BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ÚSTAV AUTOMATIZACE A INFORMATIKY

3D POINT CLOUD SEGMENTATION FOR INDUSTRIAL BIN-PICKING

SEGMENTACE 3D MRAČNA BODŮ PRO PRŮMYSLOVÉ VYBÍRÁNÍ KOŠŮ

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR	Ing. Marek Šooš

AUTOR PRÁCE

SUPERVISOR	Ing. Mhd Ali Shehadeh

VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Master's Thesis

Institut:	Institute of Automation and Computer Science
Student:	**Ing. Marek Šooš**
Degree programm:	Applied Computer Science and Control
Branch:	no specialisation
Supervisor:	**Ing. Mhd Ali Shehadeh**
Academic year:	2022/23

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

3D point cloud segmentation for industrial bin–picking

Brief Description:

Segmentation is a fundamental task in 3D point cloud processing. Given a point cloud, the goal of segmentation is to cluster points with similar patterns into one category. The segmentation process further helps with the location, recognition, and classification of objects.

Master's Thesis goals:

1. Review of robotic 3D vision approaches for bin picking.
2. A brief introduction to 3D point clouds acquisition and processing.
3. Development of a 3D point cloud instance segmentation application for the bin–picking scene.
4. Application for segmentation on several selected objects.

Recommended bibliography:

XU, Y.; ARAI, S.; LIU, D.; LIN, F.; KOSUGE, K. FPCC: Fast point cloud clustering-based instance segmentation for industrial bin-picking. Neurocomputing, 2022, vol. 494, p. 255-268. ISSN: 0925-2312.

LIU, S.; ZHANG, M.; KADAM, P.; KUO, C. C. J. 3D point cloud analysis: Traditional, deep learning, and explainable machine learning methods. Springer International Publishing AG, 2021. ISBN 978-3-030-89179-4.

LI, D.; WANG, H.; LIU, N.; WANG, X.; XU, J. 3D Object Recognition and Pose Estimation From Point Cloud Using Stably Observed Point Pair Feature. IEEE Access, 2020, vol. 8, pp. 44335-44345. ISSN: 2169-3536

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2022/23

In Brno,

L. S.

doc. Ing. Radomil Matoušek, Ph.D. Director of the Institute	doc. Ing. Jiří Hlinka, Ph.D. FME dean

ABSTRACT

This thesis deals with 3D point cloud segmentation for industrial bin-picking, a key challenge in the field of industrial robotics. The aim of the thesis is to develop and deploy a highly effective algorithm for segmenting and registering 3D point clouds, thereby improving the accuracy, speed, and efficiency of bin-picking operations.

The contribution of the thesis is the presentation of the researcher's solution to create artificially generated data needed for training. The thesis results in a symbiosis of advantages of a fast-segmentation algorithm based on machine learning, and a high quality, robust but slow algorithm based on geometric principles. Functionality, reliability and quality of the developed algorithm were also experimentally verified on different types of objects and different datasets.

Results of the work show that the proposed algorithm yields a fast, reliable, and comprehensive solution to the bin-picking problem. Customized data generation reduces the time and cost of applying such a system. In conjunction with a comprehensive problem solving system we are able to accurately and easily generate applications for diverse and specialized bin-picking tasks.

Achieved results contribute to the development of point cloud segmentation methods and their applications in various industrial and scientific fields. By putting the proposed system into practice we significantly contribute to performance and reliability of the proposed automatic line.

ABSTRAKT

Diplomová práca sa zaoberá segmentáciou 3D mračna bodov pre priemyselné výbery zo zásobníkov, čo je kľúčová výzva v oblasti priemyselnej robotiky. Cieľom práce je navrhnúť a implementovať efektívny algoritmus na segmentáciu a registráciu 3D bodov mračien, a tým vylepšiť presnosť, rýchlosť a efektívnosť bin-picking operácií.

Prínosom diplomovej práce je prezentácia autorovho vlastného riešenia vytvárania umelo generovaných dát potrebných na trénovanie. Výsledkom práce je symbióza výhod rýchleho segmentáčného algoritmu založeného na strojovom učení, a kvalitného, robustného ale pomalého registračného algoritmu založeného na geometrickom princípe. Funkčnosť, spoľahlivosť a kvalitu vytvoreného algoritmu boli verifikované aj experimentálne na rôznych typoch objektov a rôznych datasetoch.

Výsledky práce ukazujú, že navrhnutý algoritmus prináša rýchle, spoľahlivé, a komplexné riešenie problému bin-picking. Generovanie dát na mieru znižuje čas a náklady na aplikáciu takéhoto systému. V spojení s komplexným systémom riešenia problému, je možné jednoducho vytvárať riešenia pre rozmanite a špecializovane úlohy bin-pickingu.

Dosiahnutými výsledkami prispieva k rozvoju metód segmentácie bodových mračien a ich aplikácií v rôznych priemyselných a vedeckých oblastiach. Zavedením navrhnutého systému do praxe, výrazne prispeje k zvýšeniu výkonnosti a spoľahlivosti navrhovanej automatickej linky.

KEYWORDS

3D segmentation, 3D point cloud, deep learning, 3D registration, 3D dataset generation, bin-picking

KĽÚČOVÉ SLOVÁ

3D segmentácia, 3D mračná bodov, hlboké učenie, 3D registrácia, generovanie 3D dátového setu, bin-picking

INSTITUTE OF AUTOMATION
AND COMPUTER SCIENCE

2023

BIBLIOGRAPHIC CITATION

ŠOOŠ, Marek. *Title of Student's Thesis.* Brno, 2023. Available at: `https://www.vut.cz/studenti/zav-prace/detail/149277`. Master's Thesis. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, Supervised by Ing. Mhd Ali Shehadeh

AUTHOR'S DECLARATION

I declare that this thesis is my original work, I have prepared it independently under the supervision of the thesis supervisor and using professional literature and other sources of information, all of which are cited in the thesis and listed in the reference list.

As the author of this thesis, I further declare that in connection with the creation of this thesis I have not infringed the copyright of third parties, in particular I have not infringed in an unauthorised manner on the personal copyright of others, and I am fully aware of the consequences of violating the provisions of § 11 et seq. of the Copyright Act No. 121/2000 Coll., including possible criminal consequences.

V Brně dne 26. 5. 2023 . . . .

Marek Šooš

ACKNOWLEDGEMENT

On this occasion, I would like to thank my supervisor, Ing. Mhd Ali Shehadeh, for his valuable advice and comments during the course of my thesis. Furthermore, I would also like to thank my parents who have supported me throughout my study years. And last but not least, I would like to thank my loving wife who not only tolerates, but even supports me in my study escapades.

CONTENTS

1 Introduction

Each consumer wants to buy their product at the lowest possible price. In order to keep the price low, products are often manufactured in large batches on automatic lines. The design process of an automatic line involves many design nodes. One of them, for example, is the way of filling the line with input blanks. This thesis deals with the automation of the bin-picking process.

A key feature of the process is efficient sorting of objects based on their properties. Traditionally, this task has been performed manually, which in the long run was time-consuming and economically intensive with a high potential error rate. With the development of 3D sensing capabilities there are opening up new opportunities for automating this process.

In addressing the problem of automation of bin-picking method, this thesis focuses on research and analysis of existing segmentation algorithms, implementation of the solution, and application to a specific problem of bin-picking using industrial cameras.

The theoretical foundations of 3D point cloud and computer and machine vision techniques relevant to this problem are also discussed in the theoretical part of the thesis. This section also offers an overview of the technological possibilities of 3D image acquisition and the analysis of current 3D point cloud segmentation methods.

The next part of the thesis describes proposed solution in detail, including the selection of algorithm, the solution architecture, the implementation of the proposed solution, and the methods to address the challenges associated with this problem. The following part is devoted to the evaluation and presentation of results of the presented algorithms based on the experiments performed on both training data and real data.

Finally, the thesis is devoted to the evaluation of achieved results, suggesting possibilities for further research and development in this area.

Afterall, should the thesis contribute to the development of automated bin-picking in industrial processes and provide useful insights for researchers and specialists in the field of computer vision and machine learning. The ultimate goal is to design a solution that will increase the accuracy and reliability of identification, reduce cycle time, and save money relative to currently used solutions in automated lines. And also to eliminate human presence, and its associated disadvantages, such as high cost and higher error rate.

2 Literature Review

2.1 Computer, Machine, and Robot Vision

Computer vision, a branch of computer science, is the development of techniques that enable computers to recognise and understand objects and people in images and video. As a subset of artificial intelligence (AI), computer vision aims to replicate human capabilities by automating tasks related to visual perception and understanding, [40] .

Machine vision and computer vision share the ability to perform tasks faster than human vision. However, there are important differences between the two concepts.

Computer vision, involves the capture and automation of image analysis. It is the broader field of computer understanding and interpreting visual data. Computer vision techniques can be applied to a wide range of theoretical and practical areas beyond manufacturing. Its applications span various industries and include tasks such as recognising objects, classifying images and understanding scenes.

Machine vision, on the other hand, is primarily about adding vision capabilities to existing technologies. It involves image processing techniques and operates on the basis of pre-defined rules and parameters. Machine vision systems are commonly used in manufacturing applications, particularly for tasks such as quality assurance.

In summary, while computer vision encompasses a broader range of image analysis and interpretation in a variety of applications, machine vision is a subset of computer vision that focuses specifically on providing vision capabilities to support manufacturing processes, [41] .

Robot vision is closely related to machine vision, while both are closely related to computer vision. In a sense of a family tree, displayed in Figure 1, computer vision could be seen as their parent.

Robot vision encompasses techniques derived from all of the above. While Robot Vision and Machine Vision are often used interchangeably, there are subtle differences between them. Machine vision applications, such as part inspection, do not necessarily involve robotics directly.

Robot Vision is not just an engineering domain - it is a scientific discipline with its own specialised areas of research. Unlike pure computer vision research, robot vision requires the incorporation of robotic aspects into its techniques and algorithms. These aspects include kinematics, reference frame calibration, and the robot's ability to physically interact with the environment.

In summary, Robot Vision draws on techniques from various fields and is not limited to engineering. The use of visual feedback to control robot motion, as exemplified by visual servoing, is an example of a technique that is unique to Robot Vision rather than Computer Vision, [21], [16] .

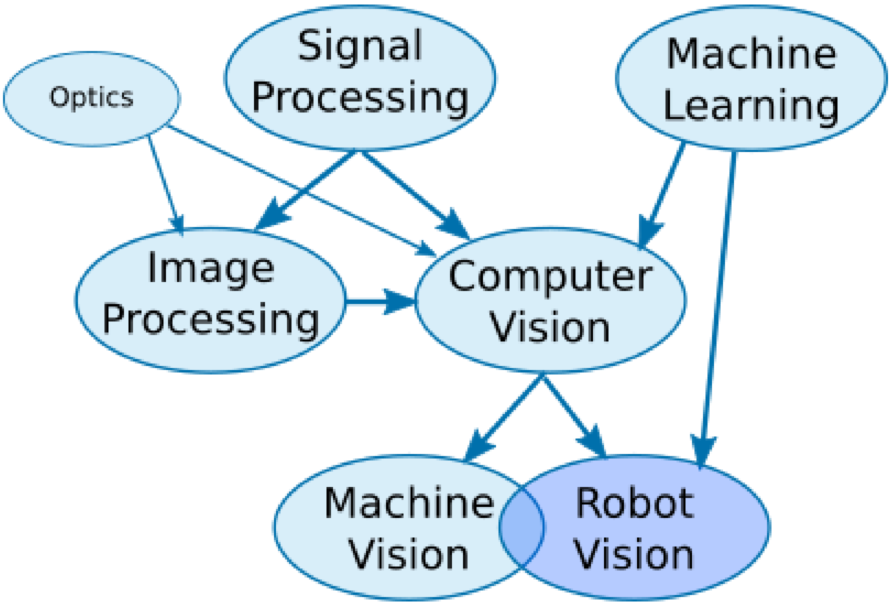


Fig. 1: Vision Family Tree, [21]

2.2 Overview of robot picking

In a modern automation processes, plenty of industrial applications require robot arm path planning using a vision system.

Robot picking is an application used in manufacturing industry, which parts to pick from a certain box, and pick them automatically. It's a well combined system, consisting of a robotic arm and a well designed and programmed machine vision system.

It mainly brings advantages to the manufacturing companies, who can use this system for the infeed of manufactured parts to the assembly line. The vision system analyses the data acquired from camera. The system then decides which part is the most suitable one to be picked first and defines its position. The results are sent to the PLC, which the instructs the robotic arm to pick a part from the box. The vision system should not only be able to find a part, but also calculate the correct position. This is essential, when the parts are chaotically arranged in the input box, [47], [51] .

We divide the processes according to whether they the position and orientations are known or not into two groups:

- Pick and place
- Bin-picking

2.2.1 Pick and place

A pick and place robot is a type of industrial robot used to handle and place products on a production line. Typically used in high volume production environments to quickly and precisely place items onto conveyors or other production devices. Compared to industrial bin-picking tasks, pick and place operations are generally simpler. Unlike bin-picking, pick and place operations typically involve objects in a 2D environment rather than a more complex 3D environment. Pick and place robots are automation solutions that pick up and accurately place objects onto surfaces in predetermined positions and orientations. The robot follows a pre-defined routine, which is staged in advance, making it less challenging to execute. The parts involved are not randomly oriented and the process is highly repeatable. From picking up the first part to placing the first and subsequent parts, the robot's operations remain consistent and do not deviate. A representation for chaotically arranged items and objects in a predefined grid is shown in Figure 2 below, [24], [15], [49] .

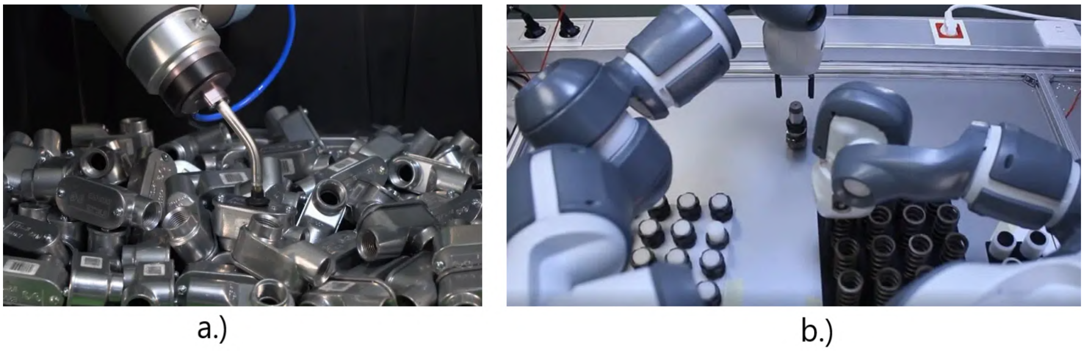


Fig. 2: a.) Items chaotically arranged. b.) Objects in a predefined grid., [51]

2.2.2 Bin-picking

Bin-picking, displayed in Figure 3, is an application that uses a combination of robotics and computer vision technology to extract parts from a disorganised container. Using a robotic arm controlled by a computer vision system, this technology enables the automated extraction of parts. The computer vision system analyses the scene, identifies part locations and determines the optimal part for extraction at each iteration. Once the positions of the parts have been determined in a 3D space, the robotic arm is able to pick them up efficiently, regardless of their orientation, [47] .

In general, the bin-picking process requires all these steps :

- Data acquisition
- Object detection
- Estimating the position and orientation in space

- Collision-free path-planing
- Object Picking
- Placing the object at the right place in the right orientation

During all these processes, most of the tasks should be done error-free to provide a solid solution. Considering the data acquisition, whole slew of methods or principles can be applied to get 3D image of the space. Nowadays, 3D laser scanners, laser triangulation using a single camera with a 2D laser or stereo vision are used to capture a 3D image of the environment.

These methods have to deal with numbers of issues such as finding a suitable position for the end-effector, finding the right gripping position or orientation etc. Another key element in industrial bin-picking is the cycle time for a specific application.

All of these tasks must be performed without damaging the object, box or any other obstacles. They are carried out completely autonomously and based only on the system's decisions.

Bin-picking also opens the possibilities to pick heterogeneous objects. These objects require the design of special grippers or end effectors. Therefore, bin-picking should be able to reliably recognise any object, calculate its orientation, pick it and place it in the box regardless of its material or geometry.

While gripper design, robot path programming to avoid collisions and robot singularity avoidance are essential elements of bin-picking, the vision system is widely recognised as the most critical factor in meeting this challenge. Industries require robust and reliable systems to meet their needs. Although there are existing bin-picking applications in operation, continuous research is dedicated to creating universal systems that are free from failure, [47], [51] .

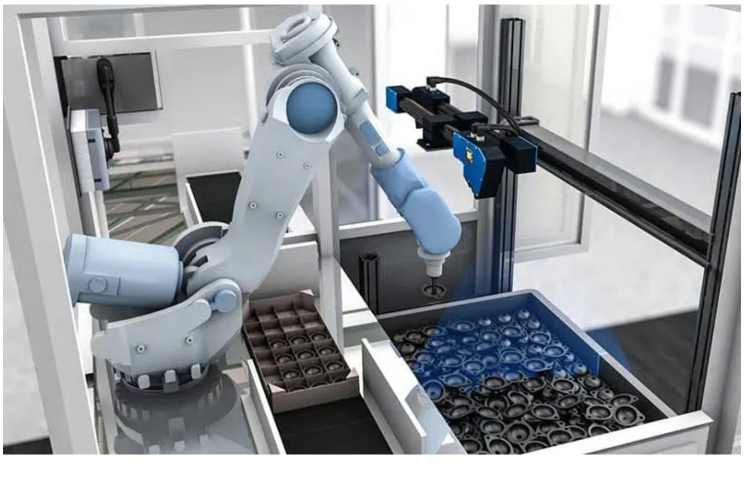


Fig. 3: Process of bin-picking, [47]

2.3 Scene reconstruction

Scene reconstruction is a process of capturing one or more pictures in order to get the information about an environment. These images are merged according to the application in such a way that they create a 3D image of the environment of the photographed surroundings.

2.3.1 2D Image Camera

The first solution that comes on mind for a picking application, uses some classical 2D RGB cameras, either in single or stereo configuration, displayed in Figure 4. These solutions are using algorithms to extract features from the acquired images in order to recognise objects and estimate their location and rotation in the coordinate space.

Algorithms for single cameras mostly rely on the object's characteristic such as color or overall structure. Due to the fact, that numerous industrial objects often contain some circular shapes, plenty of algorithms are focussing on detection of ellipses. A single camera, if calibrated and its sensor size and lens parameters are known, can determine the position and orientation of an object. However, this type of machine vision configuration cannot reliably estimate rotation and translation around each axis. For this reason, single camera systems are primarily used in pick and place applications, [17] .

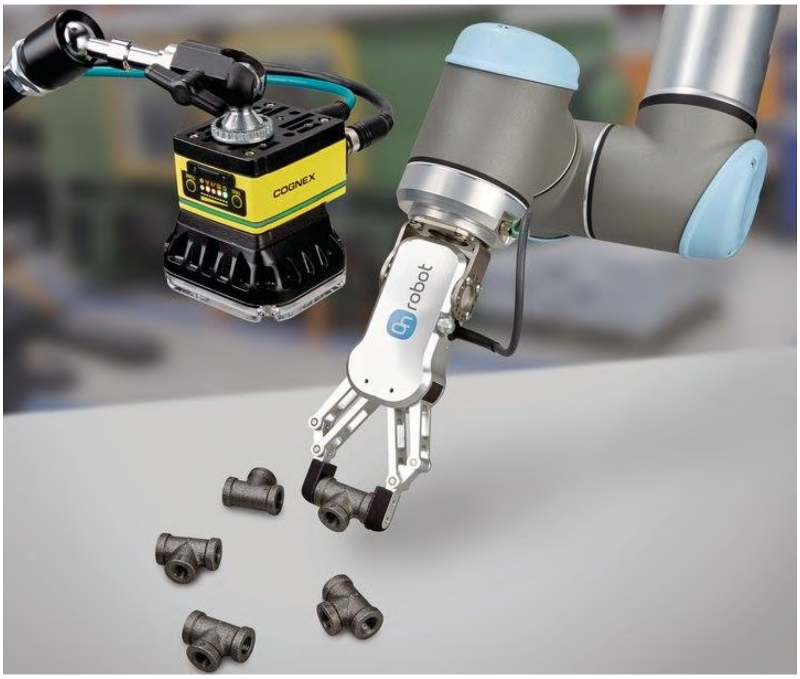


Fig. 4: Bin-picking with 2D camera, [52]

The method described above depends on the knowledge of a pickable object before designing the algorithm. Other algorithms are based on edge detection to obtain an image description. Although these methods can produce results with good accuracy, they do not provide sufficient information about the whole stage to be used for collision detection of a robotic arm. This is one of the reasons why research today focuses on registering objects in a point cloud, [17] .

2.3.2 3D Camera

Point clouds can be considered the most basic form of 3D models. They consist of individual points that are plotted in three-dimensional space and contain various measurements. These measurements typically include the coordinates of the point along the X, Y and Z axes. In addition, point clouds can contain other data such as colour information, stored in RGB format, and luminance values, which determine the brightness of each point, [3], [17] .

The 3D point cloud of an environment can be acquired by several different sensors and techniques. This section will briefly review the most common ways to collect 3D data of a scanned environment for reconstruction purposes. Common point cloud reconstructing methods are :

- 3D Laser Scanning
- Photogrammetry
- Videogrammetry
- RGB-D camera
- Stereo camera
- Structured Light

3D Laser Scanning

Light Detection And Ranging (Lidar), uses a laser scanner displayed in Figure 5, to measure the distance to a target by emitting laser beams and detecting the reflected signals from the target. There are two main techniques for laser distance measurement: time-of-flight and phase-shift. The time-of-flight technique uses a laser beam and measures the time it takes to travel to the object and reflect back to the detector. From the known speed of light, the distance can be calculated, equation 1, [30], [17], [18], [23] .

$$d = c * \frac{t}{2} \tag{1}$$

Where

d $[m]$ distance to the target
c $[\frac{m}{s}]$ speed of light
t $[s]$ time of flight

On the other hand, the technique called phase-shift, emits a continuous amplitude wave and measures the phase shift between the emitted and reflected signals. The resulting distance is calculated on the basis of the phase-shift and the wavelength of the emitted wave, equation 2. The accuracy of the phase-shift techniques is greater than the speed scanners measurement using the time-of-flight principles. On the other hand,time-of-flight scanners are the only applicable option for scanning longer distances, [30], [17], [18], [23] .

$$d = \frac{c \times \Delta\phi}{2\pi \times f} \tag{2}$$

Where

d $[m]$ distance to the target
c $[\frac{m}{s}]$ speed of light
$\Delta\phi$ $[rad]$ phase difference
f $[Hz]$ frequency

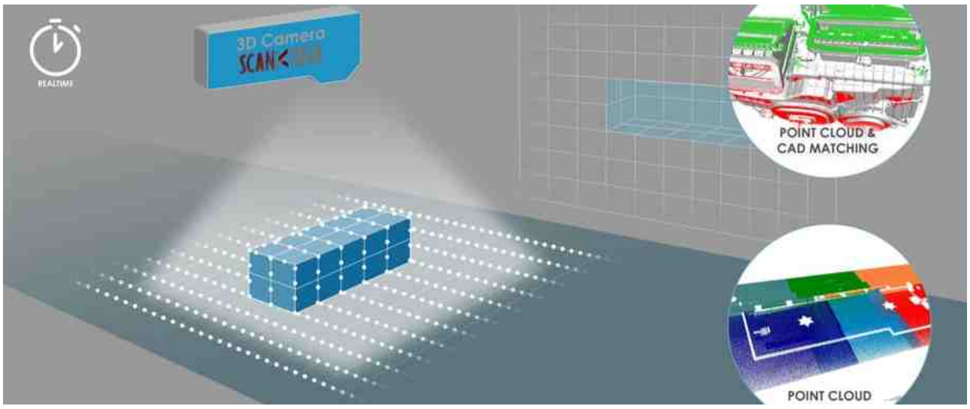


Fig. 5: 3D laser scanning [50]

Photogrammetry

Photogrammetry is a field that analyses photographic images to extract decisive information about physical objects. The principle of creating a point cloud, shown in Figure 6, involves capturing a group of overlapping images of an object from various angles, [30], [29] .

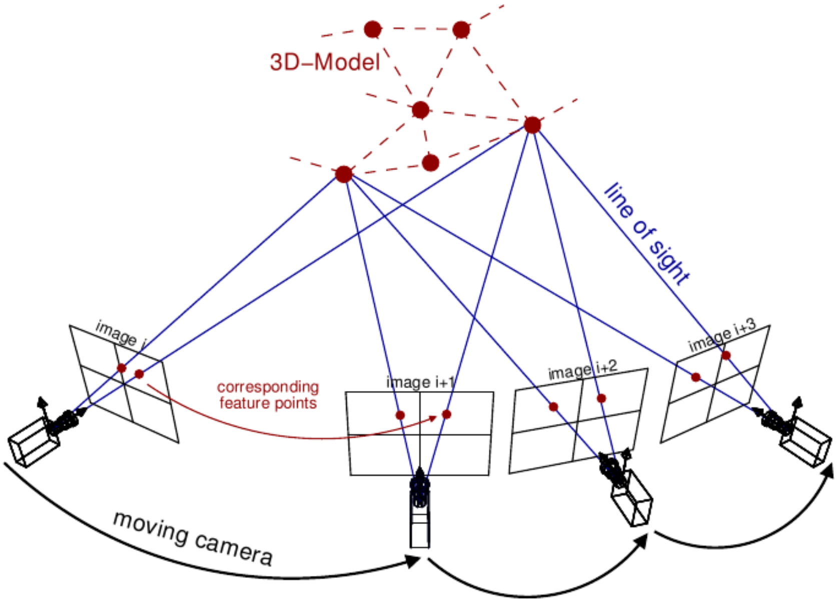


Fig. 6: Principles of photogrammetry, videogrammetry, [27]

Videogrammetry

Videogrammetry is a technique that enables the reconstruction of a point cloud by utilizing sequential video frames to progressively build up information and improve the accuracy of the final point cloud. This is achieved by tracking the features of interest between successive frames of the video, displayed in Figure 7 below, [30] [29] .

