Faculty of Applied Computer Science

Chair of Production Informatics
Prof. Dr.-Ing. Johannes Schilp

Universität
Augsburg
University

BACHELOR THESIS

AI-BASED COMPONENT RECOGNITION IN CAD ASSEMBLIES

Author:	Anonym
Student ID:	
Major:	
Examiner:	
Supervisor:	
Release date:	
Submission date:	30.09.2024

Abstract

CAD assembly models define the geometry of complex physical objects that are manufactured by fitting together parts and sub-components. These can be held together by different technical solutions including welding, bonding, fastening, interlocking and more. Especially screwing presents the challenge, that knowledge is required about the position and rotation of screws, necessary torque and what driver tool is needed. To this date such information is gained by human insight - AI component detection could change that.

Detecting fasteners in a CAD assembly furthermore is a milestone toward automatic assembly order generation and even automated creation of step-by-step assembly instructions. That facilitates setting up a production line for the manufacturing of a new product exclusively on the basis of the CAD file.

This work concludes it is feasible and practicable to identify distinctly shaped standard components like fasteners in cad assemblies.

Keywords: AI in production, CAD, component recognition, fastener recognition, screw recognition, STEP, part isolation, shape matching

Table of contents

Table of figures

List of abbreviations

Abbreviation	**Meaning**
AI	Artificial Intelligence
AOA	Assembly Order Automation
BREP	Boundary Representation
CAD	Computer Aided Design
CRCA	Component Recognition in CAD Assemblies
ML	Machine Learning
MVCNN	Multi-View Neural Network
NAUO	Next Assembly Usage Occurrence
OCC	OpenCascade
STEP	Standard for the Exchange of Product Data

1 Introduction

1.1 Problem Statement and Relevance

Component recognition in CAD assemblies (hereafter referred to as CRCA) is relevant for several key areas in industrial production:

Beginning on the micro level of individual fasteners, CRCA is an important prerequisite to automate fastener assembly by robotic manufacturing tools. To install a single screw, assembly robots require a set of information about that screw [1]. This might include position, angles, drive type, length, torque, possibly even magnetic properties and weight. Only after recognizing which components in the assembly are screws, necessary data can be derived from the assembly file [2]. Component recognition as targeted in this work therefore aids assembly robots in **installing fasteners automatically.**

This is only possible if fasteners are accessible. During assembly, accessibility of fasteners might be obstructed. Therefore, assembly must happen in some order. Installing a component too early can block access to underlying components, preventing their successive installation. Even though products are designed with an assembly order in mind, such information is not stored into the CAD file. This is because assembly order is not universal but depends on characteristics of tools used [3]. Therefore, even if some assembly order was provided with the CAD file, it would likely not be the only possible and maybe not the ideal one for the machinery on-site. Assembly order generation is a complex necessity in production industry and determining access to fastener locations is a small but essential requirement of it. That indicates: CRCA is an industry requirement even at the macro level for **assembly order automation**.

And it does not stop there. CRCA can even provide benefits on the management level **optimizing resource utilization** [4]: Shortages of standard components in an assembly line may occur at critical moments. Unless promptly resolved, downstream productivity is at risk, potentially idling production. Detecting identical components in a CAD assembly can help to dynamically reallocate available identical material reserved for other assembly steps. This can optimize productivity and ease material management.

These areas of the production process showcase the relevance of component recognition in CAD assembly files for the industry.

Putting the current market situation in perspective: While the STEP file format is considered de facto industry standard for the interchange of CAD files [5], it does not define consistent identification of standard components. Companies use custom labeling of parts in non-uniform ways or classify by hand. The absence of standardized fastener identification in STEP files presents a technological gap. To meet the needs of production industry for autonomous fastening, assembly order generation and resource management, there is an urgent need for component classification in CAD assemblies.

1.2 Objectives and Methodology

This work presents a programming-centric exploration into providing AI-classification of standard components – specifically fasteners – in CAD assemblies. Compared to other fastener types like bolts, washers, nuts and rivets, screws feature substantially more geometric variety [see chapter 3]. Consequently, a classifier able to recognize screws should be adaptable to recognize each of those fastener types as well. Therefore, the aim of this work is focused on identifying screws in CAD assemblies with a high precision.

