kenong Wu and Martin D Levine. 3-D shape approximation using parametric geons. *Image and Vision Computing*, 15(2):143–158, 1997.

[WLCL09] Chee Kit Wong, Patrick P K Lim, Roger Clist, and Raymond ShuHe Liu. Vision Strategies for Robotic Manipulation of Natural Objects. In *Australasian Conference on Robotics and Automation*, Sydney, Australia, 2009.

[YHH$^+$09] Yusuke Yoshida, Jun'ichiro Hayashi, Seiji Hata, Hirotaka Hojoh, and Toshihiro Hamada. Status estimation of cloth handling robot using force sensor. In *IEEE International Symposium on Industrial Electronics*, pages 339–343, Seoul, South Korea, 2009. IEEE.

[YK86] H Yang and Avinash Kak. Determination of the identity, position and orientation of the topmost object in a pile: some further experiments. In *Proceedings 1986 IEEE International Conference on Robotics and Automation*, pages 293–298, San Francisco, CA, USA, 1986. IEEE Computer Society Press.

[YK91] Yoshimasa Yanagihara and Toshiro Kita. Parts-picking in disordered environment. In *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, volume 2, pages 517–522, Osaka, Japan, 1991. IEEE.

[ZD04] Xiangyang Zhu and Han Ding. Planning force-closure grasps on 3-D objects. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pages 1258–1263, New Orleans, LA, USA, 2004. IEEE.

[Zha92] Zhengyou Zhang. Iterative point matching for registration of free-form curves. Technical report, INRIA, 1992.

[ZLZ$^+$16] Hao Zhang, Pinxin Long, Dandan Zhou, Zhongfeng Qian, Zheng Wang, Weiwei Wan, Dinesh Manocha, Chonhyon Park, Tommy Hu, Chao Cao, Yibo Chen, Marco Chow, and Jia Pan. DoraPicker: An autonomous picking system for general objects. In *IEEE International Conference on Automation Science and Engineering*, pages 721–726, Fort Worth, TX, USA, aug 2016. IEEE.

[ZW03] Xiangyang Zhu and Jun Wang. Synthesis of force-closure grasps on 3-D objects based on the Q distance. *IEEE Transactions on Robotics and Automation*, 19(4):669–679, aug 2003.