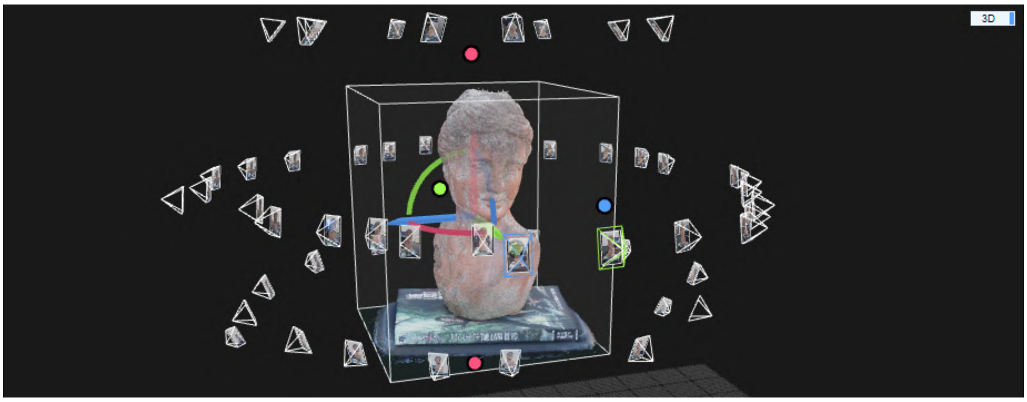


Fig. 7: Application of photogrammetry, [14]

RGB-D Camera

The RGB-D camera, visible in Figure 8, is made up of regular RGB camera and a depth sensor. The RGB camera captures classic images in red, green and blue color. The depth sensor provides depth information for each pixel. The final point cloud is composed right after the capture from both image data with RGB color information and depth data, which include XYZ coordinates. One of the most available RGB-D cameras, often used in research studies is the Microsoft Kinect, [30], [18] .

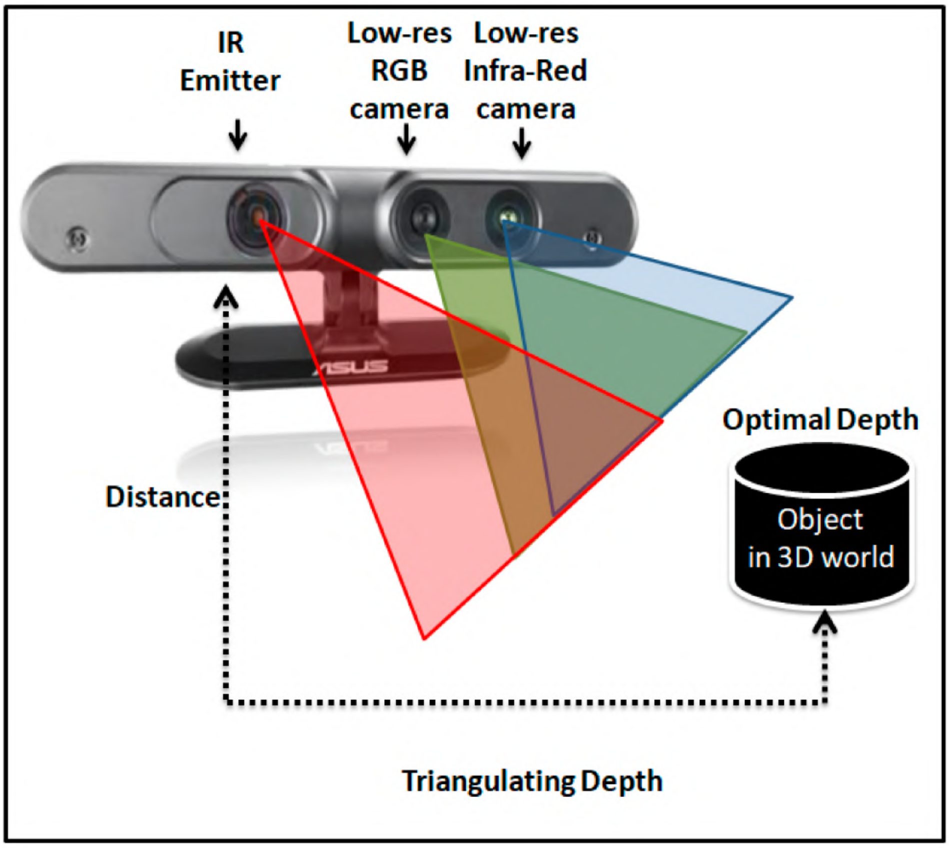


Fig. 8: Asus RGB-D Camera, [26]

Stereo Camera

A stereo camera, exposed in Figure 9, is a system consisting of two (or more) lenses and image sensors. By utilizing this camera configuration, it becomes possible to reconstruct a 3D point cloud from two or more images. This is achieved by analyzing the relative position and orientation of the lenses involved in capturing the images. The final point cloud is the result of image alignment and subsequent determination of disparities for each pixel. The system uses a fully automated pre-calibration process, which extracts a 3D point cloud from the acquired images, [30].

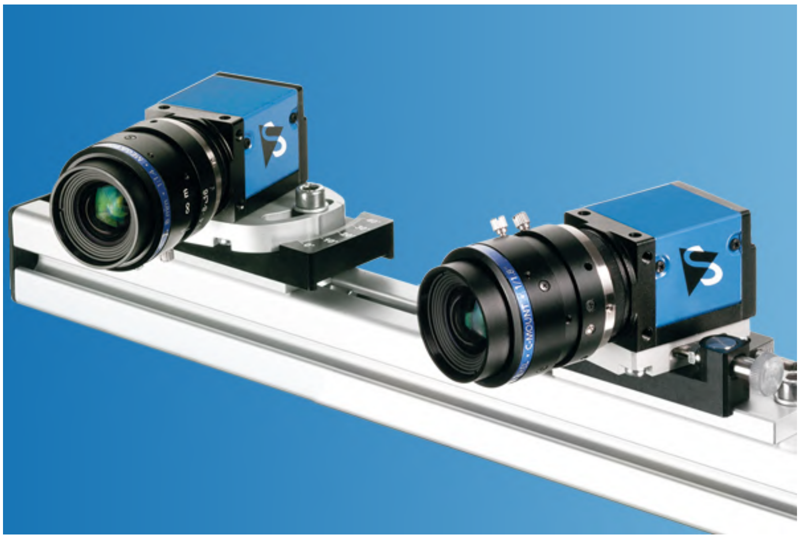


Fig. 9: Stereo camera, [44]

Structured light

Structured light is a commonly used technique to estimate a 3D representation of a scene. It is similar to a stereo camera, but instead of two cameras, it uses one camera in conjunction with a Digital Light Processing projector.

While stereo cameras might struggle when dealing with objects lacking texture, digital light processing projector solves the problem by projecting a known patter onto the surface of the scene, effectively providing texture even in areas where there is no texture.

In order to compute a depth map, the illuminated point that corresponds to a pixel of the projector in the captured image is determined. Finding the correspondence along the edges of the projected patterns is relatively straightforward, so patterns with many edges are commonly used. Where projector and camera are not rectified, random dots are typically used as projected patterns. If the system is rectified, vertical binary lines are often used. The captured image is then subjected to threshold to distinguish between the illuminated and unilluminated parts of the scene, followed by matching the illuminated parts to the projected pattern.

However, the flashing nature of the projected pattern can be irritating to humans, which is why this type of sensor is often used in closed environments. Figure 10, represents the Keyence 3D solution based on structured light, [17] .

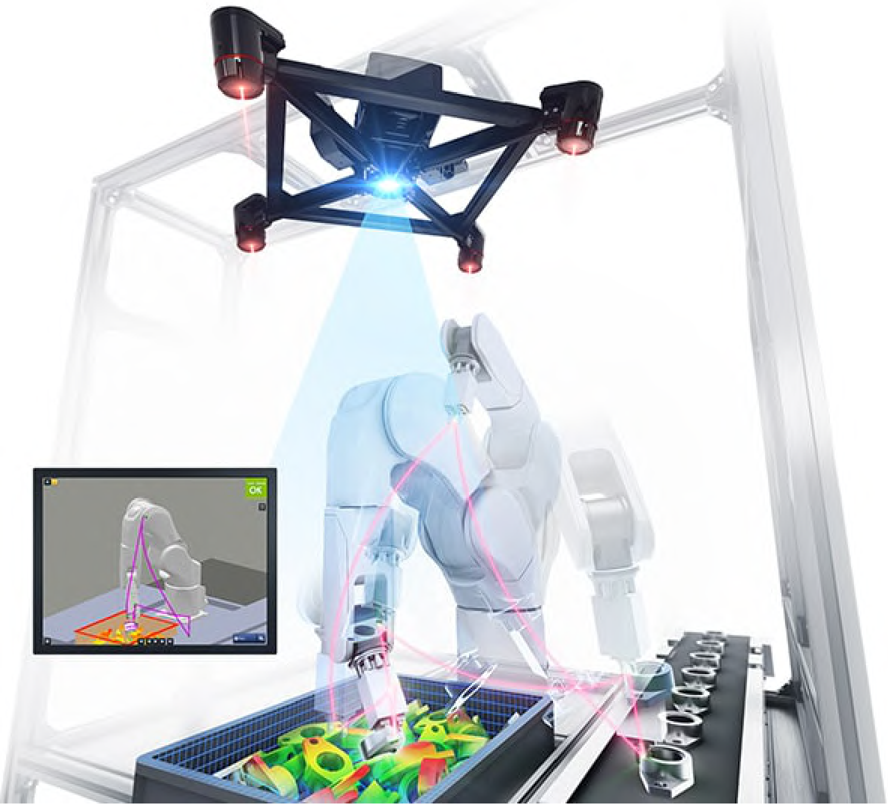


Fig. 10: Keyence 3D bin-picking camera, [43]

2.4 Computer Vision software

2.4.1 Segmentation techniques

Data segmentation is a process, which assigns the same labels to points that belong to the same region. Additionally, points with similar features within a continuous region are grouped together to generate a segment. The process of segmentation can be seen in Figure 11, each object is marked with the corresponding color. Hundreds of computational techniques and principles for point cloud segmentation have been proposed over the last few decades. The well-known segmentation methods could be categorised according to their main segmentation mechanisms. Six of the most common categories will be introduced here, which are categorized based on their main segmentation mechanisms, along with their main advantages and disadvantages, [30], [12] .

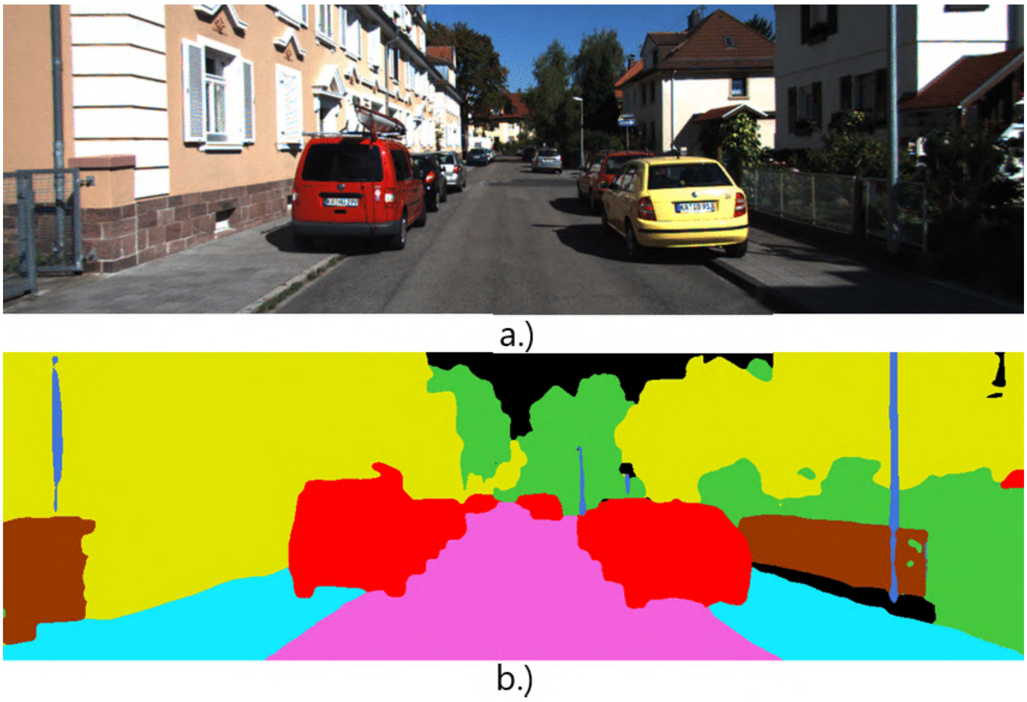


Fig. 11: a.) Image before segmentation. b.) Image after segmentation., [5]

- Clustering-based
- Edge-based
- Region-based
- Graph-based
- Model Fitting-based
- Machine Learning-based

Clustering-based segmentation

This segmentation method uses clustering algorithms, that rely on specific geometric features of the point cloud. The features can be positions, surface normals, reflectance, etc. One commonly used clustering technique is the K-means algorithm, which groups point cloud points by minimising the sum of squared distances between the point and the corresponding cluster centroid. The K-means algorithm identifies 'k' centroids and then assigns each data point to the nearest cluster, keeping the centroids as compact as possible. The term 'means', refers to the process of averaging the data to determine the centroid location. The operation of the algorithm is shown in simplicity in Figure 12.

A number of studies have reported satisfactory segmentation results using k-means clustering. Moreover, the algorithm is simple to implement and understand. However, clustering-based methods, including the K-means algorithm, have some limitations. They can be sensitive to noise in the data and may be influenced by how the neighborhood is defined. These factors can impact the accuracy and reliability of the segmentation results obtained from these methods, [30], [6] .

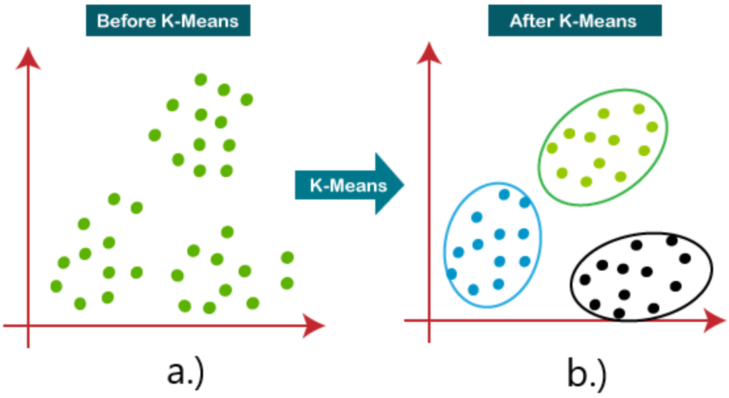


Fig. 12: a.) Input points. b.) Clustered points after K-means., [46]

Edge-based segmentation

Segmentation based on edges detects the boundaries between regions guided by specific rules based on mathematical properties, such as normals, displayed in Figure 13, gradients, higher order gradients and curvatures. Edge-based methods can identify edges using different approaches, depending on the technique. The edge-based methods can, for example, extract close contours from a binary map for segmentation, identify edges and then immediately group points within their boundaries, or, based on gradient information, fit a 3D lines to a set of points and identify changes in the unit normal vector on the surface. These are some of the principles that are nowadays used in edge-based segmentation.
Although edge-based methods are classified as fast segmentation techniques, they may lack accuracy and are more sensitive to noise and variations in point cloud density compared to other methods, [30] .

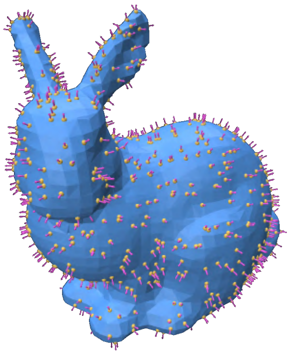


Fig. 13: Edge-based segmentation based on normals, [45]

Region-based segmentation

The basic idea of region-based segmentation is to select one or a few seed points and gradually expand the region by including neighbouring points based on certain criteria. The method of region growing is exposed in Figure 14. The algorithm typically performs by growing a region around identified point. The main characteristics for selecting the seed can vary. Nowadays, properties like geometric criteria or color features are used. Algorithms such as k-nearest neighbours, grow the region based on estimation of normal vector for each point. In the research papers, some approaches can be found that use normal and curvature constraints to obtain smooth areas, or even two stage rough and detailed segmentation. Two stage segmentation initially detects the main objects based on the normal vectors and afterwards increases the obtained amount of information by a subsequent finer segmentation. This structure of segmentation is called bottom-up.

There are also methods called top-down, which in contrast to previously mentioned bottom-up approach, do not use seeds to grow the regions. The top-down method works with the points the opposite way, the whole point cloud is taken as a single region in the beginning and is iteratively divided into smaller groups with similar characteristics called regions.

Region-based segmentation is generally more accurate than edge-based methods, but may be more sensitive for over or under segmentation and might have a problem to precisely select the boundaries. Nevertheless, this approach is also more robust to noise, due to global information, [30] .

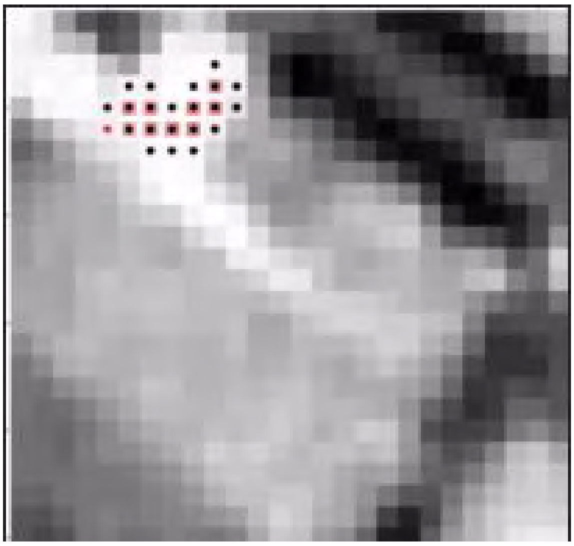


Fig. 14: Region growing segmentation, [25]

Graph-based segmentation

Point cloud segmentation using graph-based methods requires to work with point as a graph. Each vertex represents a point in the cloud, whereas the edges represent connected points in their neighbourhood, displayed in Figure 15. Some algorithms based on this methodology, build a minimum spanning tree from the graph. The others construct a 3D graph using k-nearest neighbours and together with penalty function, smooth the segmentation.

In general, these methods excel at segmenting complex point cloud data with variations in point cloud density or noise. On the other hand, they lack real-time processing and often require off-line training or some special sensor and camera system, [30], [8] .

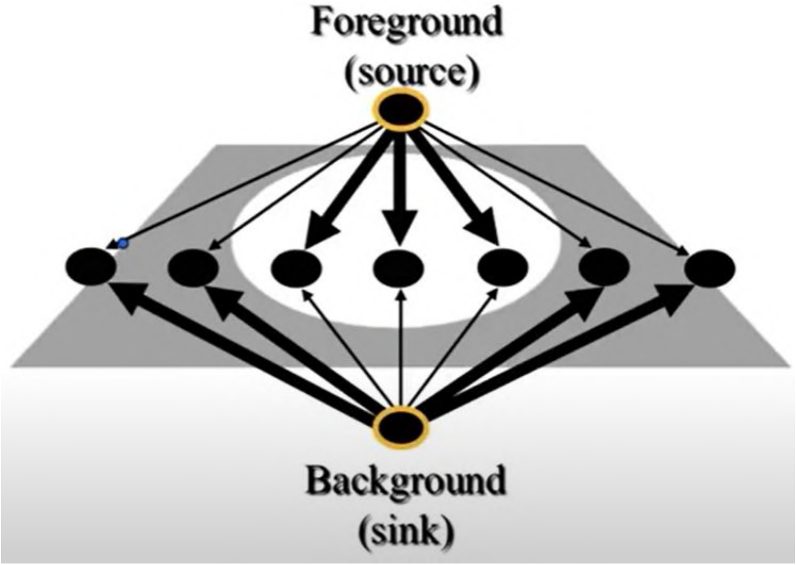


Fig. 15: Graph based segmentation, [25]

Model Fitting-based segmentation

The concept of model fitting segmentation is based on the experience, that the majority of man-made objects can be decomposed into simple geometric shapes like spheres, planes, cylinders and other primitives. These basic shapes are then fitted to the point cloud, similar to Figure 16. All the points, sharing the same mathematical objects, are then labelled in the same group. The most common and known approaches in this category are Hough transformation (HT) and Random Sample Consensus (RANSAC) algorithms.

The 3D Hough transformation identifies planes and other geometrical objects directly. However, this algorithm might be often considered as a slow and sensitive approach, especially when fitting more parameter-based objects due to its high dimensional computations.

The Ransac algorithm, on the other hand, is implemented chiefly for the robust fitting of parametric models and low sensitivity to noise. Ransac is also well known for its ability to avoid selecting outliers. Ransac produces numerous hypotheses of primitives shapes originating from a random subset of sample points. Methods derived from Ransac are able to automatically detect basic primitives in unorganised point clouds and can even locally fit basic shapes like cones, cylinders, planes, etc.

Model fitting segmentation methods are based on mathematical principles, which increases their robustness to noise and outliers. Ransac is also useful for processing large amounts of point cloud data. The Ransac algorithm in particular produces good results in a reasonable time. However, both can suffer from accuracy issues and fitting problems, [30] .

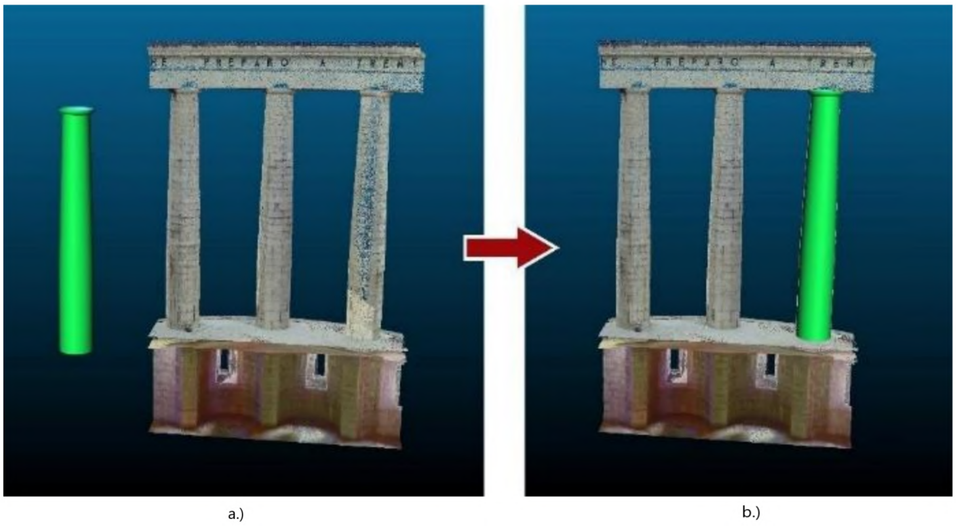


Fig. 16: a.) Model before fitting. b.) Model after correct fitting., [7]

2.4.2 Object recognition techniques

In order to create detailed 3D models of manufacturing parts from point cloud, object recognition is necessary for assigning labels to data segments or individual points. Object recognition can be defined as the recognition of objects in segmented instances. Object recognition methods can be categorised as data-driven or model-driven.

Whereas data-driven approaches use information such as shape, material or features, model-driven methods focus on predefined structure between objects. Some researches may combine these two methods or are use completely new approaches.

The next subsection summarises methods used for object recognition using point clouds. We can categorise these techniques into three groups listed below. Each group is then briefly described in its paragraph below, [30] .

- Geometric shape descriptor
- Hard-coded knowledge based recognition
- Supervised learning based recognition
- CAD model scan recognition

Geometric shape descriptor

Recognition of object instances usually uses geometric shape descriptors. These descriptors can describe manufactured object based on geometric features. The main steps in geometric shape descriptors are off-line library generation, on-line search in point cloud and a final verification.

The off-line library generation involves calculating all the geometric shape descriptors based on the CAD or BIM model and storing these information in the 'library'. For recognition of object instances are mostly used local or semi-local descriptors and global descriptors for recognition of object classes, thanks to their capability of handling more shaped variations. For example, a local geometric feature based on a point pairs, which can find pairs of points with a constant distance and calculate local feature based on normal vectors of these points.

After storing the library, on-line search is performed on a point cloud to find the target object by comparing the shape descriptors from the library with the ones calculated on the presented point cloud. Points with similar shape descriptors are then identified, based on some threshold value.

The verification process aligns to every possible matching point on a CAD point cloud model, to see if they match well. The alignment often requires a coarse registration Principal Component Analysis (PCM) and a finer registration similar to Iterative Closest Point (ICP) algorithm. A well aligned match, indicates the true location of the object in the point cloud data, [30] .

Hard-coded knowledge

Common approach to identify components which have distinct and varying geometric features is a segmentation of a point cloud data into some meaningful segments and categorisation these segments into object classes based on a predefined knowledge. Some methods use region growing algorithms with smoothing constraints and label them into specific category based on established knowledge. The know-how is usually built on observations, that the adjacent object edges are perpendicular to each other, or that upper plane of the object is often horizontal and the side edges are vertical.

The pre-defined knowledge can be classified into four main categories, such as: size, position, orientation and the topology of an object. Size is related to the dimensions of the part, so that the area of the object is within a certain range. Position relates to relative location of an element to another known element. Orientation cares about the main direction of a normal vector of an element. Lastly, topology is related to the topological relationship between the found components.

Despite the hard-coded based methodologies might seem efficient, even these approaches have limitations in recognition of irregular building components or when trying to extend this method to more complex object classes with more complicated geometries, [30] .