Data in form of CAD assembly files will be sourced from public websites. Three types of assemblies are relevant:

- collections of only screws to generate samples for the positive class
- assemblies of exclusively non-screw components for the negative class
- assemblies containing both screw and non-screw components as validation data.

Research on relevant screws in the application context is conducted along this process to collect suitable sample data [chapter 3]. Components will be isolated from the STEP files in automated fashion generating data samples. Ground truth generation will be performed by human classification of the samples while the dataset is observed to be sufficiently large and balanced to perform machine learning. Features are extracted by converting each 3D component sample into views from multiple perspectives.

Literature suggests Multi-View Convolutional Neural Networks (MVCNNs) perform very well for 3D object recognition and are considered state-of-the-art [6, 7]. The machine learning model will therefore use the MVCNN architecture. Training will be optimized with respect to precision and recall. Classifier performance will be monitored per accuracy metrics available in the confusion matrix during test and validation. On this basis hyperparameters like learning rate, classification threshold and number of epochs will be tuned.

1.3 Hypothesis

Image recognition has advanced significantly in recent years. The automotive industry already relies on 3d object recognition from depth cameras data, even for real-time applications such as autonomous driving [8]. This points out the performance of the currently available combination of computer vision and AI object recognition. In the CAD domain, 3D data is even more complete and noise-free, providing an ideal ground for this computer vision task. Thus, the hypothesis is:

"By utilizing an enhanced multi-view approach machine learning will be able to determine the affiliate class of components in a good majority of unequivocal cases."

1.4 Scope and Limitations

Intermediate goal of this work is the successful disassembly of all components of a CAD-assembly into isolated parts along the hierarchy even to the level of subparts if the source file provides such detail.

Main objective is the design and training of a neuronal network to perform binary classification on each of these isolated particles of the assembly. The classification must identify between screw and no screw.

Focus is geometric recognition as opposed to relying on parts labels, material, color or hierarchy.

Technology usage is limited to free open-source software only. Code is to be written in python.

2 State-of-the-Art in Research & Technology

2.1 3D shape classification:

In related research, the authors analyze the performance of shape classification when shapes are represented as voxels, point cloud, or multi-view images. VoxNet [9], PointNet [10] and MVCNN [6] are compared respectively. Findings present that MVCNN solidly outperforms the two compared solutions. The work goes further enhancing the views by using depth images, shaded images and binary silhouette images comparing classification performance on the MVCNN. It is concluded that shaded images perform better than depth images, while the combination of shaded and depth views is optimal [7].

2.2 Viewpoints in MVCNN:

In the field of neural networks designed to learn 3D object detection, researchers analyzed the influence of geometrically ordered versus random viewpoints in a MVCNN architecture [11]. They conclude that the classifier performs best when viewpoints are arranged in a uniformly spaced circle all facing the object in the center. Views offering different vertical angles arranged evenly on the surface of a sphere around the object proved less performant. Random arrangement of viewpoints leads to the worst performance. However, the research does indicate, that the advantage of the relatively constant performance of the proposed 'latent' circular arrangement can be outperformed by the other arrangements for view counts from 20 upwards.

2.3 X-Ray Multiview:

In airport security applications, perpendicular X-Ray images are captured from traveler's baggage to recognize a variety of threats [12]. Important conclusion from this research is, that even while X-Ray pictures offer insight into the depth of the material, occlusion can not be entirely prevented. However, implementation of a multi-view approach can lessen the impact of occlusion and increase object recognition performance. Another research in the same area of transportation safety chooses a more specialized objective classifying only handguns in X-ray images [13]. The mentioned difficulty is, that a gun is not expected to be aligned conveniently along the camera´s axes and would likely even be intentionally obscured among other metallic clutter inside the luggage. To cope with this and achieve high quality results required in such critical application, this work presents an impressive amount of data augmentation.