Machine learning

Recognition of objects in points can be done also by using the supervised learning algorithms. These methods involves training the classifier, that can segregate point cloud into object classes. The two approaches to machine learning object recognition are point-based classification and segment-based classification. Point-based classification covers classifying each point individually into a class using the local features. Whereas segment-based classification involves dividing the point cloud into homogeneous segments using data segmentation algorithms and classifies each segment into a class using feature of each segment afterwards.

Many well known approaches are grounded on Support vector machines (SVM) algorithm. For example SVM can be used as a final classifier of a previously segmented point cloud corresponding to different 3D CAD models. Some studies also rely on classifying point cloud using machine learning algorithms and a novel descriptors consisting of a few corresponding geometric features "to train the model", [30] .

CAD/BIM model scan

The above mentioned methods for object recognition do not specifically require the 3D CAD, BIM model of the whole structure. If the the point consists of only one object to be recognised, we can use much simple recognition approach known as BIM-vs-Scan. The approach involves matching the designed CAD model with the point cloud data.

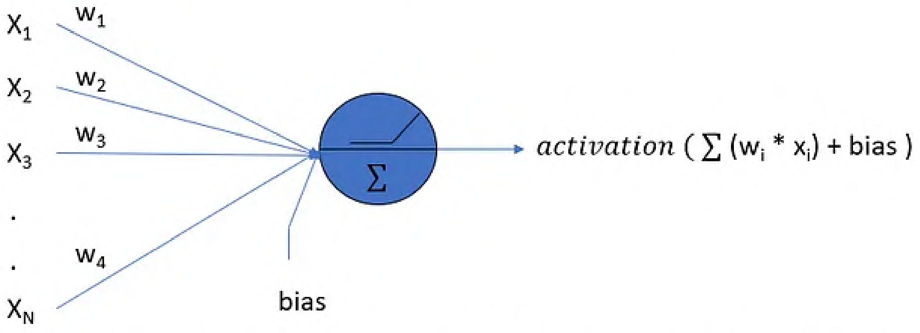
The mentioned principle consist of two steps. The point cloud data should be aligned with the CAD model, which is achieved through some manual alignment based on few parts of point or by a two step registration process containing coarse and fine ICP registration.
As for the second part, every point or data segment is matched to an element in the CAD model based on the geometric and semantic features of point in each segment, [30] .

2.5 Neural Network

2.5.1 General Neural Networks

Inspired by the way the brain processes information, neural networks mimic the basic operations of the human brain. They are being used for a range of real-time tasks, because of their ability to perform fast computations and quick responses. Basic neuron is diplayed in Figure 17.

Various components of an artificial neural network model are inspired by the biological nervous system. The network usually consists of a large number of interconnected processing elements, also known as nodes. These elements are connected to other nodes by connection links. The connection links incorporate weights that store information about the training. These weights are updated at each iteration. The neural network is trained, when all training data has been input. Such trained neural network with its architecture is used to solve specific problems within its definition. General artificial neural networks are used to solve a variety of problems including classification, pattern matching, data clustering or segmentation, [28] .



A single neuron shown with X_i inputs with their respective weights W_i and a bias term and applied activation function

Fig. 17: Basic neuron, [28]

Such artificial neural networks can learn how to solve problems very efficiently and can often adapt very well. Neural networks usually excel in their ability to learn how to solve a particular problem, in the speed at which they solve it, and also in their accuracy, [28] .

2.5.2 Working principles of Neural Networks

A neuron can be thought of as a linear model, either in single or multiple architecture, coupled with an activation function. The neuron in layer $[i]$ takes as an inputs the outputs of all the neurons $[i-1]$ in the previous layer. The neuron calculates the weighted sum, adds bias and then passes through an activation function. Forward propagation uses a fully connected architecture, which means, that the neuron in layer $[i]$ is connected to all outputs from previous layer $[i-1]$ and equivalently connected to all neurons to the next layer $[i+1]$ where the output from neuron in layer $[i]$ stands as an input to neuron in layer $[i+1]$. The principle is also visible in Figure 18.

Once the prediction is compared to an actual output, the loss is usually minimised by back-propagation method. This principle optimises the weights in order to minimise the final loss, [28], [20] .

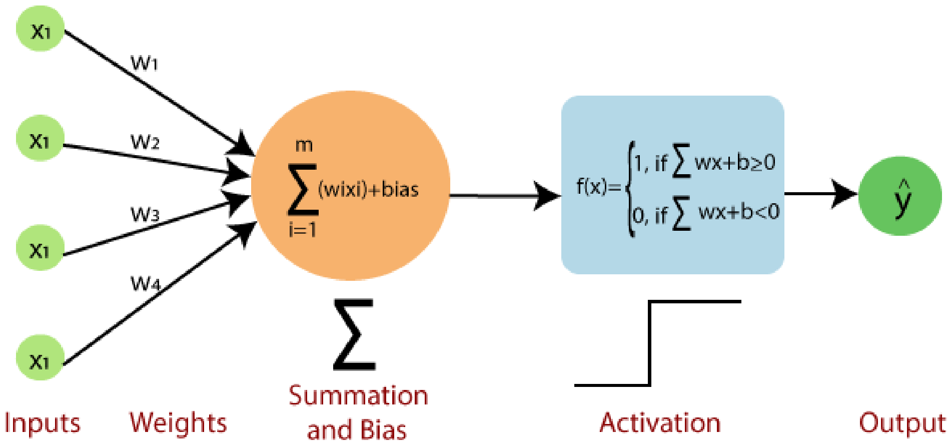


Fig. 18: Working principles of a neuron, [20]

2.5.3 Types of learning schemes

The fascinating aspect of neural networks is mainly their ability to learn from a structured data and produce output based on that learning. Neural networks can be divided into three categories based on the learning process.

- Supervised Learning

- Unsupervised Learning
- Reinforcement Learning

Supervised Learning

Supervised learning, visible in Figure 19, is a type of learning where a teacher or supervisor is involved in the training process. Input training pairs are provided, each pair being an input and a target output. The output of the model is compared to the desired output and an error is calculated. This error is fed back into the network to adjust the weights until the performance of the model matches the desired performance. This involves the environment feeding back to the model. In essence, supervised learning is similar to learning with a teacher, [20], [28] .

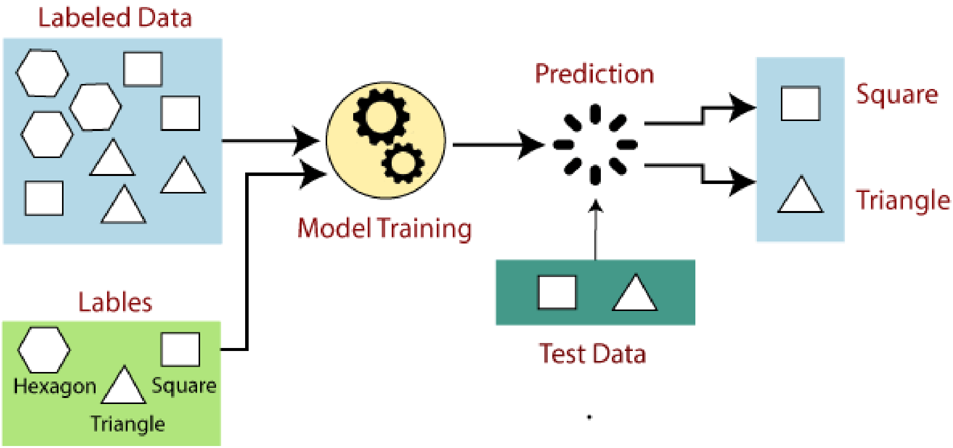


Fig. 19: Visualization of Supervised Learning, [53]

Unsupervised Learning

Unsupervised learning, shown in Figure 20, has no supervisor or teacher, unlike supervised learning. This type of learning has no feedback from environment and the model learns by itself. During the training phase, the inputs are grouped into classes based on their similarity. Each class contains patterns that tend to resemble each other. When a new pattern is input, the model is able to predict which class it belongs to based on its similarity to the other patterns. If there is no existing class for the pattern, a new class will be created, [20], [28] .

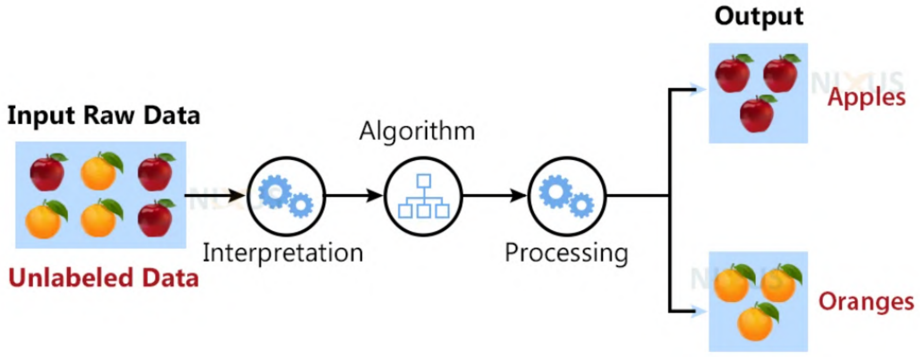


Fig. 20: Visualization of Unsupervised Learning, [54]

Reinforcement Learning

Reinforcement learning, displayed in Figure 21, combines aspects of both supervised and unsupervised learning. It can be imagined as learning with criticism. Instead of exact feedback from the environment, the feedback is in the form of a critique of how close the solution is to being correct. The model learns on its own, based on this feedback. Reinforcement learning is similar to supervised learning in that, it receives feedback from the environment, but it differs from supervised learning in that, it does not receive the desired output information. Instead, the feedback is received in the form of criticism, [20], [28].

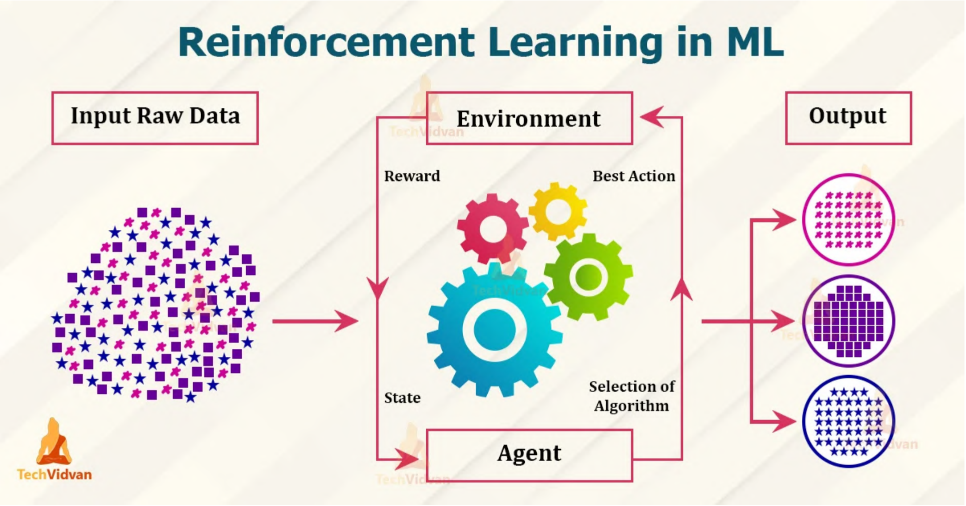


Fig. 21: Visualization of Reinforcement Learning, [28]

2.5.4 Types of Neural Networks

Neural Networks can be classified based on mathematical foundation and performance.

Feed-forward Neural Network

Feed-forward neural network, shown in Figure 22, so called because the data moves in one direction. The input data enters at an input and leaves at an output. There is no back-propagation of a signal in this type of layer. Back-propagation is only used as an algorithm to calculate the final loss function using the gradient method. Feed forward networks can also contain hidden layers and have a fixed length specified by the programmer, [20], [28] .

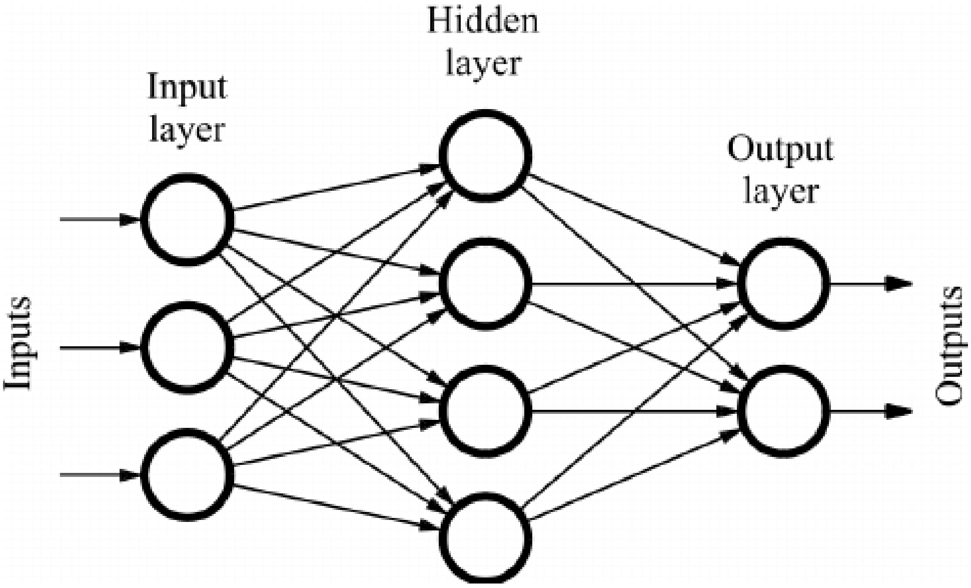


Fig. 22: Feed-forward neural network, [20]

Radial Basis Function Network

RBF networks, displayed in Figure 23, are a combination of input, hidden and output layers. By measuring the distance from a central point and interpolating, RBF networks categorises data. Interpolation resizes images and classification is done by estimating input data, with each neuron holding data. RBF networks group similar data points by searching the input area. Hidden layer outputs are summed and weighted to form a network of outputs, that are sent to the output layer, [20], [28] .

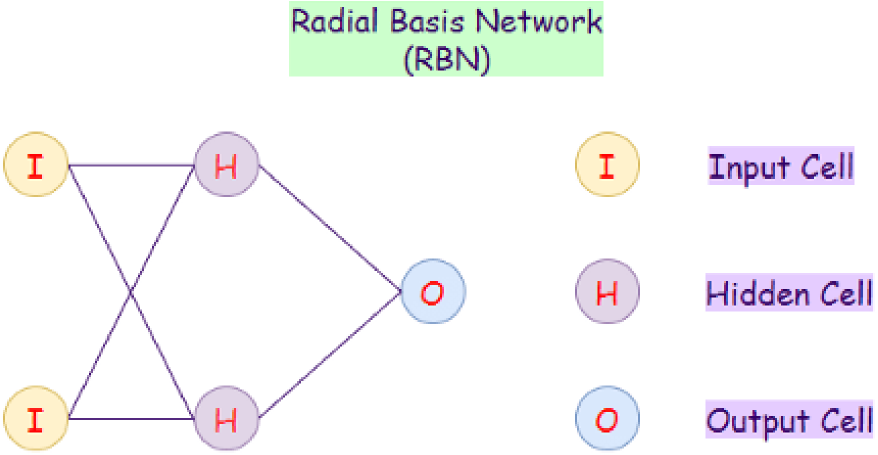


Fig. 23: Radial Basis network, [20]

Recurrent Neural Networks

The disadvantage of feed-forward network is, that it cannot remember data in past inputs. This is where RNN's come in help to solve this.

RNN is a network, visible in Figure 24, that does a good job of modelling sequential data well. Sequential data means data that follows a particular order in that a thing follows another. In an RNN, the output of the previous step can be used as the input of the current step, therefore the RNN is a feedback neural network. By storing the outputs, a better guesses can be made. In RNN, the data is passed through a loop, so each node remembers the data from the previous step. RNNs have a memory that helps the network to remember what has happened before in the data sequence. When performing predictions, neurons act as a memory cells. Most known type of RNN are Long short term memory networks (LSTM), [20], [28] .

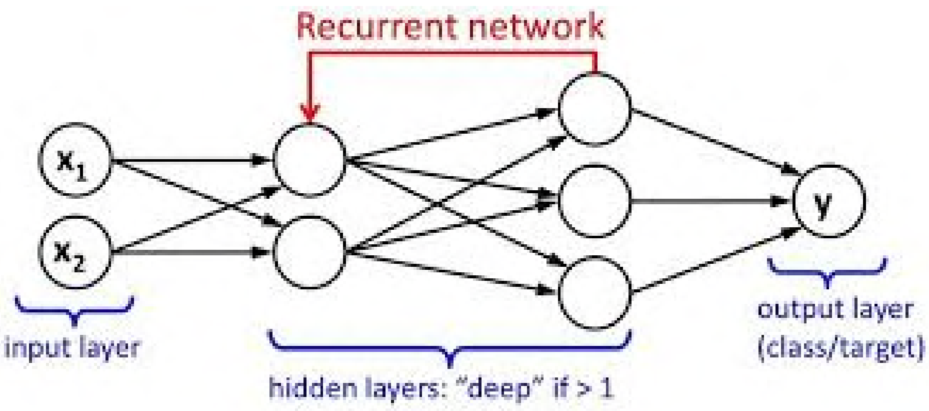


Fig. 24: Recurrent neural network, [20]

Convolutional Neural Networks

CNN architectures are usually used for image recognition, consisting of 3 stages of neuron arrangement.
The first stage is the convolutional layer, where neurons only process information from a particular segment. This is usually done by batching the input features.
Followed by a pooling stage, in which the dimensions of the features are reduced while the essential data is retained.
CNNs move on the third stage, the fully connected neural network, where the class of the image is evaluated to determine the final probabilities. The real-life application contains a combination of both, convolutional layers and neurons, [20], [28] .

2.5.5 3D Convolutional Neural Network

From a mathematical point of view, convolution is an integrative operation that measures the amount of overlap of one function on another while is shifted. Convolution mixes two functions in order to preserve the information.
In in the field of neural networks, convolutions are filters being used to extract features from an input data. In essence, convolution involves usage of a filter with adaptive weight matrices that traverses the input and computes the weighted sum as an output. This weighted sum is referred as feature space and is the input for the next layers, [2], [9] .

1D Convolution

1D convolutions, shown in Figure 25, are the simplest convolutions, usually used for sequence data. However, they have also several other objectives. One-dimensional convolutions help to extract 1D sub-sequences from the input sequences and to identify local patterns within the convolution window. The resulting features are obtained by applying a 1D convolution filter to sequence. 1D convolutions are commonly used in natural language processing to represent sentences in a sequence of words, [2] .

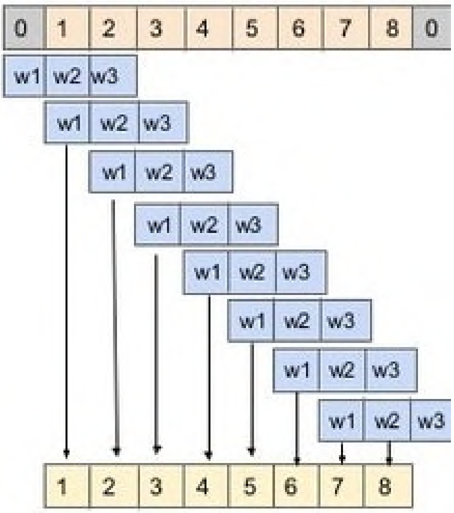


Fig. 25: Visualization of 1D convolution, [2]

2D Convolution

When dealing with image datasets, A.I. based algorithms mainly use 2D convolutional filters. The fundamental concept behind 2D convolution is that the convolutional filter moves in two directions X and Y to compute low-dimensional features from image data. 2D convolution is initially defined as the element-wise multiplication between the input and various filters. A 2D max-pool layer ($(2x2)$ filter) involves selecting the largest element from a small $(2x2)$ square defined within the input data. 2D convolution principle is visible in Figure 26. The output of this operation is a 2D matrix with a reduced dimension, [2], [9] .

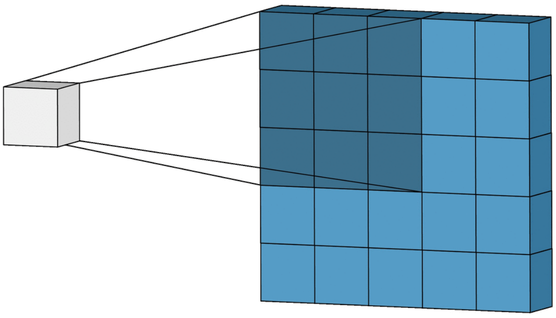


Fig. 26: Visualization of 2D convolution, [19]

3D Convolution

3D convolution, displayed in Figure 27, analytically uses a 3D filter to process the dataset, with the filter moving in three directions X, Y, Z to compute low level feature representations. The same element-wise multiplication is performed on the multiple pairs of 2D matrices in the input and filters. In a 3D max-pool layer there is a $(2x2x2)$ kernel, where the largest element is selected from a $(2x2x2)$ cube corresponding to the bounded space within the zone of the input data. The output shape of this operation is a 3D volume space, such as cube. Although they are designed for 3D space input, they can be also used for 2D space input, such as images. Due to the size of the filters used and the size of the input data itself, the number of operations increases in 3D CNN layers, including both convolution and max-pool layers. This increase in the number of operations is more significant as observed in the 2D CNN layers, [2], [9] .

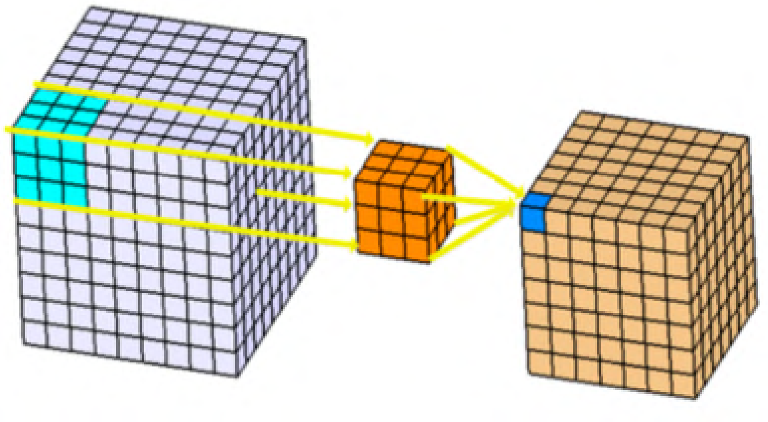


Fig. 27: Visualization of 3D convolution, [2]

2.6 Overview of used technology

In this section, we will provide an overview of the technologies used in robotic vision, specifically the Zivid 3D camera, Python programming language, and the Open3D and TensorFlow libraries.

2.6.1 3D Zivid Cam

Zivid is a Norwegian technological company headquartered in Oslo, Norway which specializes in machine vision cameras. The group produces 3D color cameras and associated vision software used in numerous industrial automation systems, such as industrial robot cells or collaborative robot cells (cobots).

The hardware field of the company specialises on products such as Zivid Two and Zivid One+ industrial colour cameras.

Zivid One+ was launched as an upgraded version of Zivid One in November 2018 at VISION 2018 in Stuttgart, Germany. The One+ product includes three 3D colour cameras, which could work in distance from 3 to 160 cm depending on the version.

Zivid One+ camera [56], is primarily used in industrial automation and robotic applications such as pick and place, bin-picking and assembly tasks. Due to its high resolution, high speed, and colour accuracy, this camera works well in a wide range of applications. The camera is also compatible with various robotic platforms and can be easily integrated into existing systems if the customised software is used.

In this thesis a Zivid One+ Small camera was used, displayed in Figure 28. This camera has an optimum working distance of 350 to 700 mm. The camera's field of view varies from 300 to 1000 mm, providing flexibility in capturing objects of different sizes. One of the features of the Zivid One+ Small camera is its noteworthy spot accuracy, which ranges from 25 to 500 µm. The camera produces a final image with a resolution of 2.3 MPx (1920 x 1200 points) with native 3D colour. In addition, the camera's point cloud output includes 3D coordinates (XYZ), colour information (RGB) and contrast for each pixel. In terms of performance, the Zivid One+ Small camera boasts an capture time, with a single frame taken in just 80ms at 1/154s exposure. This fast capture time allows fast moving objects to be captured or quick inspections to be carried out without compromising image quality or accuracy, [55],

.

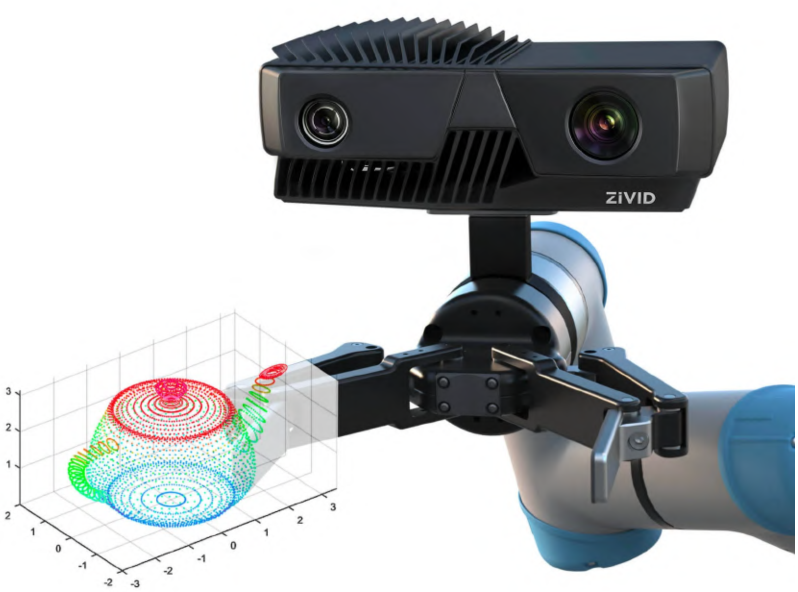


Fig. 28: Zivid One+ camera, [48]

2.6.2 Python

Python is a high-level, interpreted, object-oriented programming language with dynamic semantics. Its built-in high-level data structures and dynamic features, which make it attractive for rapid application development and as a scripting language for connecting existing components. Python supports modules and packages, which encourages a modular programming approach and code reuse. The Python interpreter and an extensive standard library are freely available in source or binary format for all major platforms. Python was chosen as the programming language for this project because of its wide range of libraries and frameworks that greatly simplify data processing, machine learning, and computer vision tasks, [42] .

2.6.3 Tensor-Flow Framework

Tensor-Flow was developed by Google as an open-source library, primarily used for deep learning applications. Originally, Tensor-Flow was designed mainly for large numerical computations without any consideration of deep learning. Eventually, it has evolved to become one of the most popular frameworks for deep learning. Ever since its first version appeared in 2017, it has become one of the most popular deep learning frameworks with an enormous library for large-scale machine learning computation.

One of its advantages is the ability to use different hardware resources such as CPU and GPU, allowing faster computations and parallel processing. TensorFlow is designed to take advantage of modern GPUs, which have high computational capacity and are optimized for parallel operations. Using TensorFlow, computations with high-dimensional tensors can be performed efficiently, which is important for processing large datasets. GPUs provide the advantage of parallel processing, which means that a number of operations can be performed simultaneously. This is useful for mathematical data processing, convolutional neural networks, and other machine learning algorithms. Overall, TensorFlow allows to efficiently take advantage of both CPU and GPU for fast and parallel computations, while being able to efficiently process large data with high-dimensional tensors, [1] .

2.6.4 Open 3D Library

Open3D is an open-source library, that is designed to make it easier to quickly create software that handles 3D data. The front-end provides a well-chosen set of data structures as well as algorithms in both Python and C++. The back-end is, on the other hand, well optimised and ready for parallel computing. Open3D was originally built from scratch with minimal dependencies. This fact allowed to easy set up the library as well as cross platform compilation. The library has been utilized in several research papers and is widely adopted in cloud environments. The main domains, it covers, are visualisation, 3D machine learning, robotics, etc., [38], [10] .

3 Process of recognition and pose estimation

Using segmentation algorithms for bin-picking tasks requires both testing performance on a sufficiently large dataset and segmentation of specific objects. In order to correctly pick different parts from a bin, the pose for each part must be correctly estimated. To train a neural network for a specific objects, a specific dataset must be used. In the following chapters, the algorithm for dataset generation, scene segmentation, and pose matching is presented.

3.1 Dataset Generation for Bin-Picking Tasks

Since bin-picking is a task mostly used in the manufacturing industry, it is usually necessary to segment different objects in most cases. The manufacturing industry usually provides 3D CAD model data for each object. The creation of a custom dataset for each specific object by labelling is very demanding, either from a time or financial point of view. Based on this knowledge, it was decided not only to test the implementation of segmentation and registration on publicly provided datasets, but also to work on the whole process of bin-picking, including the generation of datasets consisting only of objects selected by the author. The idea behind a synthetic data generator is to have an all-in-one product package that can either be trained on a self-made generated dataset, or be able to produce predictions of segmentations from a real world dataset.

The data generation algorithm that is presented in this thesis, takes as an input a 3D CAD model of an object, usually in a *.stl* format, and produces a series of scenes consisting of several 3D objects that have been previously inserted.

The second approach is to use pre-made datasets that are available for download, such as the XA Bin-picking dataset [34] .
This dataset includes both simulated and real-world scenes featuring few industrial objects. The dataset contains over 1000 training samples of scenes where the ground truth instance labels have been manually created. More than 20 objects can be found in the randomly arranged scenes. Each scene contains over 60 0000 annotated points. The parts are devoid of texture and color and both the training and test samples contain only the boundary points of the parts, [34] .

Another publicly available dataset is the Fraunhofer IPA Bin-Picking dataset, example of the dataset is visible in Figure 29. The dataset also includes simulated and real scenes with many objects. The scenes in the dataset are fully annotated with 6D poses. The multi-part scenes are generated by a physics simulation in which objects are dropped into a bin in random positions and orientations. This dataset also extends the Siléane dataset by providing additional examples. This addition is particularly useful for training deep neural networks and measuring performance, [39], [11] .

These two public datasets provide sufficiently good amount of data for testing the performance of every new architecture of Neural Network for either object classification or registration.

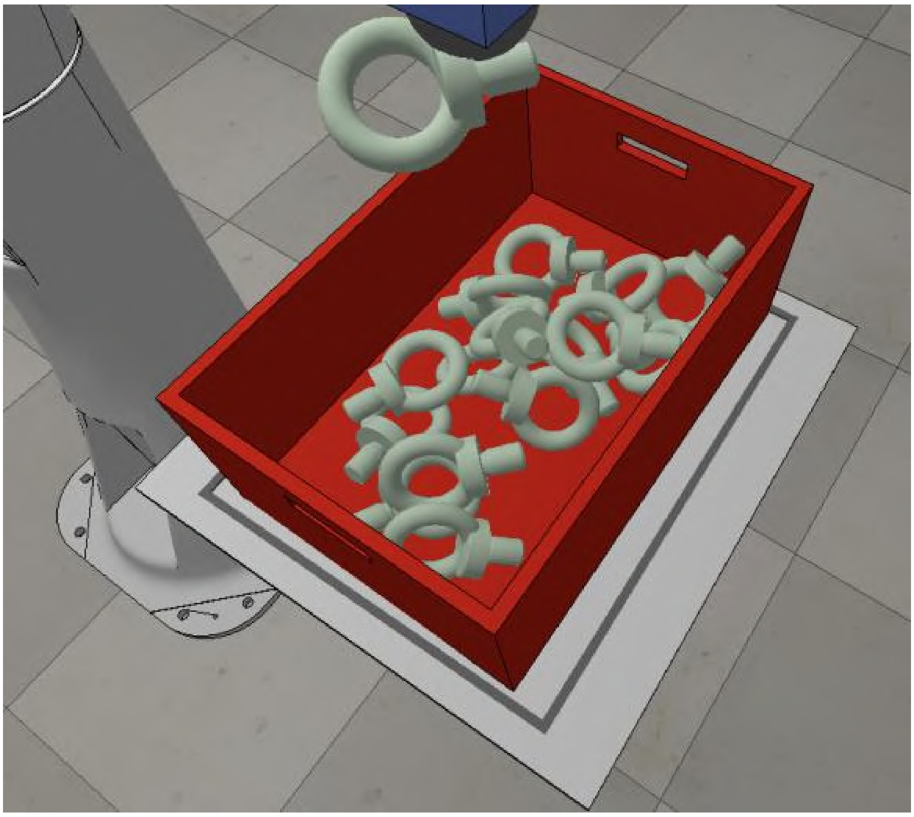


Fig. 29: FraunHofer IPA dataset, [39]

3.1.1 Data Generation Theoretical Intro

The data generator used a random object centre generator, which places an object in space by setting an X,Y, and Z coordinate. There are two different probability functions for selecting the position on the surface and the height of an object. A uniform probability function is used to select an X and Y coordinate to ensure a homogeneous distribution of objects on a surface. A uniform probability function is used to place the object in a plane to ensure a homogeneous distribution of objects in the plane. The height of the centre was initially secured by a $U-shaped$ probability distribution. Later, an option to select a height of 2 or 3 discrete values was added because it also provided very good results.

If the object is symmetric around the Z axis, one rotational axis suffices to control the rotation around the x or y axis. Other axes of rotation are frozen due to symmetry, because any rotation of the object around that axis, will not cause any changes.

If the object is inhomogeneous, the other 2 degrees of rotation can also be set to adjust the resulting position of the object in space. All axes of rotation are normally generated by a normal distribution function with a desired offset.

Simple object generation, even with correct probabilistic distribution, produces a "ton" of overlapping objects. This bug was partially fixed by randomly generating a large number of objects until a non-overlapping object was found. Since this method of randomly finding a non-overlapping coordinate vector takes an enormous amount of time, it was decided to use down-sampled objects. Example of a generated data scene exposed in Figure 30.

While working with down-sampled objects has saved some time when creating a scene, it has not brought any massive improvement. The overlapping of objects is checked by finding out if any point of a new object is in the space of all previously placed objects.

The final algorithm chooses the best generated position based on the number of overlapping points. The object coordinates with the fewest overlapping points are preselected. As the new coordinates are selected in the scene, the final object is up-sampled back to the originally selected values.

The algorithm continues in this way, creating as many objects in the scene as are selected. In this way it is able to achieve a sufficiently large and diverse dataset to train a neural network.

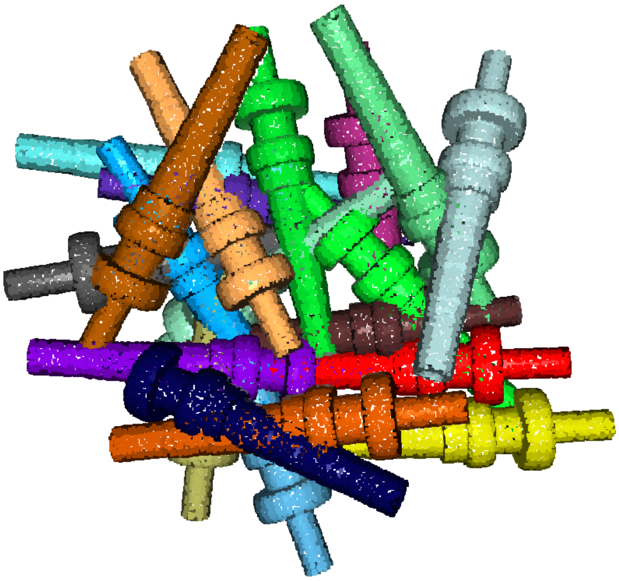


Fig. 30: Example of generated dataset

3.1.2 Data Generation Implementation

The implementation of a presented dataset generating algorithm is pretty much straightforward. A user chooses a folder to save the results and selects the CAD model in *.stl* format, from which the dataset will be generated. The user has an option to save the generated scene and to check the non-overlapping part of the algorithm. The objects are generated in a loop, while the model is down-sampled at the beginning. For each position generated each point of the newly created object in the proposed position is tested with the previously placed objects, whether it is located inside one of them. In this way all points of the new object are passed in the cycle. In case none of the points is located inside the object, the object is stored at the selected location.

If at least one point is inside one of the stored objects, a new position is generated for the placement of the model. This continues until a location that has no collision with existing objects is found. The second option is set if the maximum allowed number of place generation is exceeded. In this case, the position already generated, which had the least number of points in the collision, is selected.

After all points are placed, the algorithm up-samples points for each individual objects. The output of the algorithm is a folder containing a point cloud in '.txt' format, folder with RGB images for each generated scene as well as correctly labeled segments in '.txt' format for control.

3.2 Scene Segmentation

The presented segmentation method consists of training a network on a 3D point cloud, which is usually generated by a 3D model of the target object. The proposed method involves identifying the geometric centres of each part and using these points as the main reference for the upcoming clustering. These centres are related to the number of instances of an object in the scene. Therefore, this approach can eliminate redundant merging algorithms, which consume a lot of computing time and can introduce errors in highly overlapping scenes, [33] .

3.2.1 FPCC Theoretical Intro

The backbone of the FPCC neural network is based on the transformation of the original 3D point cloud data into a new coordinate system, where each point is represented by a six-dimensional vector XYZ, equation 3, and a normalised vector (n_X, n_Y, n_Z). The resampled point cloud is fed into the network, which outputs a 256-dimensional feature output and a percentage of a centre score for each point. The features extracted from the FPCC are then fed into two branches, an embedded features branch and a centre score branch, [33] .

$$
\begin{aligned}
&F = (\bar{x_i}, \bar{y_i}, \bar{z_i}, n_x, n_y, n_z) \\
\\
&\bar{x_i} = x_i - \min(x_1, x_2, ..., x_N) \\
&\bar{y_i} = y_i - \min(y_1, y_2, ..., y_N) \\
&\bar{z_i} = z_i - \min(z_1, z_2, ..., z_N)
\end{aligned} \tag{3}
$$

The two resulting branches output an embedded feature and a centre score for each point. The non-maximum suppression algorithm is immediately applied to all points to identify the most valuable centres of each instance. Points with a centre score greater than some threshold percentage, are automatically considered as a candidates for the centre points. The point with the highest centre score is selected as the first candidate centre point and all other points within a distance sphere are removed. This process is repeated until no more points remain.

All the points except the centroids are then clustered based on the closest distance to the centroid. The nearest centroid of the point p_i in the feature embedding space is found and then the distance between the point and a centre is calculated. If the distance exceeds the threshold value, the point is labelled as a noise and is not assigned to any other group, [33] .

This clustering method differs from the conventional clustering methods, because it does not require the entire scene to be downsampled into multiple batches and clustered in their batches. The basic process of segmentation is visible in Figure 31.

The final network loss during the training is a combination of the losses from the embedded feature branch and the centre score branch, [33] .

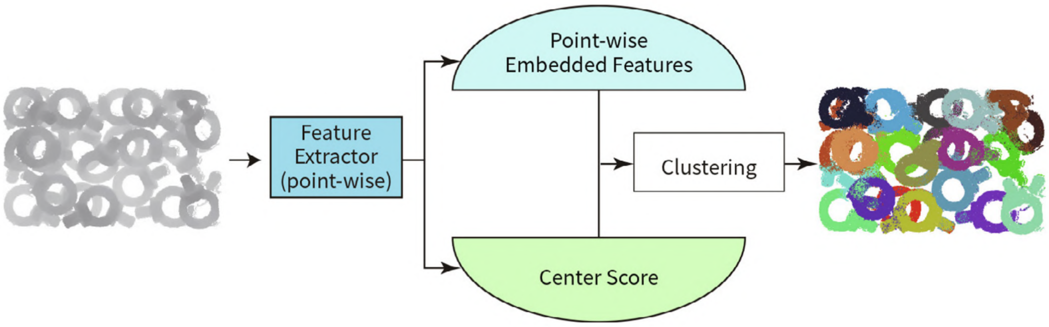


Fig. 31: FPCC segmentation process, [33]

Feature distance matrix

The feature distance matrix, which is part of the feature embedding space, ensures that the points belonging to the same instances are close to each other. On the other hand, the points belonging to different instances should be far from each other. Equation 4 calculates the distance between two feature i and j [33] .

$$d_{F(i,j)} = ||e_F^{(i)} - e_F^{(j)}||_2 \tag{4}$$

Valid distance matrix

This matrix consists only of binary elements. The aim is to train the network to discriminate whether pairs of points within a given Euclidean distance belong to the same instance or not. In the inference phase, the overall clustering of the points depends on both the feature distance and the Euclidean distance of the point pairs (centre-point - point). If the Euclidean distance between any two points is greater than twice the maximum of a threshold distance, the points are considered not to be in the same group, equation 5. These points will be ignored later to prevent them from contributing to the final loss, [33] .

$$d_{V(i,j)} \begin{cases} 1, & \text{if } ||p_i - p_j||_2 < 2d_{max}. \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Centre score

To ensure that the centre score characteristic reflects the distance between a point and its corresponding centre, the points close to the centre of an object have higher scores than those on the boundary. To evaluate this, a centre score vector is constructed according to the following equation 6, [33] .

$$S_{center(i)} = 1 - (\frac{||p_i - c_i||_2}{d_{max}}^{\beta}) \tag{6}$$

Attention score matrix

The Attention Score Matrix represents the significant pairs of dots by assigning weights between them in a matrix. This is done by calculating the weights based on the distance of the point pair from the centre position point, equation 7, [33] .

$$S_{A(i,j)} = \min(1, S_{center(i)} + S_{center(j)}) \tag{7}$$

Embedded feature loss and Center score loss

As the point pair has two possible relationship scenarios, as belonging to the same instance or not. The embedded feature loss L_{EF} is defined in equation 8. ϵ_1 and ϵ_2 are constants, which satisfy the condition $0 < \epsilon_1 < \epsilon_2$, [33] .

$$\begin{gathered} L_{EF} = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,j} \kappa_{i,j} \\ w_{i,j} = d_{V(i,j)} S_{A(i,j)} \\ \kappa_{(i,j)} \begin{cases} max(0, d_{F(i,j)} - \epsilon_1), & \text{if } p_i and p_j \text{ in the same instance .} \\ max(0, \epsilon_2 - d_{F(i,j)}), & \text{otherwise.} \end{cases} \end{gathered} \tag{8}$$

For a function for the center score branch a Smooth L_1 loss is used in equation 9.

$$\begin{gathered} L_{CS} = \frac{1}{N} \sum_{i=1}^{N} smooth_{L1}(S_{center(i)} - \hat{S}_{center(i)}) \\ smoothL1(x) \begin{cases} 0.5|x|^2, & \text{if } |x| < 1. \\ |x| - 0.5, & \text{otherwise.} \end{cases} \end{gathered} \tag{9}$$

3.2.2 FPCC Implementation

The implementation of the chosen algorithm was taken and modified from Github [33]. The whole algorithm was rewritten to the newer version of a Tensor-Flow 2.x from a Tensor-Flow 1.x, which makes it possible to run with the latest Python. The algorithm has also been rewritten into a newer structural style of OOP programming, to encapsulate all the features and functions of the bin-picking process into one class.

The implementation of the algorithm consists of two main functions for training and prediction. The architecture of the neural network is described in the following chapter. In the train function, the entered data for training is first read. In case a saved checkpoint from the previous training is found, the neural network continues with further training. It is possible to choose a backbone from the pair 'DGCNN' or 'PointNet'.

Before training, specific values are set for various parameters such as the maximum distance, number of epochs, number of input points to the net, size of training batch or weight decay.

3.2.3 FPCC Neural Network architecture

The architecture of the final neural network is based on the architecture proposed by [33]. The neural network consists of two parallel parts connected to the backbone, a feature extraction network called Dynamic Graph Convolutional Neural Network (DGCNN). The main focus of this thesis was on the DGCNN as the backbone. This was based on tests, referenced in the research paper, where they compared it with PointNet and PointNet++, especially in the area of learning the geometric features. As a feature extraction model, the DGCNN takes $n \times 6$ points as input. At the convolutional level, an edge feature set of size k is computed for each point. Features within each set are aggregated to compute the edge convolution response for the corresponding points. The basic architecture of DGCNN is displayed in Figure 32, [33], [31] .

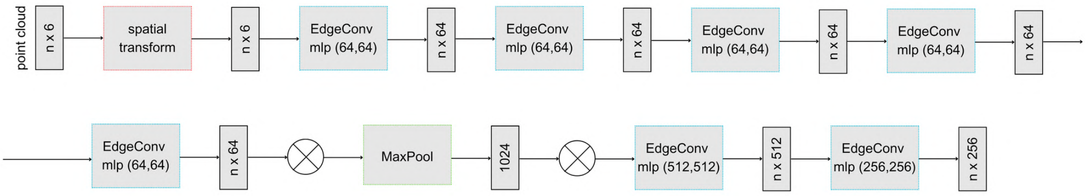


Fig. 32: DGCNN architecture, modified from [31]

The point cloud transformation block, in Figure 34, is responsible for aligning the input set of points to a canonical space by applying an estimated 3×3 matrix. To estimate the matrix, the block uses a tensor that concatenates the coordinates of each point and the coordinate differences between its k neighbours, [31] .

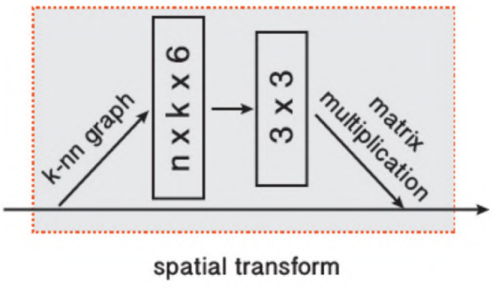


Fig. 33: Principle of DGCNN transformation, [31]

The edge convolution block, in Figure 34, takes a tensor of shape $n \times f$ as input, computes edge features for each point by applying a multi-layer perceptron (MLP) with the number of layer neurons defined as $a_1, a_2, a_3, ..., a_n$, and generates a tensor of shape $n \times a_n$ after pooling between adjacent edge features, [31] .

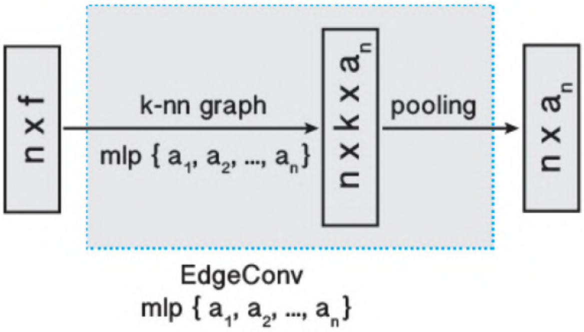


Fig. 34: Principle of DGCNN edge convolution, [31]

After the DGCNN there are two parallel branches, both taking the output of the DGCNN as input. The first part of the neural network is responsible for feature extraction. Features pass through a multi-layer perceptron (MLP), similar to the one in the DGCNN, to produce an embedded feature of size $N \times 128$.
The parallel centre score branch uses a double MLP layer to both reduce the dimensionality of an output from DGCNN and to activate pointwise features by a sigmoid function immediately after the double MLP layers. Using this approach, a centre score $s_{\hat{center}}$ prediction can be made at each point. The whole architecture of the segmentation network is visible in Figure 35, [33] .

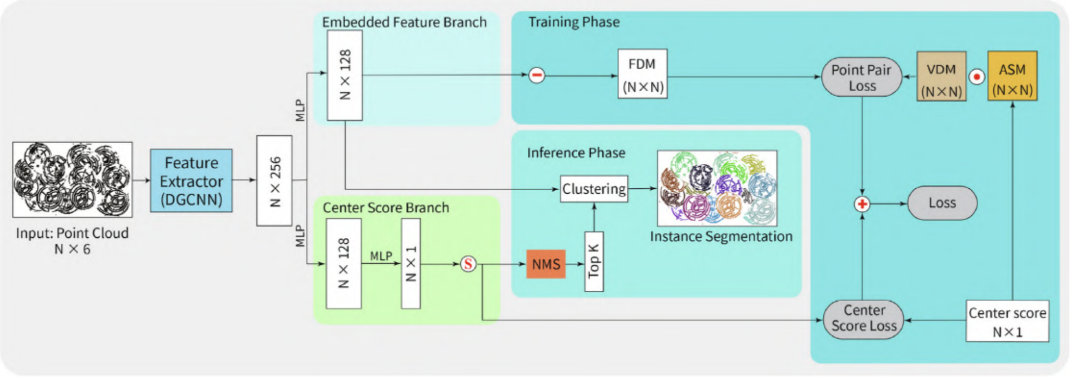


Fig. 35: FPCC neural network architecture, [33]

3.3 Object Registration

As mentioned above, registration in the bin-picking process is a challenging and difficult task. With a proper output from a segmentation part, a satisfactory close estimate of an object model transformation can be obtained. From a correct transformation estimate, information can be extracted for the gripper to select a correct location to pick up an object, etc.

Initially, a large number of publicly available algorithms were used for this task, either based on mathematical transformations or neural networks. Although a significant number of algorithms were tested on this task, such as (neural network) GMCNet, OMNet or a large (open source library) Teaser plus plus, none of these methods seemed to work and were not able to produce reasonable and correct transformation results. Only the ProBreg library was able to produce reasonably good results from time to time. The accuracy of the results was not very good and the algorithm lacked stability, [22], [32], [36], [35] .

For all these reasons, chosen algorithm has to be more "heavy" but above all robust. The point pair algorithm was originally proposed by [4]. The actual algorithm implementation followed code implementation and some ideas from [37].

3.3.1 Point Pair Feature Theoretical Intro

The algorithm assumes that both the scene and the CAD model are represented by a finite set of oriented points. Each point in the point cloud is also characterised by a normal computed from its neighbourhood.

The algorithm consists of two phases. Off-line training creates the global model description, while the on-line phase selects the reference points. All other points in the scene interfere with the reference point to create point pair features. These features are mathematically related to the model description. The voting of a pose estimation algorithm is similar to the voting in Hough transform. Each potential point match of a point pair votes for an object pose that provides the optimal pose estimation, [4] .

Feature vector

The point pair feature describes the relative position and orientation of two points and searches for those with similar characteristics to the model point pairs. It uses the a vector, which consist of a values such as distance between points, normals and relative normals, to categorise point pairs. We can define vector F for each point p_1 and p_2 with normals n_1, n_2, distance $d = p_1 - p_2$ in equation 10. The visualization of this vector is shown in Figure 36 [4] .

$$F(p_1, p_2 = [\|d\|_2, \angle(n_1, d), \angle(n_2, d), \angle(n_1, n_2)] \tag{10}$$

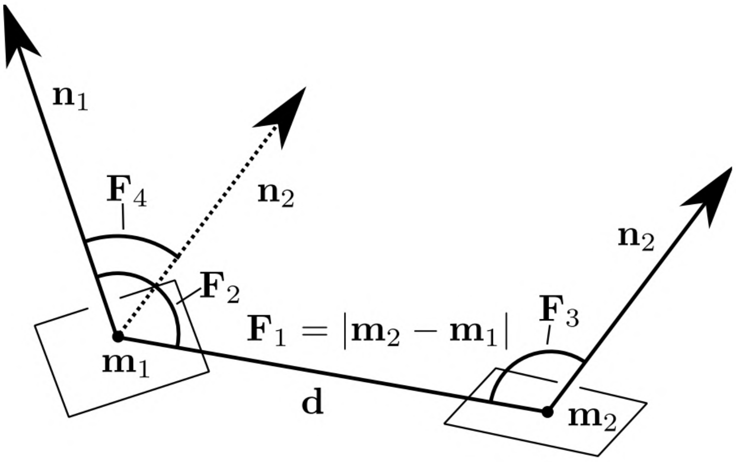


Fig. 36: Visualisation of a feature vector, [4]

Global Model Description

The aforementioned point pair features are used to construct a global model representation during the off-line training phase. The model consists of a collection of point pair features, where features with similar feature vectors are grouped together. To achieve this, the feature vector F described by the equation 10 is computed for all point pairs (m_i, m_j) belonging to the model M on the model surface. Distances and angles in the feature vector are sampled at intervals of d_{dist} and $d_{angle} = \frac{2\pi}{n_{angle}}$. All the features, that have identical discrete versions are then combined together. This can describe the global model representation as a mapping from the feature space of sampled point-pairs to the model, equation 11, [4].

$$L : Z^4 \Rightarrow A \subset M^2 \tag{11}$$

Knowing this, the four-dimensional point-pair feature vector previously defined in the equation 10 is mapped onto the set A, containing all the pairs $m_i, m_j) \in M_2$ that define equivalent feature vector, [4] .