2.4 Stereo-based depth & Stereo Depth Network

Research in the autonomous driving field has produced another interesting approach training object detection on a basis of a LIDAR-like data derived from stereo views. [14]

2.5 Visual screw head classification in industrial robotic context:

Researchers at the KIT have performed a practical exploration into screw detection, screw type recognition and automated disassembly with an industry robot. They employed the popular object detection CNN YOLOv5 on camera footage and classified a limited set of M3 and M6 screws of Hex, Allen, Slotted, Philips, Pozi and Torx into six head classes and the two size classes [15]. The resulting validation precision was about 87%.

2.6 Human vision of 3D vs machine:

Based on the assumption that humans have acquired the skill to sense depth in 3D scenes presented on a 2D plane, research in the area of psychophysics was performed that aims to analyze what visual stimuli aid human depth recognition. The work starts giving insight into understanding human performance in distinction of depth in an image on the basis of occlusion. It than introduces a neural network modeling the same behavior. The finding is that machine vision is able to confidently derive relative depth information from occluded objects in 2D images [16].

High risk environments like airport security and autonomous driving are applications where confidence in the classifier’s prediction decides about life or death. Applying relevant principles of optimizing object detection from these areas is expected to be beneficial for improving the classification performance of this work.

3 Different Screw Types

Before identification of screws is possible, it is necessary to get an overview about the distinct shapes of screws to be recognized. Samples from each type must be collected for training the classifier.

3.1 Head Shape

An overview of common head shapes, excluding those designed for manual installation:

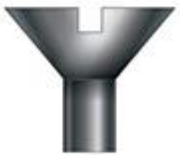
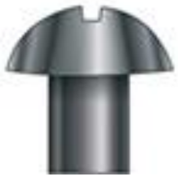
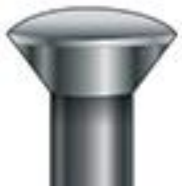
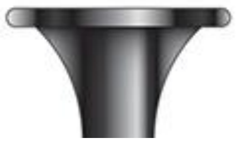
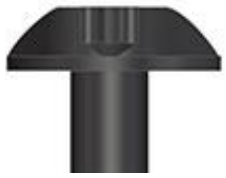
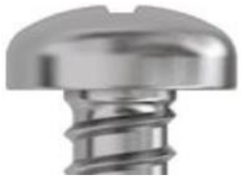
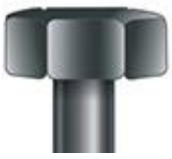
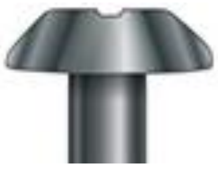
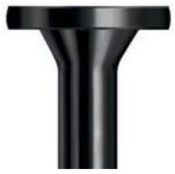
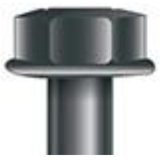
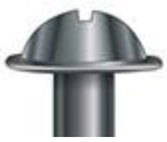
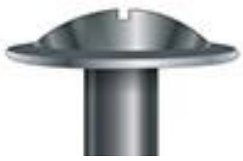
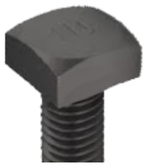
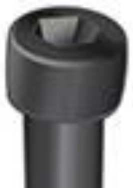
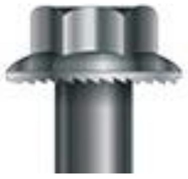
 **Countersunk / Flat**	 **Round**	 **Raised**
 **Bugle**	 **Button**	 **Pan**
 **Hex**	 **Truss / Mushroom**	 **Pancake**
 **Hex Flange**	 **Round Flange**	 **K-Lath**
 **Square**	 **Socket / Cylinder**	 **Serrated**

Table 1: Head shapes, after [17–20]

3.2 Drive

Apart from the two outer drive shapes: hex and square, screws can also be driven by different slots. Many more drive shapes exist, but here are the most popular:

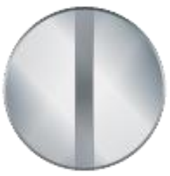
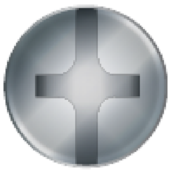
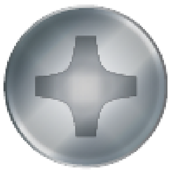
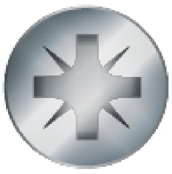
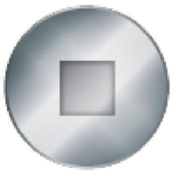
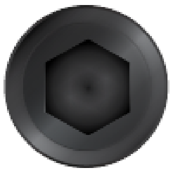
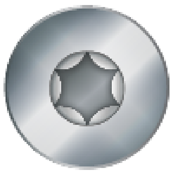
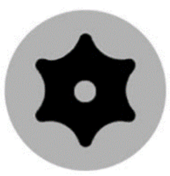
 Slotted	 Combi	 Philips	 Pozi
 Robertson	 Allen	 Torx	 Torx-TR

Table 2: Drive shapes [21, 22]

3.3 Thread shapes and types

Thread shapes vary application-specific aiming at reduced friction, economical production water- and airtight sealing and optimal load distribution [Figure 1: Thread Shapes]. Details are again out of the scope of this work, but the variety must be considered during data collection.

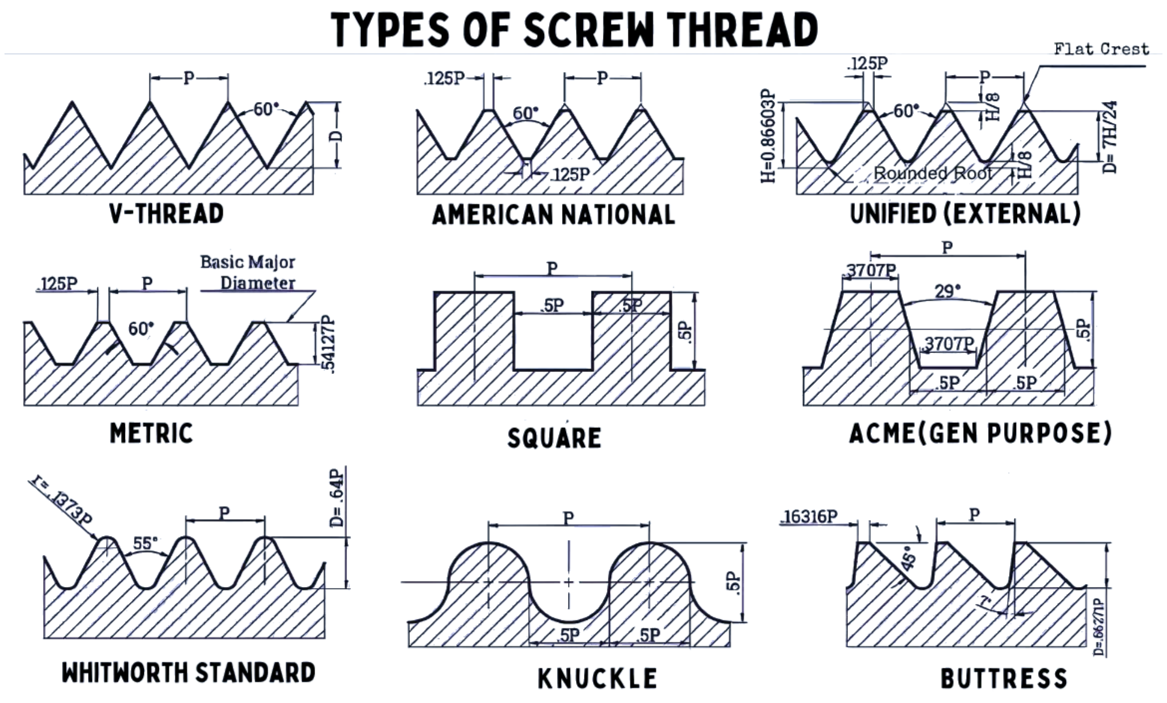


Figure 1: Thread Shapes [23]

Wood screws designed for application in soft mating materials are recognized by their wide threading. Variants with self-tapping capabilities are distinguished by having drill-like geometries at the tip [Figure 2].