Voting Scheme

Consider any arbitrary referenced point $s_r \in S$ of the scene and assume, that it is located on the object, which is searched. If this assumption is correct, then there exists a point $m_r \in M$ that corresponds to s_r. Once these two points and their normals are aligned, the object can be rotated about the normal of s_r to align the model with the scene. This removes another degree of freedom for the pose of the model in the scene. The fixed movement from model space into scene space can be described by a point on the model and a rotation angle α. Such pair (m_r, α) is called a local coordinates of the model, relative to the reference point s_r. In this method, a point pair $(m_r, m_i) \in M^2$ is aligned with a scene pair $(m_r, m_i) \in S^2$ where both pairs have a similar feature vector F. The transformation itself from the local model coordinates to the scene coordinates is defined by the equation 12 and displayed in Figure 37. Note that the local coordinates have 6 degrees of freedom (one for the rotation angle α and two for a point on the model surface), whereas a general rigid motion in movement in 3D has 6 degrees of freedom, [4] .

$$s_i = T^{-1}_{s\to g} R_x(\alpha) T_{m\to g} m_i \tag{12}$$

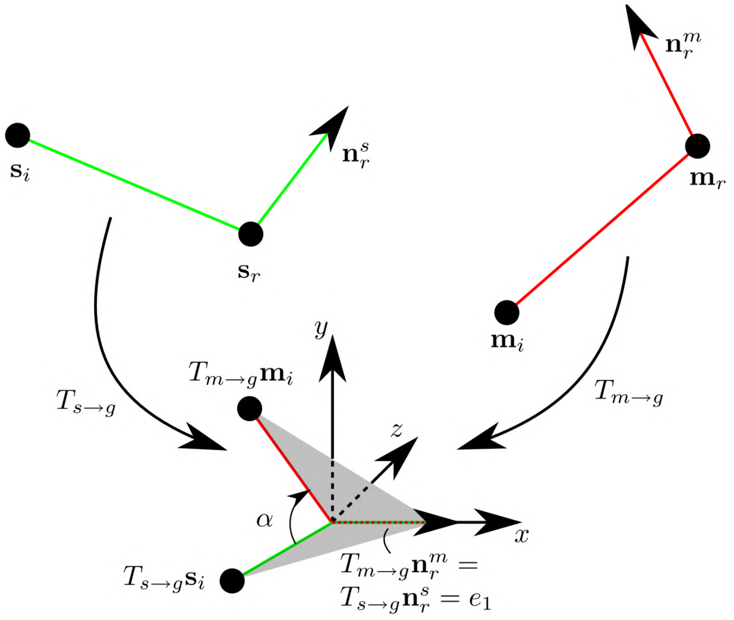


Fig. 37: PPF transformation between model and scene, [4]

Given a fixed reference point s_r, our goal is to identify the optimal local coordinates that maximise the number of scene points lying on the model. This is a similar approach to the generalised Hough transformation, which is usually very efficient when the local coordinates have only three degrees of freedom. Once the optimal local coordinates are found, the global pose of the object can be reconstructed.

In the actual voting process, the model surface is searched for the point pairs (m_r, m_i) with similar distance and normal orientation to (s_r, s_i), by pairing the reference point s_r with each point $s_i \in S$ from the scene. This search determines where on the model the scene point pair (s_r, s_i) could be located, and is performed using the off-line precomputed model description. The feature $F_s(s_r, s_i)$ is computed and also used as a key to the global model description hash table, which returns a set of similar feature vectors on the model. For each and every matching pair (m_r, m_i) or each possible position of (s_r, s_i) as shown in Figure 38. Then it's voted for the local coordinates (m_r, α). Figure 38 displays also the voting process.

After the processing all points s_i, the peak of the accumulator array corresponds to the optimal local coordinate from a which a global rigid motion can be computed. For stability reasons, all peaks that receive a certain number of votes relative to the maximum peak are used, [4] .

To increase the efficiency of the algorithm, α is split into two parts $\alpha = \alpha_m - \alpha_s$, where α_m and α_s depend only on the point par in the model and scene respectively. $R_x(\alpha)$ is also split into $R_x(-\alpha_s R_x(\alpha_m))$ and $R^{-1}x(-\alpha_s) = R_x(\alpha_s)$. This is achieved by using an equation 12.

$$t = R_x(\alpha_s) T_{s \to g} s_i = R_x(\alpha_m) T_{m \to g} m_i \tag{13}$$

In this case, t lies on the half-plane which is defined by the $x - axis$ and the non-negative part of the y-axis. For each pair of points in the model or scene, t is unique. Consequently, α_m can be precomputed for each pair of model points during the off-line phase and stored in the model descriptor. Using approach, α_s only needs to be calculated once for each scene point pair (s_r, s_i) and the final angle α is a simple difference between the two values, [4] .

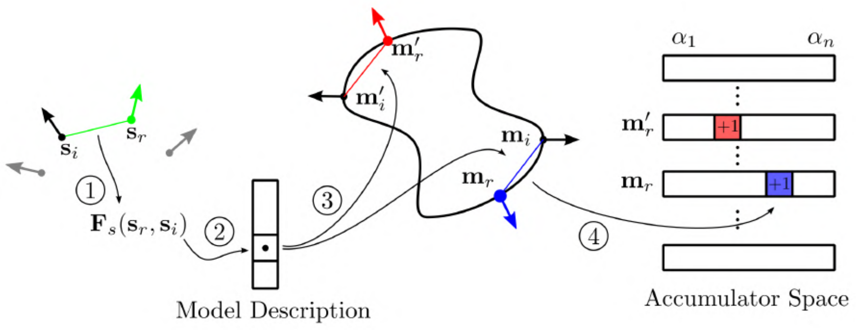


Fig. 38: PPF voting process, [4]

Clustering

If the reference point is on the surface of the object, then the object pose is determined by the voting scheme above. As a result, there is a need for multiple reference points in order to ensure that at least one of the reference points is on the object of interest.

As shown above, each reference point provides a set of potential object poses corresponding to the peaks in its accumulator array. However, these retrieved poses are only approximate to the ground truth due to different sampling rates of the scene, the model and the rotation sampling in local coordinates.

In order to both eliminate incorrect poses and increase the accuracy of the final result, the retrieved poses are clustered so that all poses within a single cluster do not differ in translation and rotation by more than a predefined threshold. The total score of a cluster is the sum of the poses it contains, where the score of a pose is the number of votes it received in the voting scheme. After identifying the cluster with the highest score, the final pose is computed by averaging the poses within that cluster. Since the scene may contain multiple instances of the object, the method may return multiple clusters. Pose clustering improves the stability of the algorithm by discarding isolated poses with low scores, while the averaging step improves the accuracy of the final pose. In Figure 39 , the complete process of PPF registration is displayed, [4], [37] .

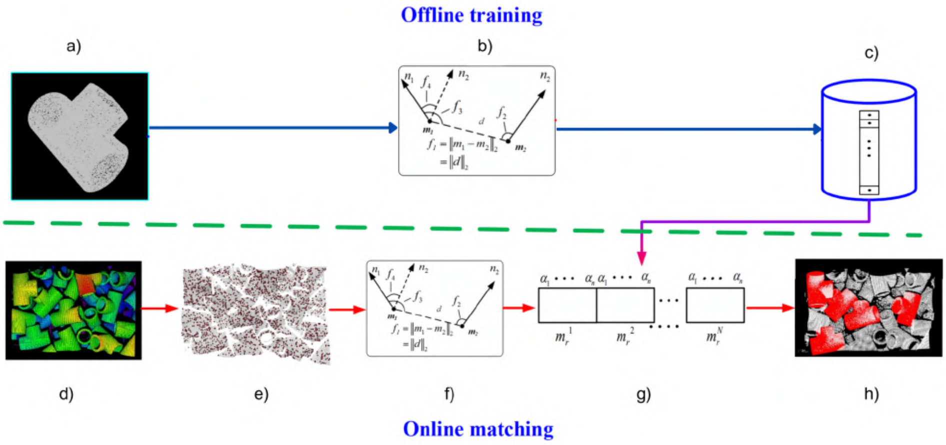


Fig. 39: Visualisation of PPF algorithm : a.) 3D Model. b.) Generation feature vectors c.) Extracted PPF stored in Hash table. d.) Input scene. e.) Preprocessing. f.) PPF Extraction. g.) Hashtable lookup and voting. h.) Final registration., modified from [13]

3.3.2 Point Pair Feature Implementation

This code implementation followed code ideas from [37] .
The code is also written by the OOP programming paradigm. The main class has two essential functions for offline training and online matching.

For model training, the inserted point cloud can be downsampled if selected. To find and add vectors in certain sets, the dictionary structure was used as a hash table. The training itself involves a cycle mapping relationship between each two different points. In the implementation, the model representation is stored as a hash table indexed by a feature vector F. A hash function is in the structure of the Python dictionary was used to map the 4D vector F to a 1D integer. Using this approach to accessing the hash table, all model features $F_m(m_i, m_j)$ that are similar to a given scene feature $F_s(m_i, m_j)$ can be quickly retrieved. This process continues until the features are mapped for every two points.

Fitting the scene to the pre-trained model requires similar initial steps as downsampling or computing the maximum interval coordinates of the scene. The model fitting itself consists of a double loop in the first step of the fitting. Each time the hash generated from the scene matches a hash in the stored model hash table, the accumulator matrix is incremented by a certain index. We implement the matching scheme as a one-dimensional accumulator array. The number of elements in the array corresponds to the total number of sample points of the model $\|M\|$ times the sample steps of the rotation angle α, n_{angle}. For a fixed reference point, this accumulator array represents the discrete space of local coordinates.

Then only the most voted points are retrieved. After selecting the most relevant locations in the accumulator matrix by a threshold, the actual poses are computed and stored in a list structure. Since many of the estimated poses may have very similar transformations, the clustering method collects estimated poses with similar transformation matrices.

To further increase the accuracy of the final pose estimation, a score recalculation has been implemented. The score recalculation is performed on each point in a cluster and changes the final score of a cluster. After performing the proposed pose transformation in the cluster, a space around each point of the cluster within a certain threshold distance is searched to check for some existing model points. Finding atleast one scene point in the predefined space around the searched point, indicates a correct pose estimation for that point. On the other hand, this may indicate an incorrectly proposed pose transformation if no points from the scene are found nearby. The number of points that have model points within the defined distance is summed, and the score is recalculated according to this sum, [37] .

The code implementation is available on :
https://github.com/marecek199/3DGenerationSegmentationRegistration.git

4 Experiment Results

This chapter presents both partial results for each section and a full test of the segmentation process on a captured dataset. The chapter is divided into 4 sections for a detailed look at each area. This text structure provides enough space to present the results for each section with different settings. The first section presents the results of an artificial dataset generation. This is followed by the results of neural network segmentation. The third section focuses on object registration. The final section presents the complete process, including each process described in the sections, as well as tests on real acquired data.

4.1 Dataset Generation

The algorithm used for this task was not that complicated at first. All that was needed to complete the task was to randomly place more objects in a coordinate space and save the results in a suitable file format. However, as the algorithm was implemented, various flaws were discovered.

4.1.1 Data generation : Initial settings

In Figure 40, we can see the first scene generated by our dataset generator. The image shows scenes with $[3, 7, 15]$ objects in the order. No overlapping prevention has been added as the first phase of data generation.

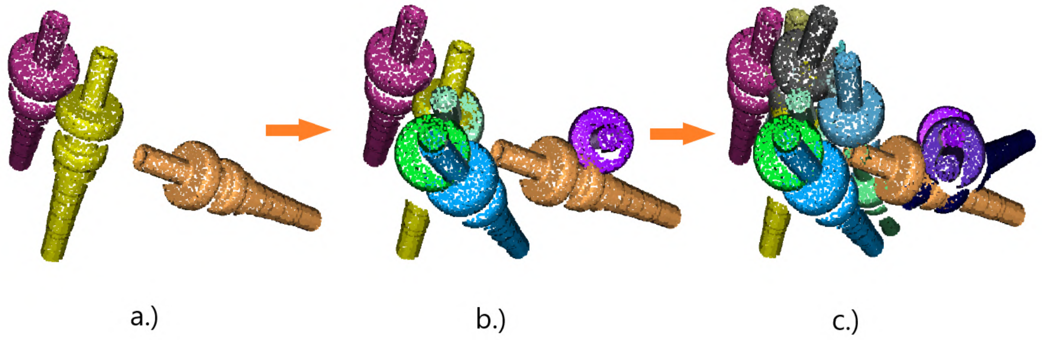


Fig. 40: Data generation default settings : a.) 3 items. b.) 7 items. c.) 15 items

The objects from the default generator are very overlapping. It is obvious that even though the neural network can learn something from a dataset generated by this approach, it would struggle to generalise if the scenes with overlapping objects are given.

4.1.2 Data generation : Second stage of settings

In the second stage of data generation, object overlap prevention has been added, visible in Figure 41. The prevention algorithm is based on object collision detection. Each new object added to the scene is checked at a point level. Based on whether the points of the newly added objects are in collision with existing objects, the new object is added or not.

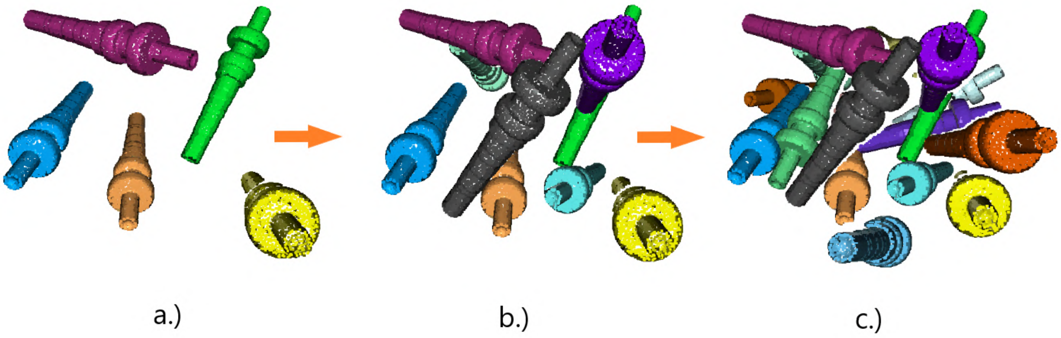


Fig. 41: Data generation no-overlap settings : a.) 5 items. b.) 9 items. c.) 16 items

The objects in the presented scene were generated by an initial settings. Each object was placed in a specific location selected by a random generator using a random distribution. The angles of rotation around each axis and the centre height of an object were chosen in the same way.

4.1.3 Data generation : Final settings

As the results of the previous two settings were analysed, the different probability distributions were added to the data generation. The intervals in one generation of rotation around the x and y axis were changed from $(0, 2\pi)$ to $(0.25\pi, 0.75\pi)$ with a normal distribution. This was done to generate objects mostly in the lying position. The generation of a random rotation around the z axis has been changed to a uniform distribution. The uniform distribution was also added to replace the one in the x, y position to ensure a homogeneous distribution of objects in space and not to create large clusters around the centre of the coordinate system. Results are displayed in Figure 42.

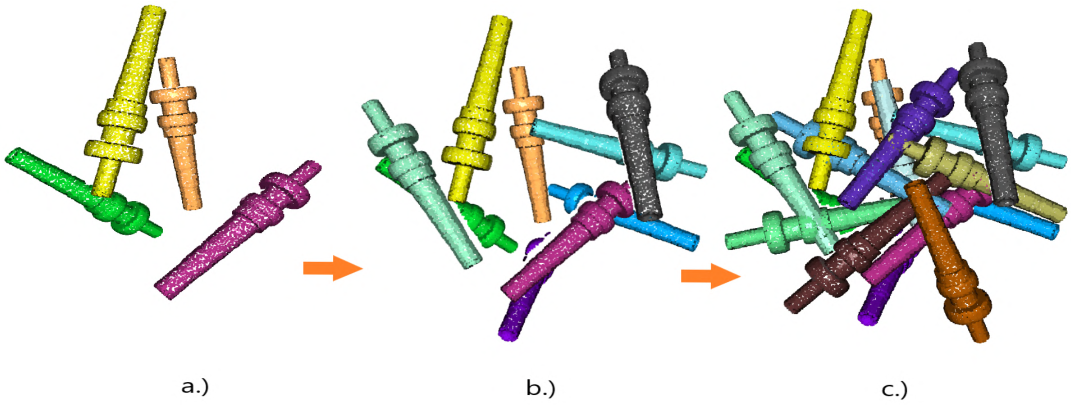


Fig. 42: Data generation final settings : a.) 4 items. b.) 9 items. c.) 15 items

Dataset generator created in this way provides an approach to train the segmentation neural network using only the CAD model of the object. As a consequence of the fact that the creation of the dataset is fully secured by a single algorithm, all point clouds belonging to a single object are simply labeled. The marking of points into segments occurs by color. This procedure completely eliminates the problem of manually labeling the training data. Since this approach provides a way to set object rotations or heights in a certain interval, or to set a minimum or maximum number of objects in the scene, the final results of the resulting dataset can be better adapted to the desired conditions.

4.2 Segmentation

This section presents the results of a data segmentation. The segmentation will be shown on a few different objects, both on test data from the original dataset and on different datasets. In this thesis, the neural network is trained on scenes with (0 : 20) objects. All segmentation results presented here are only compared on scenes with exactly 20 objects. This is a reasonable amount to show the segmentation performance of the neural network and to ensure consistent conditions throughout the comparison.

The next sections show different outputs based on different settings to get a general idea of how the training or prediction settings affect the final output of a segmentation network. All different outputs were trained on the GPU, using batch size 2, due to the small memory of a GPU and the high memory requirements of this particular architecture of the neural network. All trained networks share some variables, such as weight decay or a backbone of a neural network. Some of the network variables are specific to the subjects being trained, such as distance to centre. Due to the fact that the registration algorithm used in this thesis depends on a CAD model, only datasets containing this 3D model are presented. However, the segmentation itself does not require this type of data and works completely without it. As part of the research, the network's ability to distinguish between different objects in different point clouds was also experimentally tested. Since the results were insufficient, the focus was shifted to segmentation of a single object type.

4.2.1 IPA Gear shaft

First segmented object in this section is a gear shaft model from a IPA dataset, visible in Figure 43 .

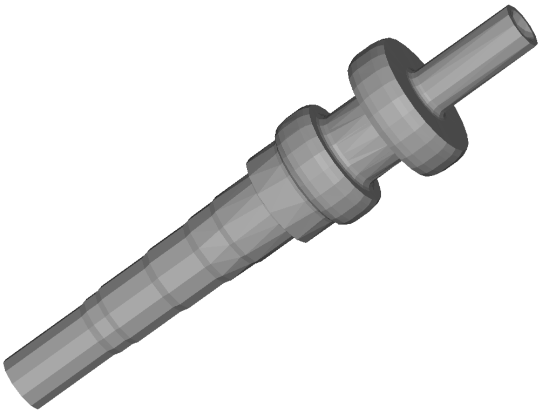


Fig. 43: IPA Gear shaft

Comparison of different results affected by a distance from centre variable

Although this object is not ideal for segmentation training on this particular neural network because of the centre score condition. Because the topology of the object body is not nearly homogeneous in all directions, the centre score must be set higher to avoid separating far edges from the object body. If the distance from centre threshold is set very low or high in the prediction, the neural network will produce false results. The final segmentation with both high (a.) and low maximum (b.) distance from a centre R_{max} can be seen in Figure 44 .

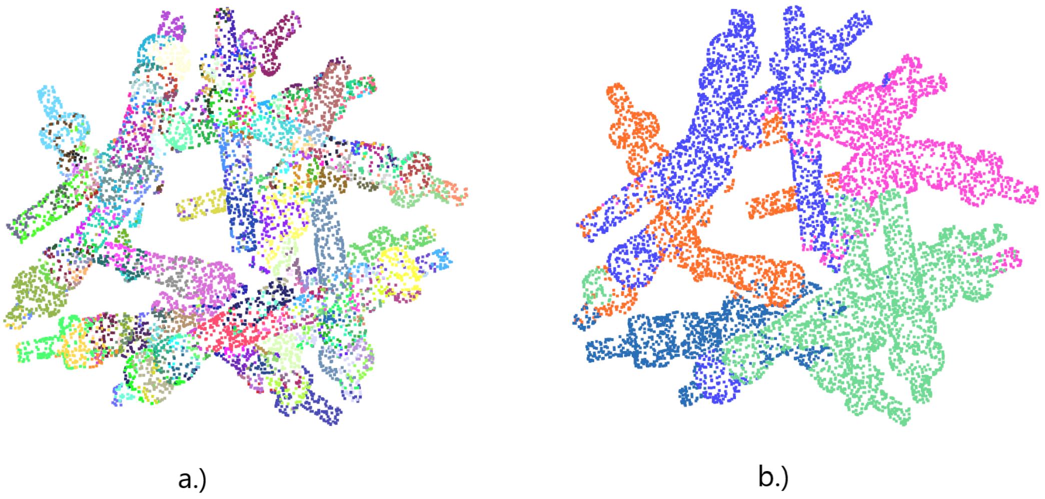


Fig. 44: Variable distance from center : a.) Too small. b.) Too big

Comparison of different results affected by a center score threshold variable

The centre point threshold is also very important. If set too low, many false positive segments will be revealed. On the other hand, setting it too high could result in a very under-clustered segmentation. The final results of both under-clustered (b.) and over-clustered (a.) results, set by a variable *'center_socre_th'*, can be seen in Figure 45 below.

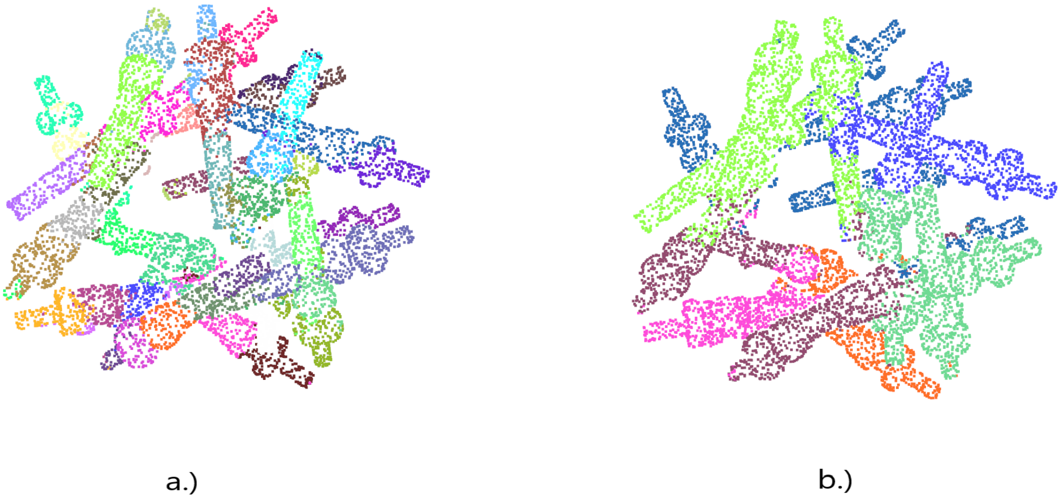


Fig. 45: Variable center score percentage : a.) Too small. b.) Too big

Results of a NN trained on a IPA dataset

In Figure 46 below, you can see the scene with a randomly placed gear shaft in the box. On the left (a.) is the input data for a neural network. The image shows that the segmentation of this neural network is not based on colour at all. On the right side (b.) of figure, the ground truth data are displayed.

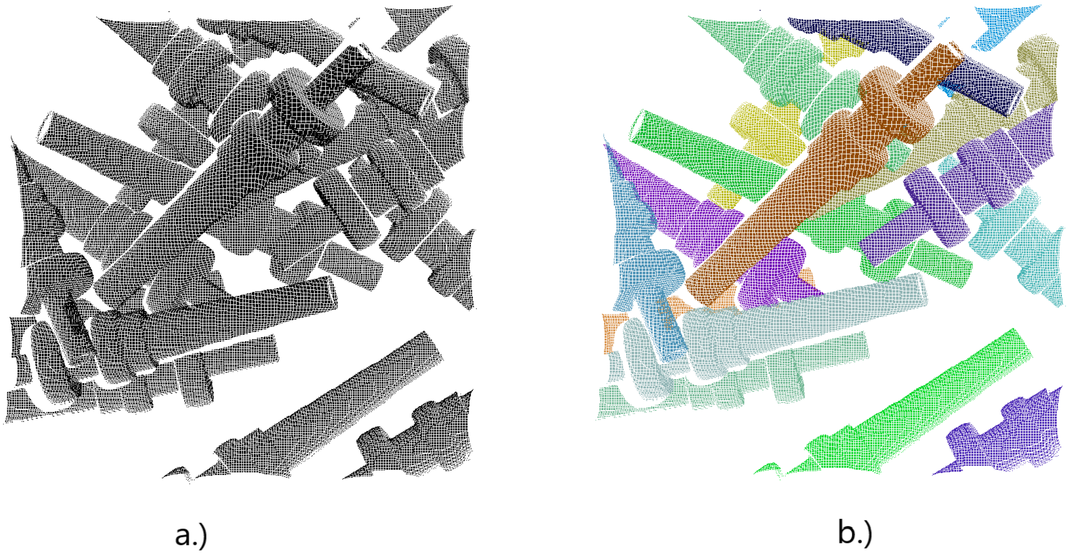


Fig. 46: Gear shaft : a.) Input data. b.) Ground truth data

The training of this neural network was performed on a IPA dataset. The prediction results of the trained neural network can be seen in Figure 47 below. Both views show that the network can be trained on the IPA dataset. The segmentation results seem to be excellent with very few errors.

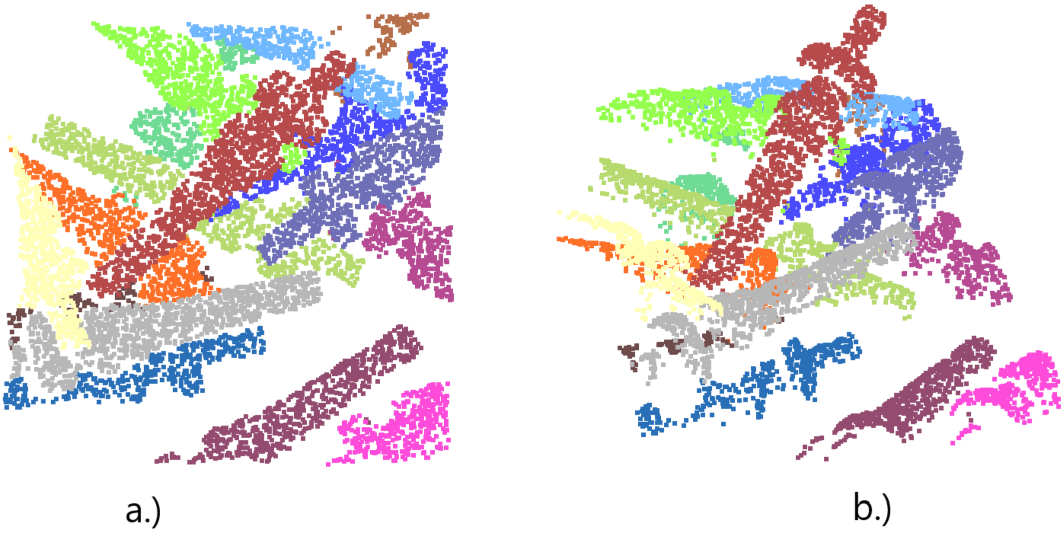


Fig. 47: Prediction on IPA data : a.) Top view. b.) Side view.

The trained network was also tested on FPCC data, which is not part of the IPA dataset, but contains a trained gear shaft object. The segmentation results are not so good, with many small and large errors, visible in Figure 48. The distribution of the objects in space is not much similar to the ones, the neural network was trained. This problem can be reduced by using a training environment similar to the one, on which the neural network will make predictions. The error can also be reduced by training on a larger dataset, which could increase the network's ability to generalise. On the other hand, training on a larger dataset increases the total training time.

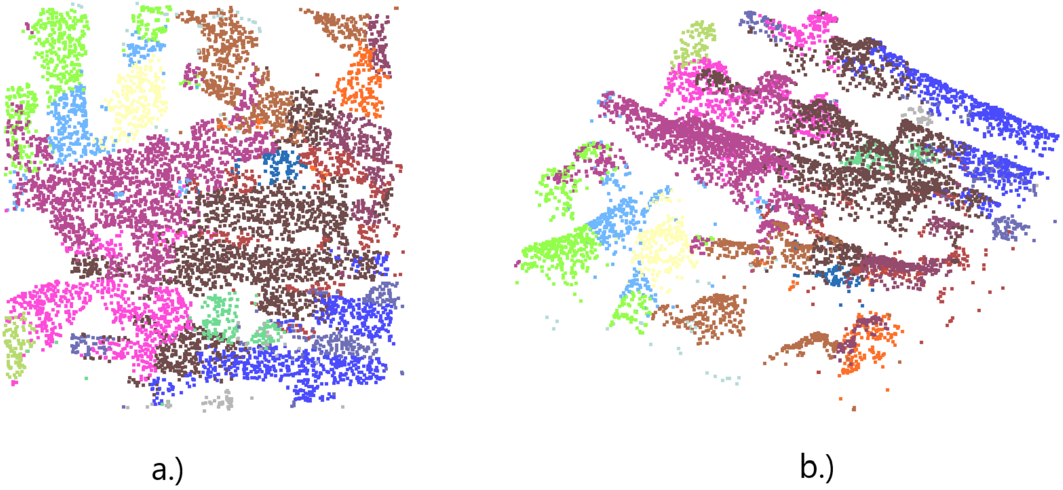


Fig. 48: Prediction on FPCC data : a.) Top view. b.) Side view.

Results of a NN trained on a generated dataset

The next three figures show the prediction on different datasets of a neural network trained on an artificially generated dataset. The results of a neural network on each dataset presented, will mostly vary from prediction to prediction due to the stochastic part of the neural network. The next Figure 49, displays the ground truth data (b.). As was mentioned in the previous paragraph, the left side (a.) of this figure represents the input data to the network, and the right side (b.) represents the true labelled data.

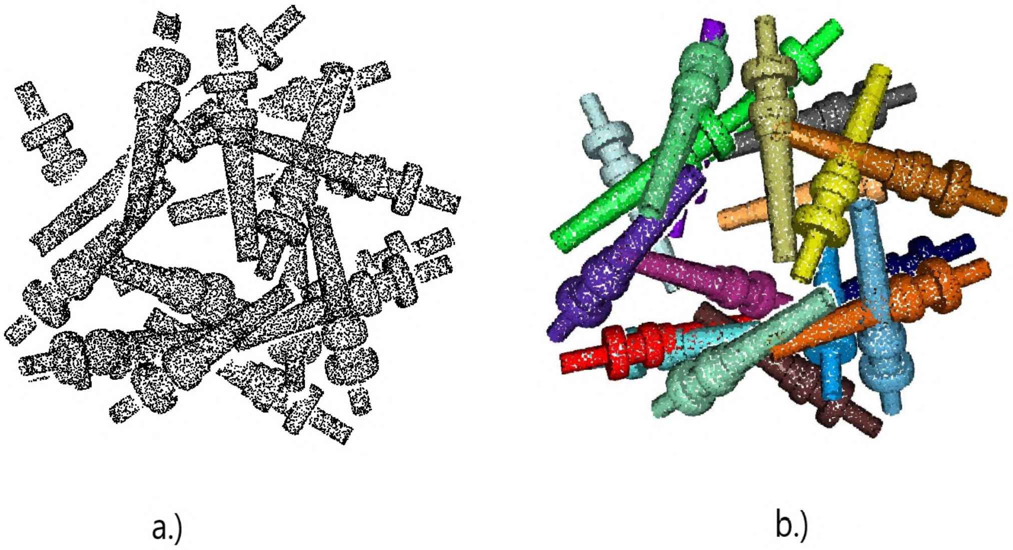


Fig. 49: Gear shaft : a.) Input data. b.) Ground truth data

The prediction made by the trained network can be seen in Figure 50 below. The image shows the prediction results of the segmentation on the proposed point cloud. In general, the results can be described as more than good, even though there are incorrectly defined segments. Despite larger or smaller errors, the neural network is able to segment most objects regularly and reliably on a test data that is part of the generated dataset.

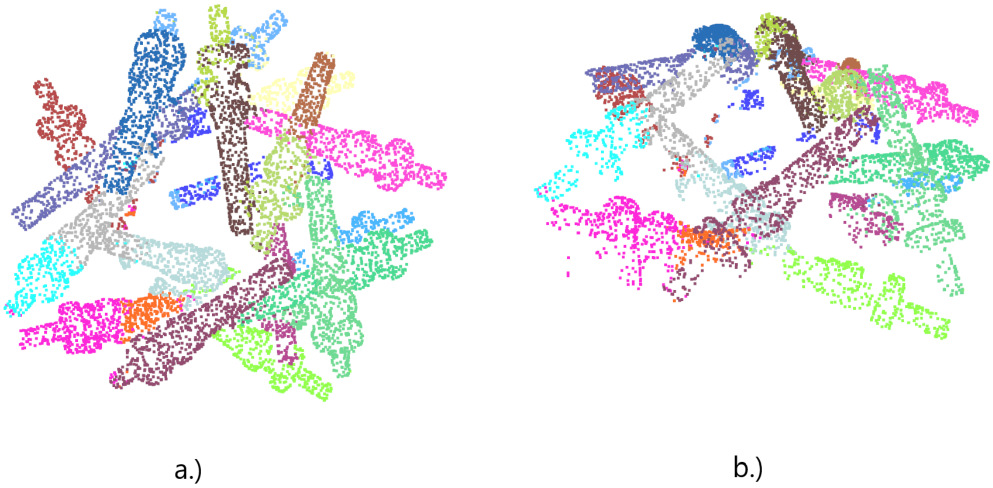


Fig. 50: Segmentation on testing data : a.) Top view. b.) Side view.

The results of the neural network trained on generated data and performing predictions on a FPCC dataset can be seen in Figure 51. Obviously, the results are worse. However, the network mostly underclusters the input point cloud. Since this type of error predicts fewer segments than it should, it can mostly be solved in the following section of the thesis.

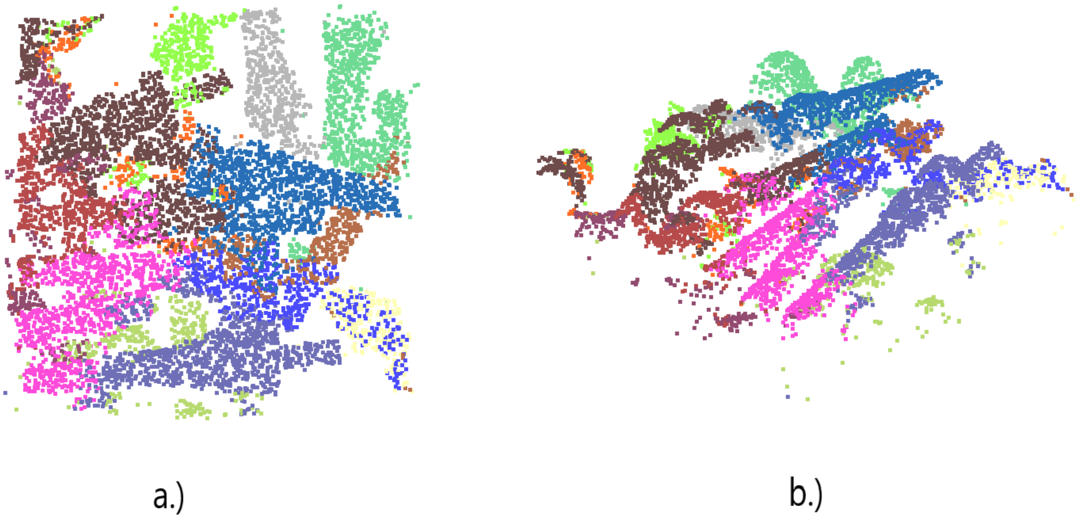


Fig. 51: Segmentation on FPCC data : a.) Top view. b.) Side view.

4.2.2 IPA Ring screw

The proposed neural network was also trained on different datasets, one of which was also an IPA dataset consisting only of ring screws, which model is on Figure 52. The ring screw is a different object from the gear shaft in terms of topology and geometry. Unlike the gear shaft, the ring screw also contains a hole.

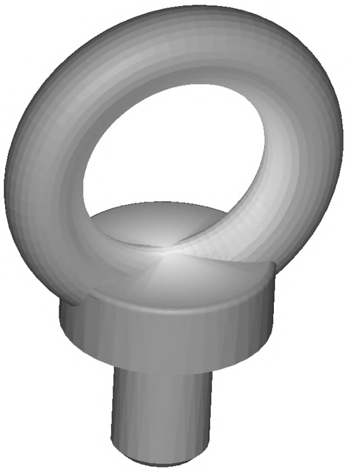


Fig. 52: IPA Ring screw

Results of a NN trained on a IPA dataset

Analogous to the previous subsection, the neural network is first trained on a IPA dataset and then on a generated dataset. Figure 53 shows a correctly split point cloud on the right (b.) and an image of the input data on the left (a.).

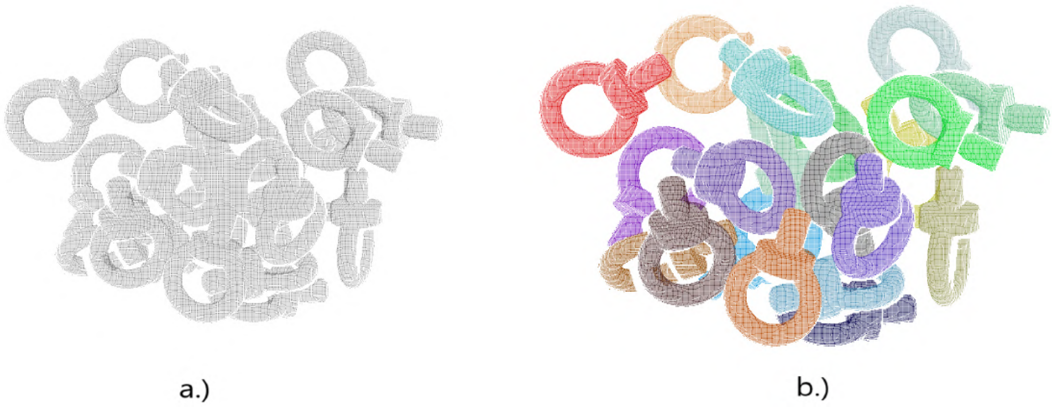


Fig. 53: Ring Screw : a.) Input data. b.) Ground truth data

The prediction output of a segmentation network trained on IPA data is shown in Figure 54 below. The segmentation results also look very promising on the ring screw. Figure shows that the segmentation network can cope very well with holes in the body of the object. Even though there are some small errors, it can be assumed that this structure of the segmentation network can cope very well with different object geometries.

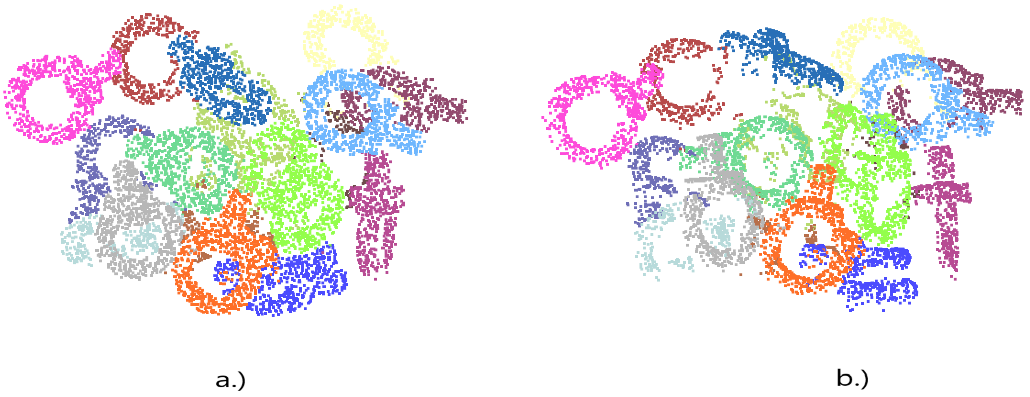


Fig. 54: Segmentation on testing data : a.) Top view. b.) Side view.

As the author do not have an FPPC dataset containing a ring screw to test the performance of the neural network, an artificially generated dataset was used for testing purposes. The next Figure 55, shows the result of a network segmentation prediction.

The result contains several incorrect assignments, but as mentioned in the text above, a larger dataset might reduce this error.

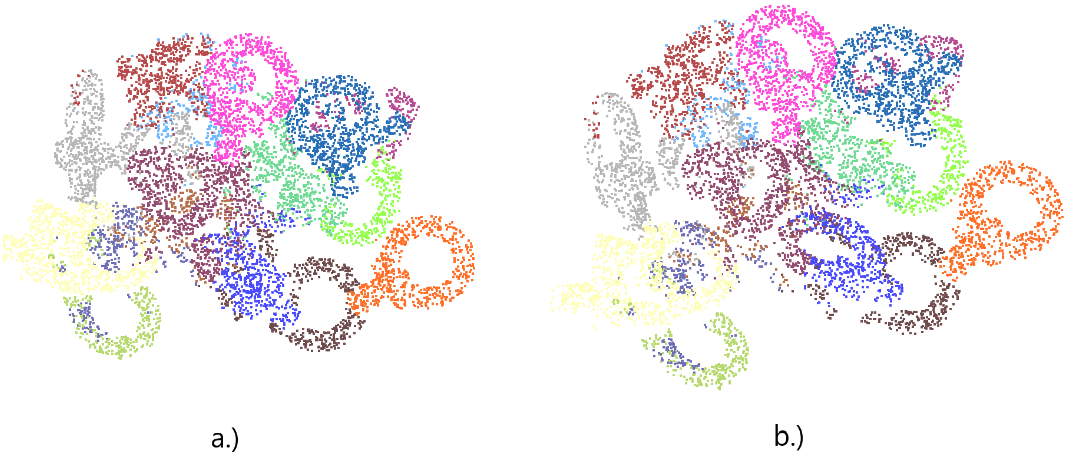


Fig. 55: Segmentation on generated data : a.) Top view. b.) Side view.

Results of a NN trained on a generated dataset

The next Figure 56, shows the ground truth data on the right (b.) and the input data on the left (a.). The results of the segmentation network were also tested in reverse, the neural network was trained on a custom artificial dataset and tested on test data from this dataset as well as on public data from an IPA dataset.

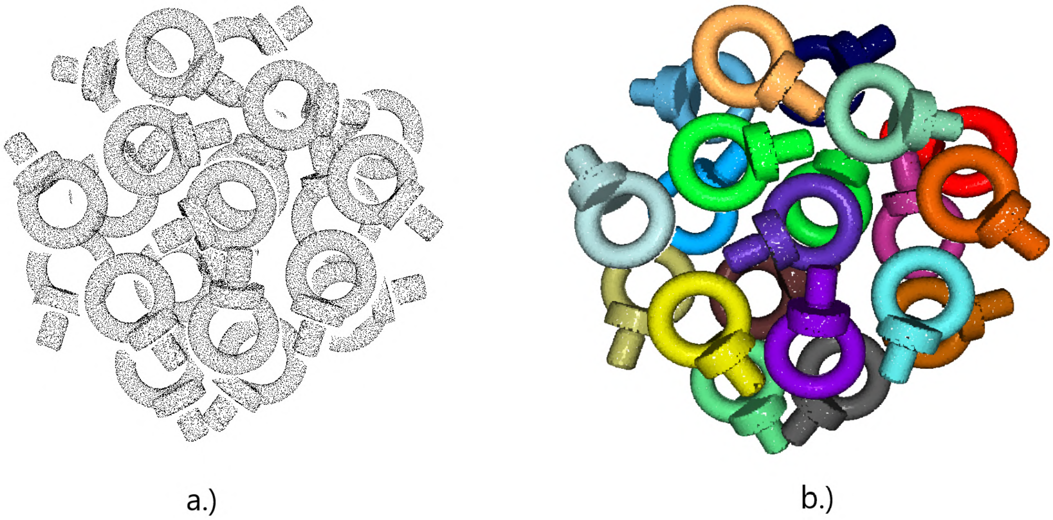


Fig. 56: Ring Screw : a.) Input data. b.) Ground truth data

The network's predictions look stable and with very few errors in Figure 57. This means that the network architecture has no problem learning the basics on both acquired and generated images.

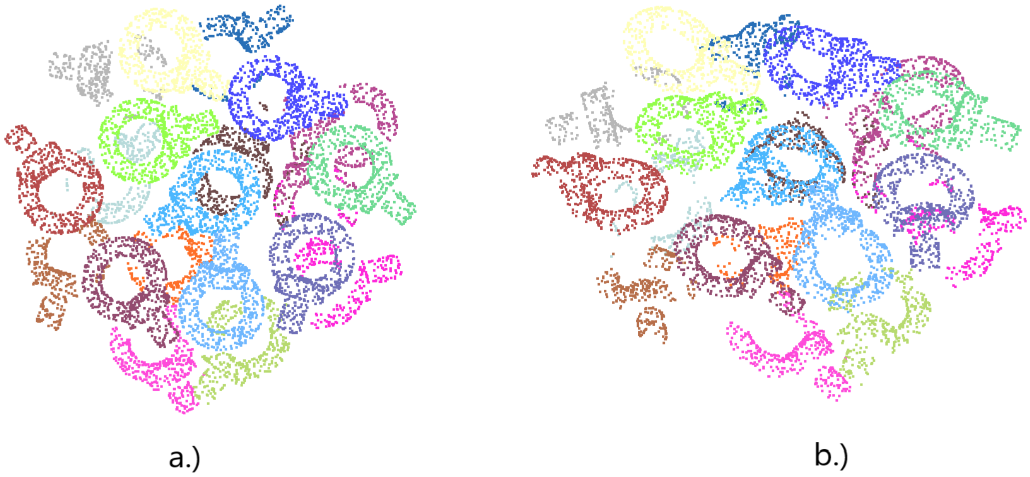


Fig. 57: Segmentation on testing data : a.) Top view. b.) Side view.

The results on the other data are a bit worse, in Figure 58, with bit more errors. Anyway, the network still does its job to segment whole point cloud into smaller segments.

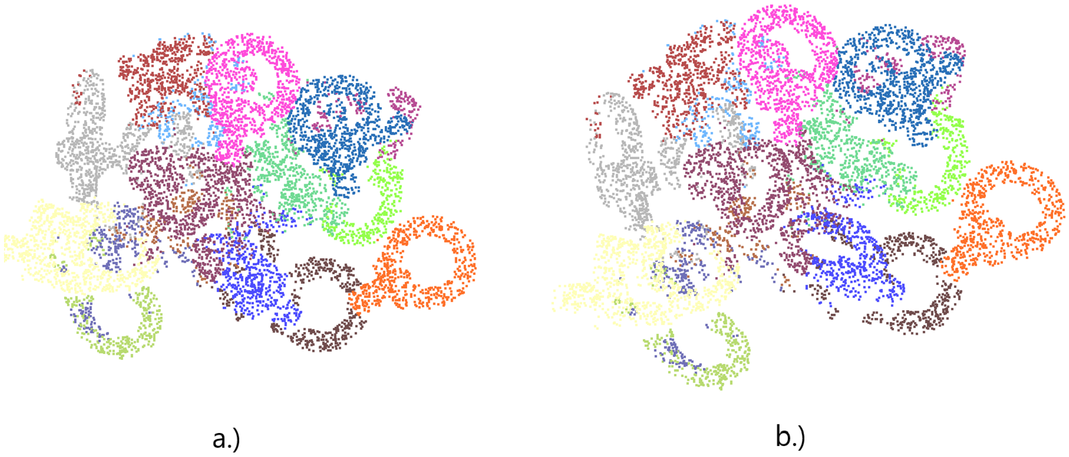


Fig. 58: Segmentation on generated data : a.) Top view. b.) Side view.

The performance comparison between on different objects is shown in **??**. As expected, the segmentation results on the training data perform better than on other datasets. This fact is underlined by the graphical results shown above.

Object	Trained on	Prediction on	Precision [%]	Recall [%]
Gear Shaft	Generated data	Testing data	75.2	75
	Generated data	Public FPCC	39.6	32.7
	IPA data	Testing data	68.5	68.5
	IPA data	Public FPCC	28.6	33.3
Ring Screw	Generated data	Testing data	72.4	74.3
	Generated data	IPA data	39.8	35.1
	IPA data	Testing data	69.6	70
	IPA data	Generated data	50	45

Tab. 1: Segmentation results on different data

4.3 Registration

This section presents the results of the registration. The registration is performed on the output of a segmentation neural network. For presentation purposes, a few segmented clusters from a segmentation part are shown and a CAD model of an object is registered to this segment. The main reason for this part of the algorithm is to find the exact position and rotation of a CAD object in the segmented point cloud. The accurate pose estimation of an entire object, and not just a visible segmented body part, will provide complete information for a robotic gripper. In the next subsections, the final registration for both the gear shaft and the ring screw from an IPA dataset is presented. The registration is performed on three segmented surfaces, as shown in the images for each object. Since this algorithm works better when working with coordinate values greater than one, i.e. not normalised, the resulting segments from the segmentation mesh are up-scaled. After performing the registration part on a segment, the transformed model is downscaled.

4.3.1 Registration on : IPA Gear Shaft

Figure 59, shows the raw output of a segmentation mesh. The more highlighted areas represent three segmented areas that will be aligned by a CAD model of a gear shaft. The main task is to take a CAD model shown in Figure 43, which by default which is in the centre of a coordinate system, and to align it to all three segments shown.

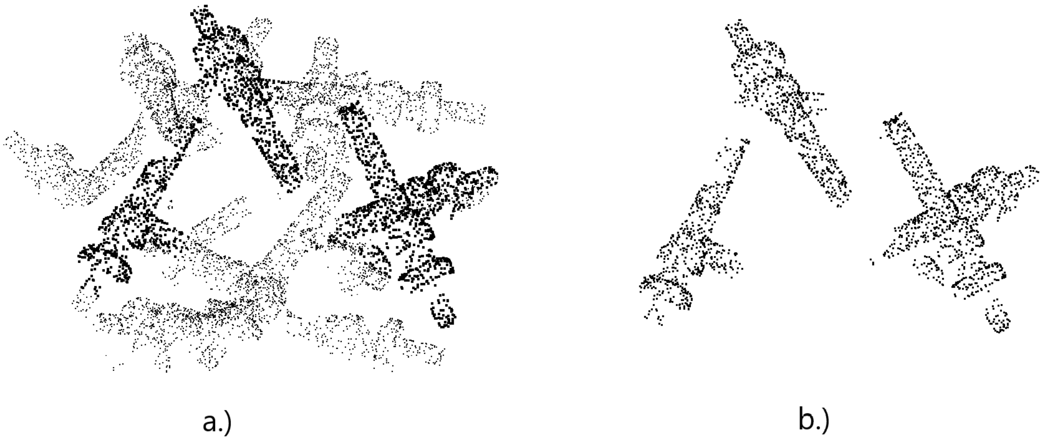


Fig. 59: Gear shaft Registration : a.) Highlighted segments. b.) Isolated segments

Registration of the first segment

The process of aligning the first segment is shown in Figure 60. The output segment shows a clustering error because it contains two gear shafts instead of one. As mentioned in the text above, the registration algorithm can actually solve this underclustering problem. The algorithm chooses the best match to align the gear shaft model with the segmented area, ignoring the outliers or the other gear shaft objects.