Rapid construction screws aimed at gaining more depth per rotation feature even wider thread slope or several parallel threads instead of one [Figure 3], after [28].

Wave shaped thread shapes prevent wood from splitting along grain [Figure 4].

Metal screws are mostly installed in machine-threaded holes or nuts. In such applications high thread slopes could result in slippage. As a result, metal screws are designed with narrower pitch [Figure 5]. Exception are U-type screws that feature very steep thread angles since they are hammered into the surface [see below].

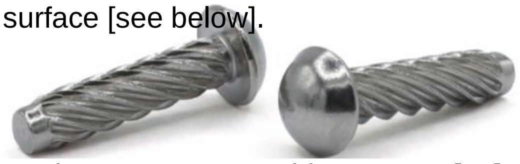


Figure 8: Hammer drive screws [29]

The screw shoulder is the area between head and the main threading of the screw and can have a variety of forms and functions [Figure 6].

Further thread altering is employed to generate prevailing torque increasing friction at the threads and thus preventing the loosening of screws [Figure 7].

3.4 Material and Dimensions

Material characteristics are defined in the STEP format, but this work aims to recognize based on geometry only. Of course, screws also vary in dimensions like length and thickness, multiplying the presented distinct shapes even further.

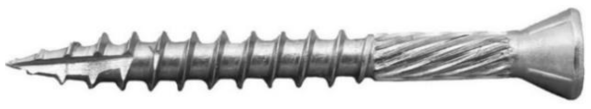


Figure 2: Self-tapping tip, knurled shoulder, self-countersinking ribs, after [24]

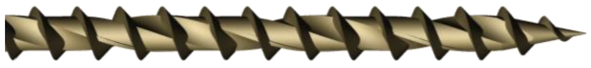


Figure 3: Rapid construction screw, after [25]

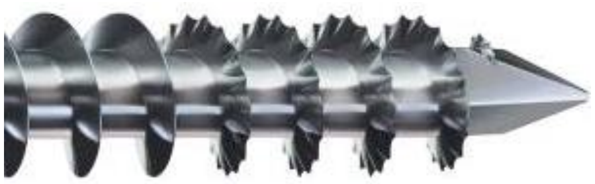


Figure 4: Wood screw with wave shape threads, after [26]

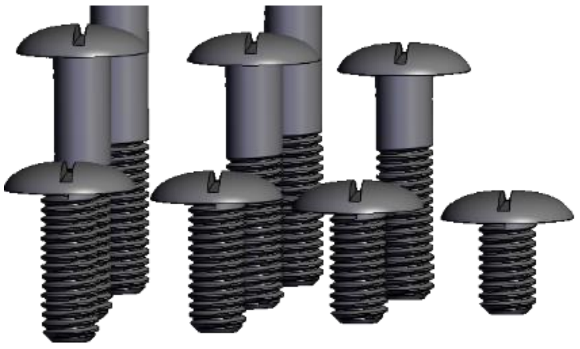


Figure 5: Metal screws with different shank lengths, after [23]

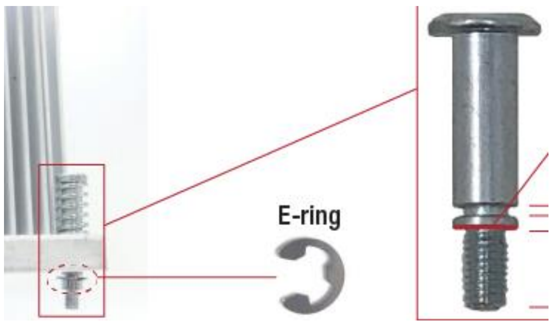


Figure 6: Wide shoulder with groove [20]

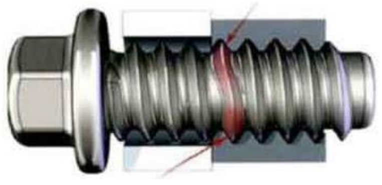


Figure 7: Curved threads are one of various techniques to generate prevailing torque [27]

This overview gives insight on diversities to consider when collecting samples for training data and how to classify them.

4 Implementation

4.1 System Overview

Available hardware is a commercial notebook with 8 AMD 4700u CPU cores, 16 GB RAM, 512 GB SSD and no GPUs. The following open-source software is utilized: PyCharm Professional for educational use, FreeCAD python console.

A brief overview over the dataflow before details are explained:

Input CAD assemblies are disassembled into components, these components are further disassembled into shapes, the 3D shapes are transformed into a voxalized 3D representation and from there into multiple 2D views [see below]. On those features the machine learning classifier is operating.

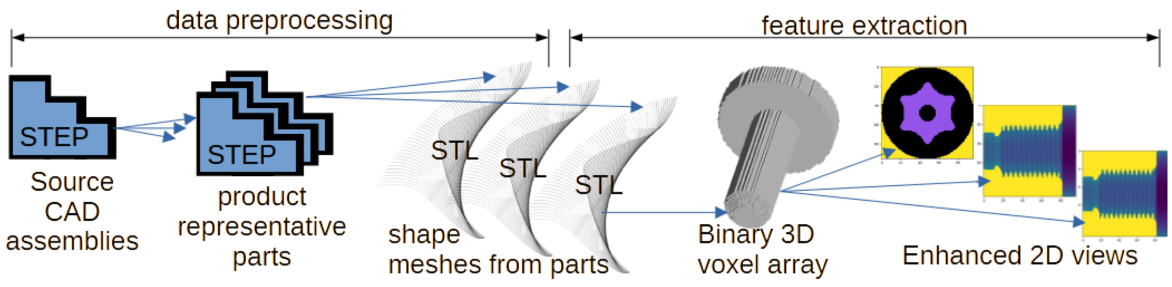


Figure 9: data preparation until the training of the machine learning model

4.2 Data Preprocessing

At the start of the workflow CAD assemblies in STEP file format are expected as input. Those assemblies consist of a hierarchy of sub-components and shapes that are to be classified. Consequently, part extraction defines the first necessary milestone of the dataflow. Various implementations exist to extract parts from STEP files. Most prominent among those is the OpenCascade Technology, which also works at the basis of the FreeCAD modeling tool. This tool provides an extensive manipulation feature set via its python API which is very helpful for this work to transform and compare shapes and geometries.

In a STEP file 'parts' consist of other parts, 'shapes' and shells. By this recursion a hierarchy is formed. Relevant objects in this hierarchy are the shapes, containing specific instances of object geometries including position and rotation. Meanwhile, parts are just logical containers serving to distinguish and reuse whole components. Each part is linked to a 'product'. A product defines a geometry template from which one or several part instances can inherit. For example, in an assembly there might be a number of identical nuts [Figure 10], which should inherit the geometry from one product. In that optimal case it is sufficient to extract only one of the identical parts for classification. The first extraction stage therefore is to extract a list of products and then extract one representative part per product. This part will be called the 'product representative'.