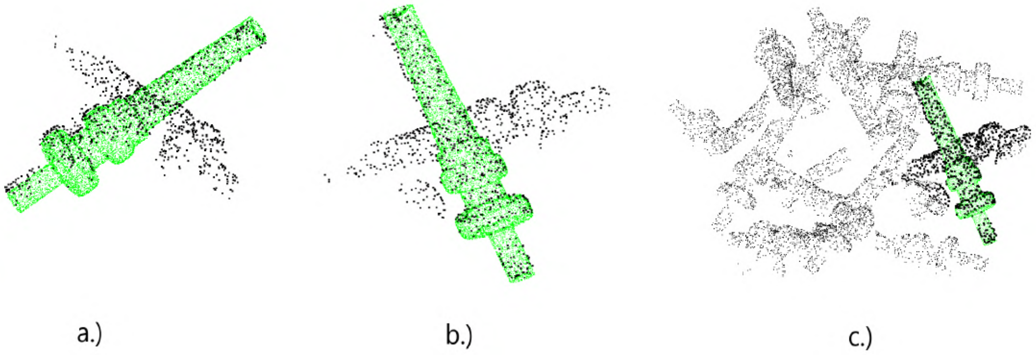


Fig. 60: Gear shaft Registration : a.) Isolated view 1. b.) Isolated view 2. c.) Highlighted in the point cloud.

Registration of the second segment

The second gear shaft object, representing the PPF registration, is located at the top of the box. There are some minor imperfections in the registration. Figure 61, shows the final registration from two different views and the view of the alignment of the CAD model in the scene.

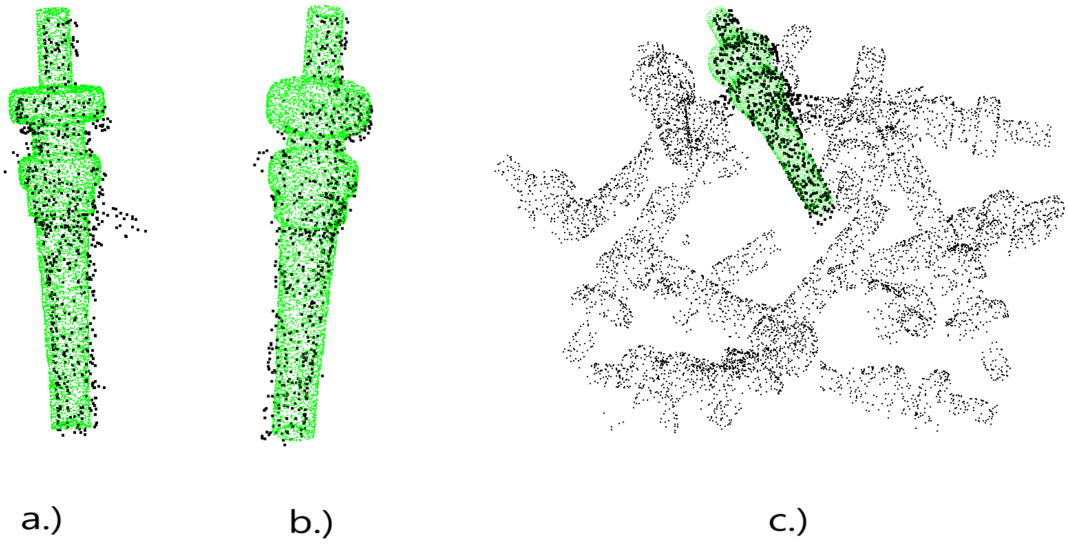


Fig. 61: Gear shaft Registration : a.) Isolated view 1. b.) Isolated view 2. c.) Highlighted in the point cloud.

Registration of the third segment

The third registration of a gear shaft shows very similar results to the first two. The PPF algorithm in Figure 62, shows precise registration and very fine alignment in both focused views as well as in the environment with other gear shafts in the box.

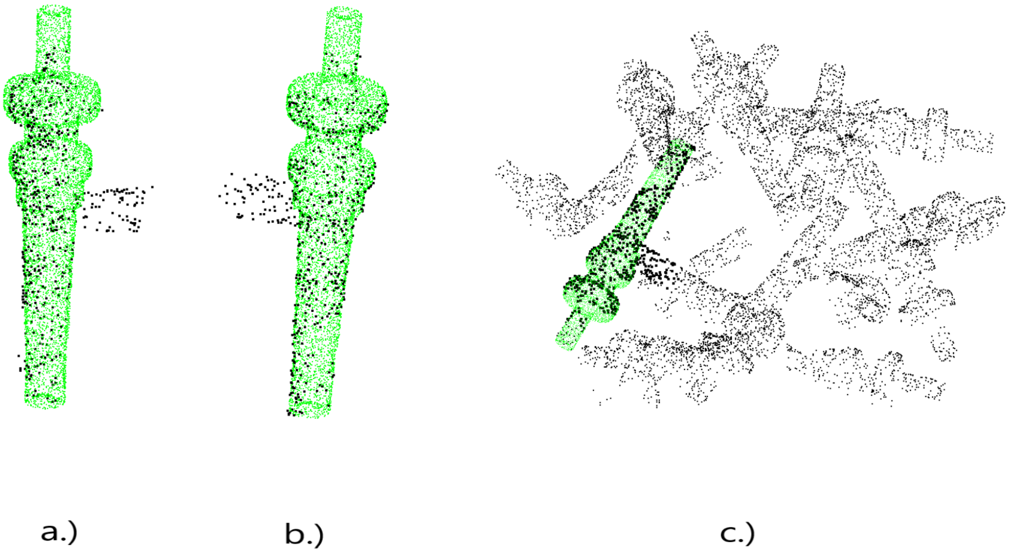


Fig. 62: Gear shaft Registration : a.) Isolated view 1. b.) Isolated view 2. c.) Highlighted in the point cloud.

Final registration of all three segment

The final summary of all three registrations is displayed in Figure 63. On the left side (a.) are highlighted all three segments, with final gear shaft alignment on the right (b.).

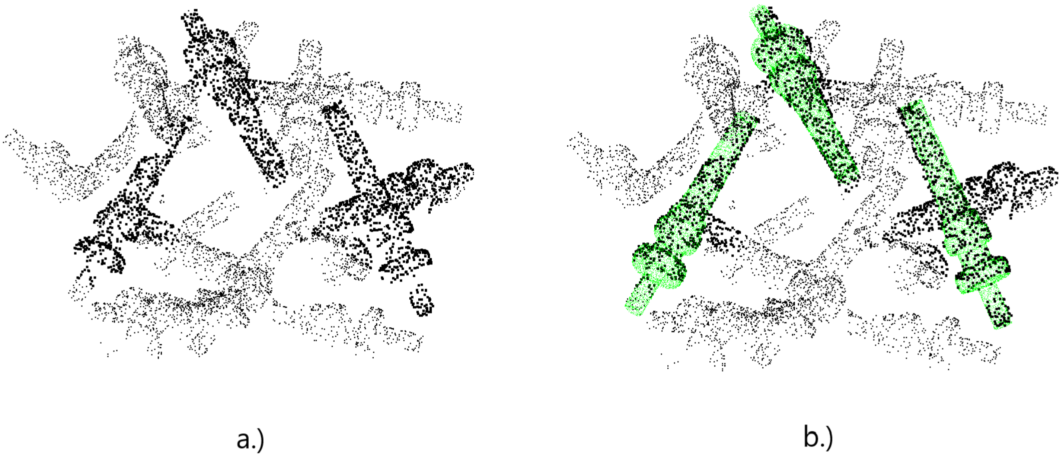


Fig. 63: Gear shaft Registration : a.) Highlited segments. b.) Aligned models into segments

4.3.2 Registration on : IPA Ring Screw

A similar registration is performed on the IPA ring screw object. Obviously, the fine tuning can be performed on all the segmented areas of the segmentation network, but the results within an object would be very similar. The point cloud on the left side (a.) of Figure 64, shows the output of the neural network in one colour. The highlighted areas represent all three segments selected for registration.

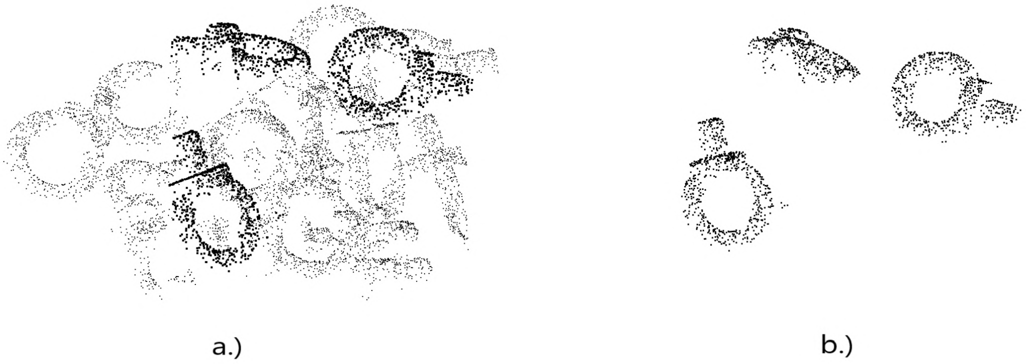


Fig. 64: Ring screw Registration : a.) Highlighted segments. b.) Isolated segments

Registration of the first segment

As can be seen in Figure 65, the PPF algorithm is not very dependent on the topological and geometrical level of an object. Thanks to this fact, the final alignment of a ring screw from both views is fine, with a small error in the leftmost view. The right view shows the alignment with other ring screws.

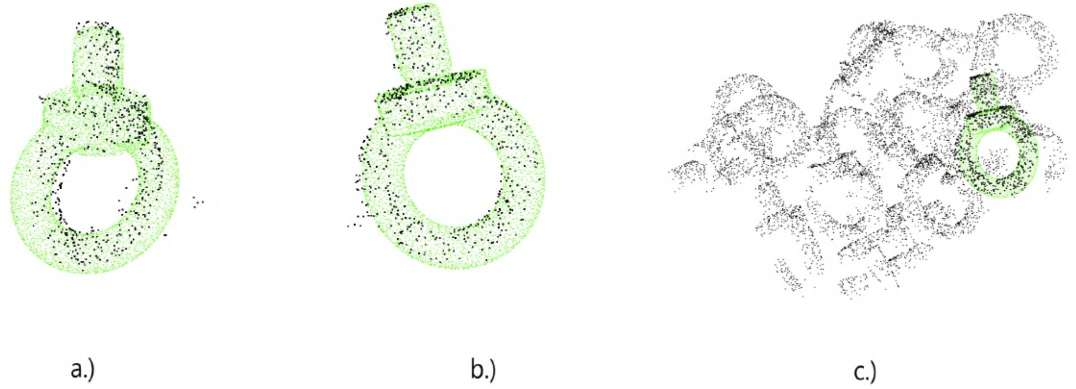


Fig. 65: Ring screw Registration : a.) Isolated view 1. b.) Isolated view 2. c.) Highlighted in the point cloud.

Registration of the second segment

The next Figure 66, displays the alignment on the second segment. The final ring screw co-ordinates with its rotation in space provide the stable results required for bin-picking.

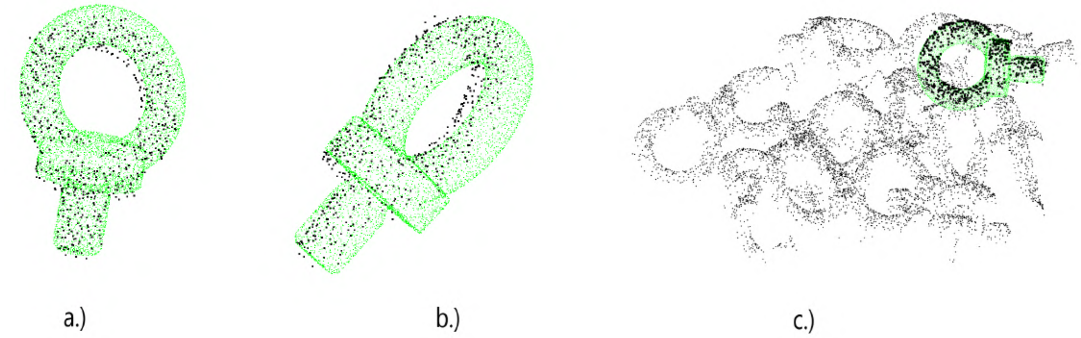


Fig. 66: Ring screw Registration : a.) Isolated view 1. b.) Isolated view 2. c.) Highlighted in the point cloud.

Registration of the third segment

The final registration of the third ring screw on the third segments is exposed in Figure 67 below. Very similar results can be evaluated from the third alignment with a stable characteristic.

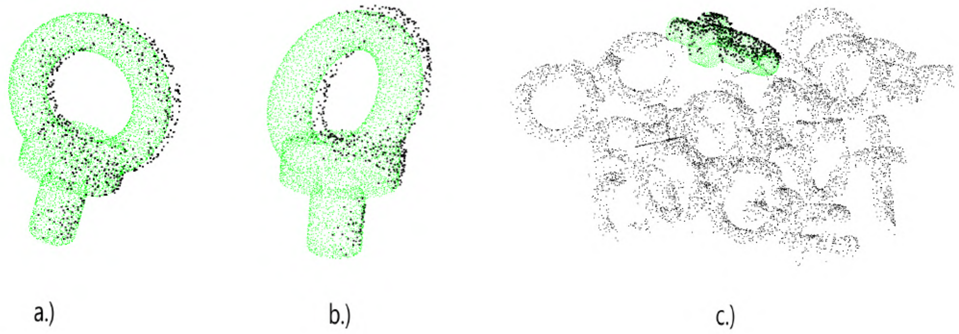


Fig. 67: Ring screw Registration : a.) Isolated view 1. b.) Isolated view 2. c.) Highlighted in the point cloud.

Final registration of all three segment

The results of the PPF registration end with the final Figure 68. As in the previous subsection, the left part of the image (a.) highlights the three selected segments that have been aligned and the results are present in the form of text and image above. The right side (b.) shows all three segments with their final and correct registration, demonstrating the robustness and stability of a proposed algorithm to work with different objects in different environments.

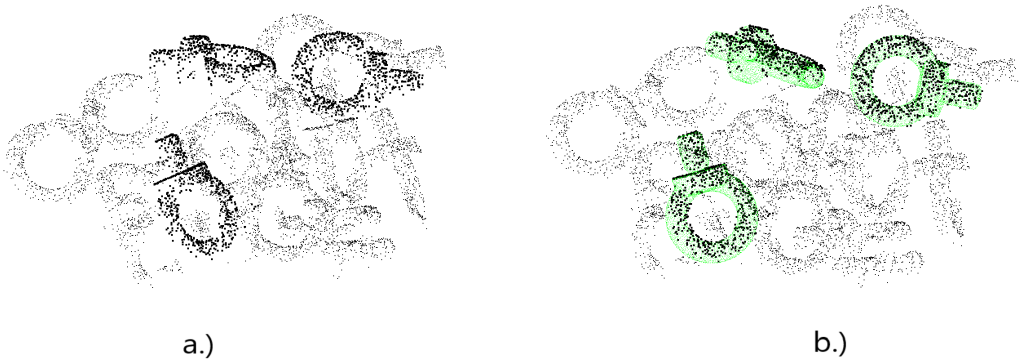


Fig. 68: Ring screw Registration : a.) Highlited segments. b.) Aligned models into segments

4.4 Complete process on acquired data

Since the aim of our thesis was to make this solution applicable in practice, the presented solution method is tested on a dataset created with the Zivid One+ camera. The complete dataset consists of 20 point cloud scenes, each containing seven objects freely distributed in space. The tested object is a T-join pipe displayed in Figure 69.

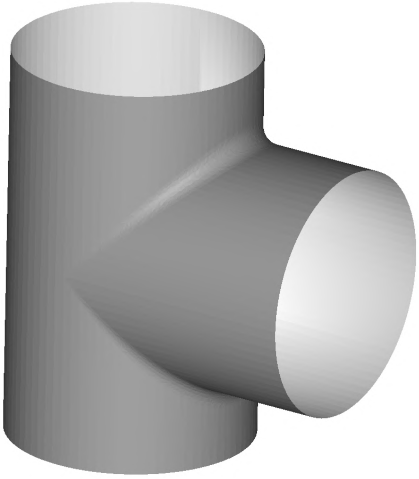


Fig. 69: T join CAD model

The captured scene can be seen on the left side (a.) of Figure 70. The right side (b.) displays the scene after the most surface was filtered by a threshold.

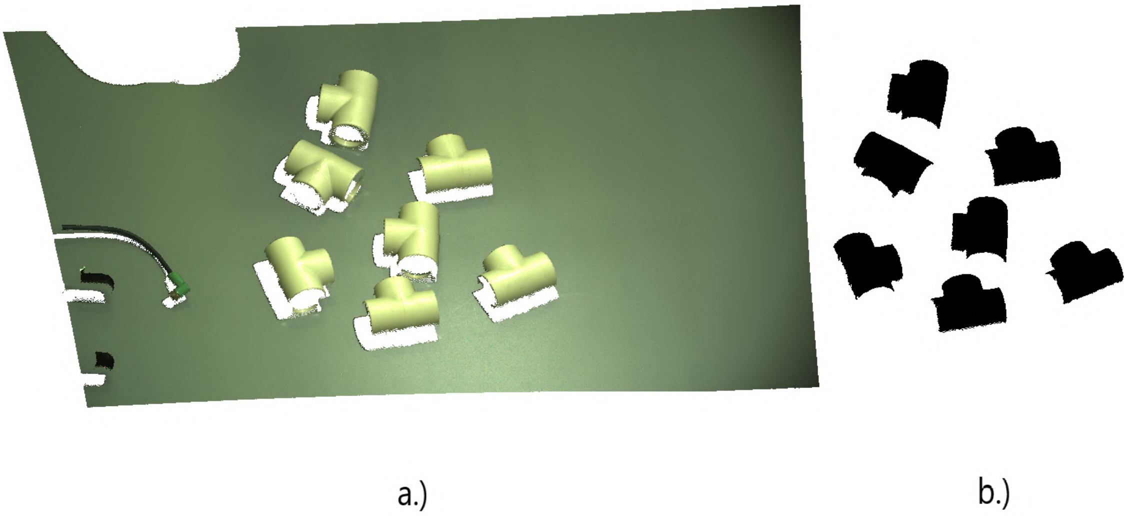


Fig. 70: Captured 3D environment : a.) Actual output from the camera b.) Point cloud after filtering the surface

4.4.1 Data generation

The availability of the CAD model of a segmented T-join allowed the creation of an artificial dataset. This allowed the segmentation network to adapt its internal parameters needed for correct prediction, rather than just guessing regions from previous training on other objects.

As in the previous experiments, the dataset generation went smoothly. Only [1 : 10] objects were simulated to generate a proper dataset for this specific task within a 15 × 15 cm area box.

The result scenes can be seen in Figure 71, and in Figure 72 below.

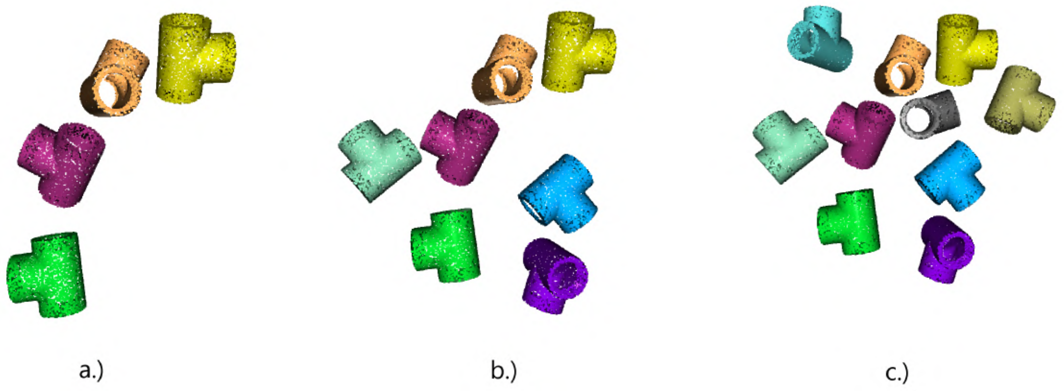


Fig. 71: Generating scene 1 : a.) 4 items. b.) 7 items. c.) 10 items.

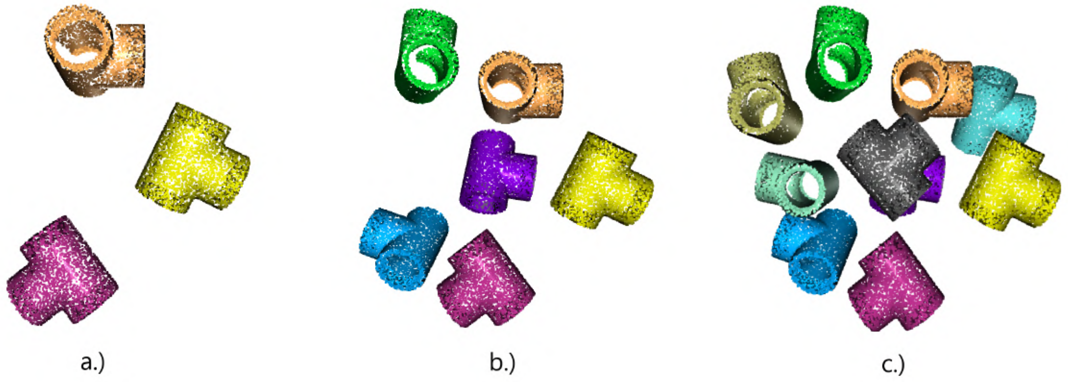


Fig. 72: Generating scene 2 : a.) 3 items. b.) 6 items. c.) 10 items.

4.4.2 Segmentation

Before segmentation, the point cloud was cleaned of the plane on which the objects were placed. The result can be seen in Figure 70 b.) and was achieved by keeping only the points that were within a certain distance, threshold. The results of the segmentation on the acquired data look very good. Only small errors appear on two of the seven segmented objects in Figure 73. Since the neural network has been fully trained only on the generated data shown above, it can be said that a neural network trained in this way can, in principle, used directly for practical applications in industry. Since our dataset was created with units of meters, but the output from the 3D camera is in millimeters, the created 3D image of the scene had to be scaled down.

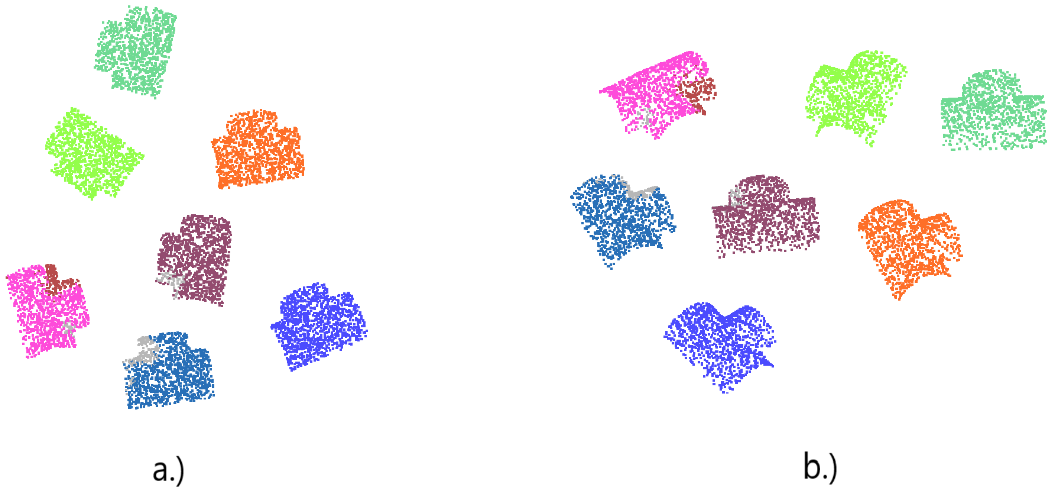


Fig. 73: Segmentation on captured point cloud : a.) Top view. b.) Side view.

4.4.3 Registration

Figure 74 below, shows all seven registrations for each part. Each of the T join registrations has a similar characteristic. It can be seen that the registration results appear to be stable on the recorded data. The only thing that was changed in the algorithm was a threshold for the score recalculation part in the final phase of the algorithm.

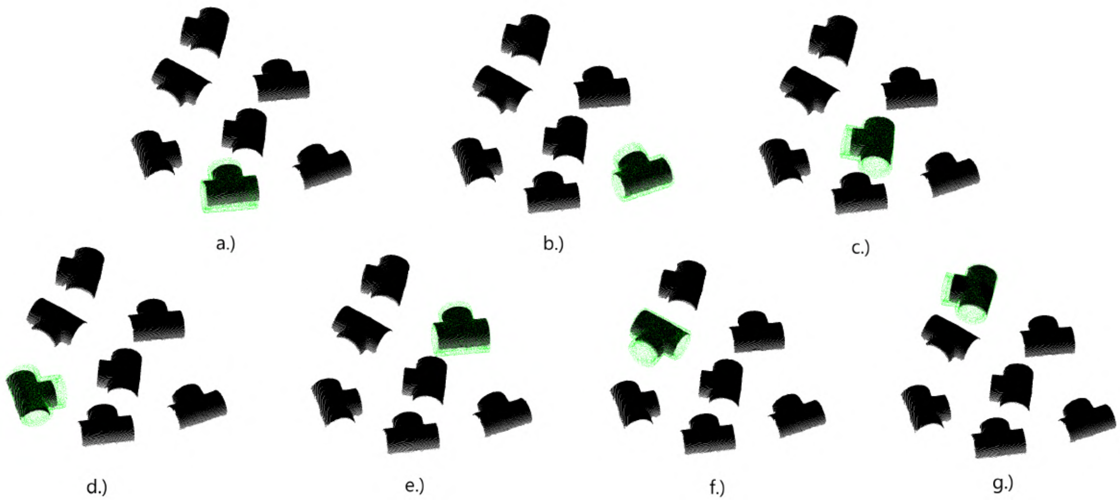


Fig. 74: Model registration on segments : a.) Segment 1. b.) Segment 2. c.) Segment 3. d.) Segment 4. e.) Segment 5. f.) Segment 6. g.) Segment 7.