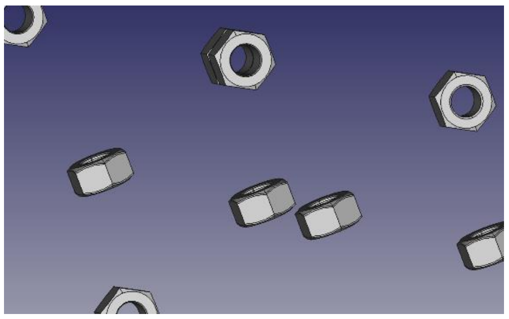


Figure 10: Isolating one product results in several identical parts

4.2.1 Extract Product Representatives

In theory, this should result in the straightforward approach depicted in Figure 9. However, a common problem discussed in literature is, that reuse of shapes and parts in CAD files currently poses serious consistency flaws. A part can be properly traced back to its origin only if duplicated by ‘linking’. If duplicated by just copying, it can not be identifies as being the same product since a new product instance is generated and linked while the original reference is lost. This is the expected behavior, but unfortunately a common user error when creating CAD assemblies [30]. Assume an assembly contains several identical shapes, like the turbine in Figure 11. Each identical shape should ideally have been linked to a single product definition, implementing the geometry from the product and just transforming position coordinates and rotation vector for each shape individually. While the optimal design process involves linking all identical parts to the relevant product, the sampled CAD assemblies demonstrated that this inconsistency appears to be widespread. As a result, assemblies must be expected to contain duplicate identical products and solids. This issue is partly addressed in the proposed solution: Specifically on the shape level of a part.

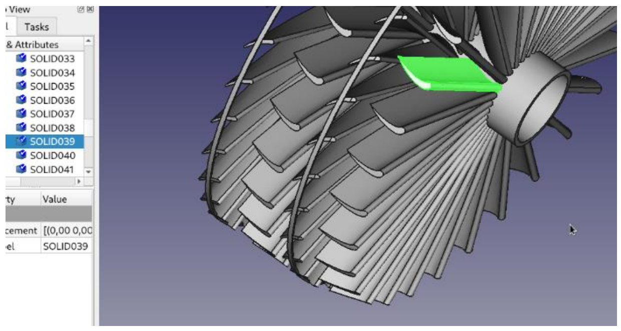
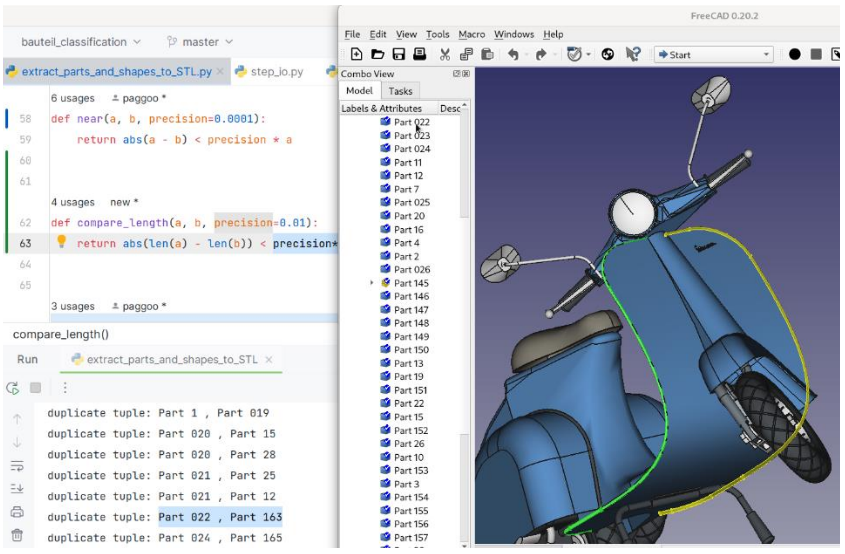


Figure 11: Turbine with identical solids

The proposed solution offers a method to compare and exclude duplicate shapes by pairwise comparison. For this, shapes are normalized by transformation into the zero position and zero rotation, then the union overlay is calculated among other metrics. Pairwise comparison generally has computational complexity of $\binom{n}{2} = \frac{n(n-1)}{2}$ which is in $\Theta(n^2)$. Shapes within a part might be in the order of $n \approx c \cdot 10^2$. Comparing all such shapes within a part generates pairs in the order of 10^3 to 10^5, which still performs reasonably well on the available hardware. However, comparing all shapes in the entire CAD assembly was attempted but had to be excluded due to the resulting runtime. This is understandable, given that an assembly might have parts in the order of $n \approx c \cdot 10^3$. This results in $n \approx c \cdot 10^2 \times c \cdot 10^3 \approx c^2 \cdot 10^5$ shapes in the assembly which makes comparisons in the order of 10^9.