Since the partial results on each presented object look very good, the final registration is displayed in Figure 75, for each object also looks good. Based on these facts, the results show stability and fine tuning in the registration process as well as in the segmentation process. It should not be forgotten, that the registration and segmentation were performed on the basis of a neural network trained purely on an artificially generated dataset.

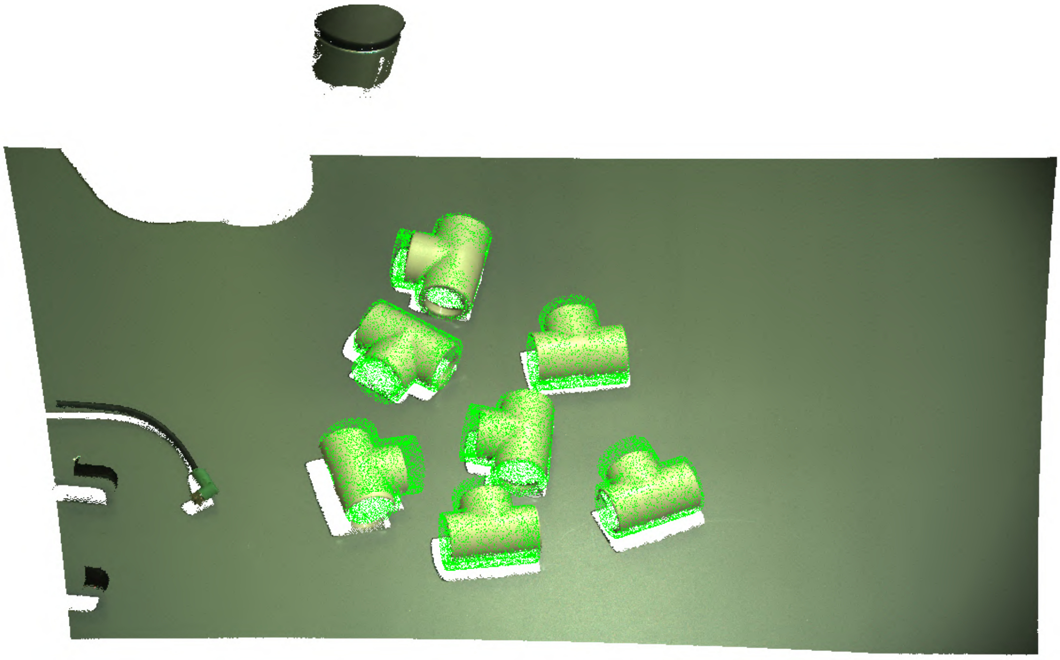


Fig. 75: Final registration on captured data

5 Discussion

The introduction part of the thesis provides an overview of the technologies that are currently available, and used to create and process 3D point clouds. The paper aims at an analysis and principles of different algorithms that focus on segmentation and registration of objects in the point cloud. The overview makes it possible to identify strengths and weaknesses of the different approaches and to propose a new approach that exploits this best of the available methods.

One of the main advantages of this research is a development of an algorithm for generating a custom dataset. This dataset allows to generate specific scenes that can be adapted to different practical applications. This includes, for example, embedding the generated object, selecting the maximum number of objects in the scene, the size of the generated scene, the maximum height objects, or adjusting the orientation of objects in space.

This thesis has studied the detailed theoretical descriptions and implementations of specific stand-alone algorithms for segmentation and registration of objects in a point cloud. The segmentation algorithm, which is based on deep learning, represents a modern approach to solving this problem. The added registration algorithm is based on geometric properties. Robustness of registration algorithm, allows using the registration part without the need for segmentation.

The resulting system created by combining both the above-mentioned algorithms, benefits from the advantages of both methods. The registration PPF algorithm can efficiently eliminate most of the segmentation flaws that can happen using a neural network. Further, due to a very fast FPCC segmentation based on deep learning, the registration algorithm is able to process the results within seconds.

The third chapter of the thesis focuses on the presentation of the experimental results that were obtained while testing each algorithm. The algorithms were tested on data that was either generated as part of this research or obtained from publicly available sources. It was shown how the different algorithms perform on different types of objects. Obtained results vary depending on different settings of algorithms. This part of the thesis thus provides a comprehensive view of the functionality and flexibility of the designed algorithms.

The key areas for improvement presented solutions include a more efficient implementation of the PPF registration algorithm, which could significantly improve the overall efficiency of the system. Parallelization of this algorithm would likely reduce computational time. Another possible area for improvement lies in data generation, where current approaches to collision detection encounter limitations when generating a large number of objects in a limited area. An extension that could also be considered is the use of a registration algorithm for an automatic labelling of 3D objects in the point cloud. This possibility would allow offline datasets to be generated from directly measured data and further enrich the training dataset.

The results of the conducted research showed that accurate segmentation and registration of objects in the 3D point cloud can be achieved for industrial bin-picking, which is a great benefit in the field of robotic vision and machine learning. Despite the fact that there is a continuous space to improve, this work opens the way to new possibilities and brings valuable insights for future research in this area. The outputs of this thesis can be easily built in already used automatic lines and industrial cameras and spare much finances for industrial companies.

6 Conclusion

The aim of the thesis was to deal with 3D point cloud segmentation for industrial picking of spatially disordered parts from a bin, which is the key problem in process automation using robotic vision systems. The structure of the thesis consists of three logically connected chapters, in which all aspects of this problem are discussed from different points of views.

The first chapter focuses on theoretical foundations of computer, machine and robotic vision, and how these areas support the segmentation and registration of objects in a point cloud. This chapter elaborates on the knowledge of pick and place and bin picking issues, and outlines the possibilities of reconstructing 2D and 3D scenes into a point cloud. In this section there are also analyzed principles of algorithms aimed at segmentation and registration of objects in the point cloud.

The second chapter is devoted to the actual solution of point cloud segmentation. In the beginning of the chapter the procedure of generating labeled objects was presented in the form of a point cloud for neural network training segmentation. The selection and implementation of the machine learning based segmentation algorithm is further described. The conclusion presents the selection and thorough implementation of the registration algorithm that processes segments from the neural segmentation network based on geometric features.

The third chapter presents the experimental results using algorithms presented in the theoretical section and their analysis of different parts. Consequently, these algorithms were applied to both generated datasets and publicly available datasets. The functioning of the algorithms was presented on different objects with various settings. The whole process from the generation of the custom training data, through the segmentation of the self-collected data using a 3D camera, to the registration of the object's CAD model in the 3D point cloud scene was closely presented in this chapter.

The result of this thesis is the progressive 3D point cloud segmentation method that has full potential to significantly increase the efficiency of bin-picking by industrial robots. The generation of custom 3D datasets allows customization of the initial training conditions as much as possible. Machine learning-based segmentation provides a very fast and accurate processing of the input point cloud. The complexity of the whole system is underlined by robust model registration, which can very efficiently find transformation matrices for models in each segment. The efficiency of the registration method is precisely achieved by using segmentation as an additional step before model registration. This solution is so robust and reliable so it can be used in many industrial automated lines without the necessity of using expensive softwares or human operators.

BIBLIOGRAPHY

[1] BANOULA, Mayank. What is Tensorflow? Deep Learning Libraries and Program Elements. *Simpli Learn* [online]. 2023, 16.2.2023 [cit. 2023-05-06]. Dostupné z: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow

[2] BANSAL, SHIVAM. 3D Convolutions : Understanding + Use Case. *Kaggle* [online]. 2019, 2019 [cit. 2023-05-06]. Dostupné z: https://www.kaggle.com/code/shivamb/3d-convolutions-understanding-use-case

[3] DOGGETT, Stetson. What Are Point Clouds, And How Are They Used?. *Dronegenuity* [online]. [cit. 2023-05-20]. Dostupné z: https://www.dronegenuity.com/point-clouds/

[4] DROST, B, M ULRICH, N NAVAB a S ILIC. Model globally, match locally: Efficient and robust 3D object recognition. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [online]. IEEE, 2010, s. 998-1005 [cit. 2023-05-06]. ISBN 1424469848. ISSN 1063-6919. Dostupné z: doi:10.1109/CVPR.2010.5540108

[5] , Editorial Team. Semantic Segmentation: A Complete Guide. *Towards A.I.* [online]. 21.10.2021 [cit. 2023-05-20]. Dostupné z: https://towardsai.net/p/l/machine-learning-7

[6] , Education Ecosystem. Understanding K-means Clustering in Machine Learning. *Towards Data Science* [online]. Medium, 12.9.2018 [cit. 2023-05-20]. Dostupné z: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

[7] GRILLI, E., F. MENNA a F. REMONDINO. A REVIEW OF POINT CLOUDS SEGMENTATION AND CLASSIFICATION ALGORITHMS. In: *International archives of the photogrammetry, remote sensing and spatial information sciences* [online]. Gottingen: Copernicus, 2017, s. 339-344 [cit. 2023-05-20]. ISSN 2194-9034. Dostupné z: doi:10.5194/isprs-archives-XLII-2-W3-339-2017

[8] GUO, Yulan, Hanyun WANG, Qingyong HU, Hao LIU, Li LIU a Mohammed BENNAMOUN. Deep Learning for 3D Point Clouds: A Survey. *IEEE transactions on pattern analysis and machine intelligence* [online]. IEEE, 2021, **43**(12), 4338-4364 [cit. 2023-05-06]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2020.3005434

[9] CHAN, Michael. Step by Step Implementation: 3D Convolutional Neural Network in Keras. *Towards Data Science* [online]. 2020, 28.3.2020 [cit. 2023-05-06]. Dostupné z: https://towardsdatascience.com/step-by-step-implementation-3d-convolutional-neural-network-in-keras-12efbdd7b130

[10] INTEL REALSENSE. Open3D: A Modern Open-Source Library for 3D Data Processing. *YOUTUBE* [online]. USA: Google, 2005, 1.7.2019 [cit. 2023-05-07]. Dostupné z: https://www.youtube.com/watch?v=Rsh4poEpahI&ab_channel=IntelRealSense

[11] KLEEBERGER, Kilian, Christian LANDGRAF a Marco F. HUBER. Large-scale 6D Object Pose Estimation Dataset for Industrial Bin-Picking. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* [online]. IEEE, 2019, s. 2573-2578 [cit. 2023-05-08]. ISBN 1728140048. ISSN 2153-0858. Dostupné z: doi:10.1109/IROS40897.2019.8967594

[12] KUO, Hao-yuan, Hong-ren SU, Shang-hong LAI a Chin-chia WU. 3D object detection and pose estimation from depth image for robotic bin picking. In: *2014 IEEE International Conference on Automation Science and Engineering (CASE)* [online]. IEEE, 2014, s. 1264-1269 [cit. 2023-05-06]. ISBN 978-1-4799-5283-0. ISSN 2161-8070. Dostupné z: doi:10.1109/CoASE.2014.6899489

[13] LI, Deping, Hanyun WANG, Ning LIU, Xiaoming WANG a Jin XU. 3D Object Recognition and Pose Estimation from Point Cloud using Stably Observed Point Pair Feature. *IEEE access* [online]. Piscataway: IEEE, 2020, **8**, 1-1 [cit. 2023-05-06]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.2978255

[14] LIEVENDAG, Nick. UPDATED FOR 2018: REALITYCAPTURE REVIEW. In: *3D ScanExpert* [online]. [cit. 2023-05-20]. Dostupné z: https://3dscanexpert.com/realitycapture-photogrammetry-software-review/

[15] LIGHTSTEAD, Andrew. What is a Pick and Place Robot? Uses and Types. *Robotics 247* [online]. 20.7.2022 [cit. 2023-05-20]. Dostupné z: https://www.robotics247.com/article/what_is_a_pick_and_place_robot_uses_and_types

[16] LUTKEVICH, Ben. What is machine vision?. *Tech Target* [online]. 2018, 7.2022 [cit. 2023-05-06]. Dostupné z: https://www.techtarget.com/searchenterpriseai/definition/machine-vision-computer-vision

[17] MRÁZEK, Filip. *Reconstruction of a 3D scene for bin picking.* Brno, 2022. Diploma Thesis. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Pavel MATULA.

[18] MÚČKA, Jan. *Bin Picking a robotické vidění.* Brno: Vysoké učení technické v Brně. Fakulta strojního inženýrství, 2019.

[19] NAMYALG. What is 2-Dimensional Convolution?: Create your own image filters-. In: *Medium* [online]. Medium [cit. 2023-05-20]. Dostupné z: https://medium.com/theleanprogrammer/2-dimensional-convolution-189abb174d92

[20] NDUATI, Judy. Introduction to Neural Networks. *Section* [online]. 2020, 25.10.2020 [cit. 2023-05-06]. Dostupné z: https://www.section.io/engineering-education/introduction-to-neural-networks/

[21] OWEN-HILL, Alex. Robot Vision vs Computer Vision: What's the Difference?. *Robotiq blog* [online]. 7.7.2016 [cit. 2023-05-06]. Dostupné z: https://blog.robotiq.com/robot-vision-vs-computer-vision-whats-the-difference

[22] PAN, Liang, Zhongang CAI a Ziwei LIU. *Robust Partial-to-Partial Point Cloud Registration in a Full Range* [online]. 2021 [cit. 2023-05-06]. Dostupné z: doi:10.48550/arxiv.2111.15606

[23] SQUIZZATO, Stefano. *Robot bin picking: 3D pose retrieval based on Point Cloud Library.* Padova, 2012. Diploma Thesis. UNIVERSIT- DEGLI STUDI DI PADOVA. Vedoucí práce Menegatti, Emanuele.

[24] STEVENS, Ryland. Industrial Bin Picking vs. Pick and Place. *ENERGID* [online]. Massachusetts, 2001, 10.11.2021 [cit. 2023-05-20]. Dostupné z: https://www.energid.com/blog/industrial-bin-picking-vs-pick-and-place

[25] ŠOOŠ, Marek. *Segmentace 2D Point-cloudu pro proložení křivkami.* Brno: Vysoké učení technické v Brně. Fakulta strojního inženýrství, 2021.

[26] TYCHOLA, Kyriaki A., Ioannis TSIMPERIDIS a George A. PAPAKOSTAS. On 3D Reconstruction Using RGB-D Cameras. *Digital* [online]. Nicosia: MDPI, 2022, **2**(3), 401-421 [cit. 2023-05-20]. ISSN 2673-6470. Dostupné z: doi:10.3390/digital2030022

[27] VAN RIEL, Sjoerd. *Exploring the use of 3D GIS as an analytical tool in archaeological excavation practice* [online]. Sweden, 2016 [cit. 2023-05-20]. Dostupné z: https://www.researchgate.net/publication/303824023_Exploring_the_use_of_3D_GIS_as Diploma Thesis. Lund University. Vedoucí práce Nicolo Dell'Unto.

[28] VARIN, Anand. Introduction to Neural Networks. *Analytics Vidhya* [online]. 2022, 31.1.2022 [cit. 2023-05-06]. Dostupné z: https://www.analyticsvidhya.com/blog/2022/01/introduction-to-neural-networks/

[29] WAANG, Qian a Min-koo KIM. Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018. *Advanced engineering informatics* [online]. OXFORD: Elsevier, 2019, **39**, 306-319 [cit. 2023-05-06]. ISSN 1474-0346. Dostupné z: doi:10.1016/j.aei.2019.02.007

[30] WANG, Qian, Yi TAN a Zhongya MEI. Computational Methods of Acquisition and Processing of 3D Point Cloud Data for Construction Applications. *Archives of computational methods in engineering* [online]. Dordrecht: Springer Netherlands, 2020, **27**(2), 479-499 [cit. 2023-05-06]. ISSN 1134-3060. Dostupné z: doi:10.1007/s11831-019-09320-4

[31] WANG, Yue, Yongbin SUN, Ziwei LIU, Sanjay SARMA, Michael BRONSTEIN a Justin SOLOMON. Dynamic Graph CNN for Learning on Point Clouds. *ACM transactions on graphics* [online]. NEW YORK: ACM, 2019, **38**(5), 1-12 [cit. 2023-05-06]. ISSN 0730-0301. Dostupné z: doi:10.1145/3326362

[32] XU, Hao, Shuaicheng LIU, Guangfu WANG, Guanghui LIU a Bing ZENG. OMNet: Learning Overlapping Mask for Partial-to-Partial Point Cloud Registration. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* [online]. IEEE, 2021, s. 3112-3121 [cit. 2023-05-06]. Dostupné z: doi:10.1109/ICCV48922.2021.00312

[33] XU, Yajun, Shogo ARAI, Diyi LIU, Fangzhou LIN a Kazuhiro KOSUGE. FPCC: Fast point cloud clustering-based instance segmentation for industrial bin-picking. *Neurocomputing (Amsterdam)* [online]. Ithaca: Elsevier B.V, 2022, **494**, 255-268 [cit. 2023-05-06]. ISSN 0925-2312. Dostupné z: doi:10.1016/j.neucom.2022.04.023

[34] XU, Yajun, Shogo ARAI, Fuyuki TOKUDA a Kazuhiro KOSUGE. A Convolutional Neural Network for Point Cloud Instance Segmentation in Cluttered Scene Trained by Synthetic Data Without Color. *IEEE access* [online]. Piscataway: IEEE, 2020, **8**, 70262-70269 [cit. 2023-05-08]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.2978506

[35] YANG, Heng a Luca CARLONE. *A Polynomial-time Solution for Robust Registration with Extreme Outlier Rates* [online]. 2019 [cit. 2023-05-06]. Dostupné z: doi:10.48550/arxiv.1903.08588

[36] YANG, Heng, Jingnan SHI a Luca CARLONE. TEASER: Fast and Certifiable Point Cloud Registration. *IEEE transactions on robotics* [online]. IEEE, 2021, **37**(2), 314-333 [cit. 2023-05-06]. ISSN 1552-3098. Dostupné z: doi:10.1109/TRO.2020.3033695

[37] ZEDNÍK, Pavel. *Detection and Localization of Texture-Less Objects in RGB-D Images*. Prague, 2015. Diploma Thesis. Czech Technical University in Prague. Vedoucí práce Tomáš Hodaň.

[38] ZHOU. Introduction. *Open3D* [online]. 2018, 2018 [cit. 2023-05-06]. Dostupné z: http://www.open3d.org/

[39] Dataset: Fraunhofer IPA Bin-Picking dataset. *Fraunhofer IPA* [online]. Stuttgart, 2019 [cit. 2023-05-08]. Dostupné z: https://www.bin-picking.ai/en/dataset.html

[40] What is computer vison?: Learn what computer vision is, how computer vision works, and what computer vision is used for. *Azure* [online]. Washington: Microsoft [cit. 2023-05-20]. Dostupné z: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-computer-vision/

[41] WHAT IS THE DIFFERENCE BETWEEN MACHINE VISION AND COMPUTER VISION?: WHAT IS THE DIFFERENCE BETWEEN MACHINE VISION AND COMPUTER VISION?. *ZEBRA* [online]. [cit. 2023-05-20]. Dostupné z: https://www.zebra.com/us/en/resource-library/faq/general-technology/what-is-the-difference-between-machine-vision-computer-vision.html

[42] What is Python? Executive Summary: What is Python? Executive Summary. *Python* [online]. Python [cit. 2023-05-20]. Dostupné z: https://www.python.org/doc/essays/blurb/

[43] Unmatched Detection Capability and Ease of Use for Robot Bin Picking. In: *Keyence* [online]. Itasca [cit. 2023-05-20]. Dostupné z: https://www.keyence.com/landing/vision/pr_3d_robot_vision.jsp

[44] 3D Stereo Kamera-System. In: *Imaging Source* [online]. 2017 [cit. 2023-05-20]. Dostupné z: https://www.theimagingsource.com/de-de/resource/newsletter/2017/10/04/

[45] Generating Point Samples on a Mesh: Generating random samples on a mesh. In: *FWilliams* [online]. [cit. 2023-05-20]. Dostupné z: https://www.fwilliams.info/point-cloud-utils/sections/mesh_sampling/

[46] K-Means Clustering Algorithm: What is K-Means Algorithm?. In: *Java T Point* [online]. [cit. 2023-05-20]. Dostupné z: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning

[47] Computer vision and robots, the perfect couple?. *ATRIA* [online]. Zaragoza, 2020, 29.9.2020 [cit. 2023-05-06]. Dostupné z: https://www.atriainnovation.com/en/computer-vision-and-robots-the-perfect-couple/

[48] Collaborative 3D Zivid point cloud captures. In: *Zivid* [online]. [cit. 2023-05-20]. Dostupné z: https://www.zivid.com/hubfs/images/www/Collaborative%203D%20Zivid%20point%20cloud%20

[49] Pick and Place: Pick and Place and Bin Picking with Motoman Robots. *Yaskawa* [online]. [cit. 2023-05-20]. Dostupné z: https://www.cz.yaskawa.eu.com/use-cases/applications/application/pick-place_a10963

[50] ScanXtream 3D Point Cloud and Analysis: 3D Point Cloud Software for Manufacturing. In: *Unblink 3D* [online]. Ontario [cit. 2023-05-20]. Dostupné z: https://unblink3d.com/sxm-point-cloud-processing/

[51] Bin picking in the industry. *ATRIA* [online]. Zaragoza, 2021, 9.2.2021 [cit. 2023-05-06]. Dostupné z: https://www.atriainnovation.com/en/bin-picking-in-the-industry/

[52] Advantages and Disadvantages of Robotic Automation. In: *SP Automation* [online]. March.2019 [cit. 2023-05-20]. Dostupné z: https://sp-automation.co.uk/advantages-disadvantages-of-robotic-automation/

[53] How does Machine Learning Work?: Machine Learning: What is it?. In: *Java T Point* [online]. [cit. 2023-05-20]. Dostupné z: https://www.javatpoint.com/how-does-machine-learning-work

[54] Unsupervised Learning Types, Algorithms and Applications. In: *Nixus Technologies* [online]. [cit. 2023-05-20]. Dostupné z: https://nixustechnologies.com/unsupervised-machine-learning/

[55] Zivid. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 28.3.2023 [cit. 2023-05-06]. Dostupné z: https://en.wikipedia.org/wiki/Zivid

[56] ZIVID. *Zivid One+: Technical Specification, Datasheet*. Norway, 2021, 21 s. Dostupné také z: https://www.zivid.com/hubfs/Zivid%20One%20Plus%20Datasheet%20(1).pdf?hsCtaTracking=f6b
7444-47ac-83a3-44d4840aa429%7Ccabb4557-32f2-4851-9612-99fe7552235b

SYMBOLS AND ABBREVIATIONS

A.I.	Artificial Intelligence
2D	Two-Dimensional
3D	Three-Dimensional
RGB	Red Green Blue
RGB-D	Red Green Blue-Depth
RANSAC	Random Sample Consensus
CAD	Computer-Aided Design
HT	Hough Transform
PCM	Principal Component Analysis
SVM	Support Vector Machine
ICP	Iterative Closest Point
RBF	Radial Basis Function
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
1D	One-Dimensional
cobots	Collaborative Robots
CPU	Central Processing Unit
GPU,	Graphics Processing Unit
XYZ	Coordinates x,y,z
XA	XA Bin picking dataset
IPA	Fraunhofer IPA Bin-Picking dataset
6D	Six-Dimensional
FPCC	Fast Point Cloud Classification

OOP	Object-Oriented Programming
DGCNN	Dynamic Graph Convolutional Neural Network
MLP	Multilayer Perceptron
PPF	Point Pair Features
4D	Four-Dimensional
NN	Neural Network
etc.	Et cetera
.stl	standard tessellation language format
GMCNet	Graph Matching Consensus Network
OMNet	OMNet: Learning Overlapping Mask
d	distance to the target
m	meter
cm	centimeter
c	speed of light
s	seconds
ms	miliseconds
t	time of flight
rad	radian
$\Delta\phi$	phase difference
f	frequency
Hz	Hertz
i	index i
.txt	plain text file
(nx, ny, nz)	normalizated location coordinates
F	Feature vector
$[x_i, y_i, z_i]$	Coordinates on index i

β	beta - distribution of center score
p_i	point on index i
c_i	corresponding center on index i
$d_{F(i,j)}$	feature distance matrix
ASM	attention score matrix
VDM	valid distance matrix
d_{max}	maximum distance from center
D_F	feature distance matrix
D_V	valid distance matrix
$S_{center(i)}$	center score vector
$\hat{S_{center(i)}}$	predicted center score
$\kappa_{(i,j)}$	loss based on point pair
ϵ_1, ϵ_2	constants
L_{CS}	Center score loss
$w_{(i,j)}$	weight matrix
$S_{A(i,j)}$	attention score matrix
$p_{1,2}$	point 1, 2
m_1, m_2	model points 1, 2
s_1, s_2	scene points 1, 2
d_{dist}	sampled distance
d_{angle}	sampled angle
n_{angle}	the number of times the circle is sampled
$\mathbf{M}$	Set of model points
$\mathbf{S}$	Set of scene points
F_m	model features
F_s	scene features

s_r	reference point
$\mathbb{R}_x(\alpha)$	Rotation by angle α
$T_{s\Rightarrow g}$	Translation reference to origin

LIST OF FIGURES

LIST OF TABLES