Apart from being computationally infeasible, comparison of parts can entirely be prevented by designing the CAD assembly with care to reusing parts. This situation explains very well why there is no available solution for preventing duplicates when isolating parts and shapes yet.



```python
def near(a, b, precision=0.0001):
    return abs(a - b) < precision * a


def compare_length(a, b, precision=0.01):
    return abs(len(a) - len(b)) < precision*
```

```
duplicate tuple: Part 1 , Part 019
duplicate tuple: Part 020 , Part 15
duplicate tuple: Part 020 , Part 28
duplicate tuple: Part 021 , Part 25
duplicate tuple: Part 021 , Part 12
duplicate tuple: Part 022 , Part 163
duplicate tuple: Part 024 , Part 165
```

Figure 12: Not identical (mirrored) shapes

For this reason, the solution implemented for this thesis is a combination of finding identical parts based on the mentioned links and identifying identical shapes within parts by pairwise geometric comparison.

Identical parts:	based on linked product
Identical shapes:	based on shape characteristics gained using FreeCAD console

- volume of intersection between shapes transformed into zero location and rotation
- similar values in memory size, area and mass
- similar numbers of edges, faces, wires, vertices

This approach enables isolating the single product representative in the primary stage and isolating unique solids from within those product representatives in a further stage. To find the link from part to product the STEP standard was analyzed. Related literature revealed: parts are described by a 'next assembly usage occurrence' (NAUO) entry [31]. The NAUO entry contains the relevant link from a part to a product. Here an example NAUO entry:

```
#144691 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('NAUO682',' ',' ',#97553,#19786,$);
```

This line provides some relevant info about a part. Primarily the fifth parameter will be called 'product entry' since it references the linked product: `#19786` in the above example. For identifying parts belonging to the same product such as the nuts in Figure 10 this is the essential information.

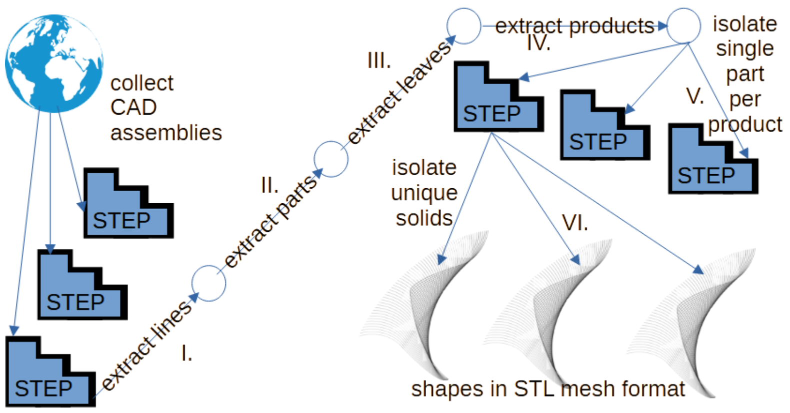


Figure 13: Preprocessing in detail

With this knowledge at hand, isolation of the product representative can be archived in five stages as presented in Figure 13 (created after [32], [33])

Stage I is reading the lines from the STEP file and transforming them such that entries spanning over several lines are concatenated into one line for uniformity of methods in following steps.

In **stage II** NAUO entries of all parts are gained by collecting such lines from the result of stage I that contain the NAUO keyword.

For **stage III** it is important to understand that extraction should only happen at the leaves of the hierarchy. In the context of tree structures the terms parent, child and leaf are commonly used. If a part is not a leaf, it is an assembly containing further children. Since the aim is to extract and classify the finest particles, there is no gain in determining that any subassembly is not a screw. The screws are to be found in the leaves of the hierarchy. Going through all NAUOs extracted in stage II, the fourth NAUO parameter is observed. This entry (`#97553` in the example above) will be called 'parent entry' since it references the parent product. Selecting those NAUOs, whose product entry is not mentioned in any other NAUOs parent entry produces the leaves´ NAUOS.

Stage IV: A list of all product lines is gained by finding all references in the NAUOs products entry.

```
#19786 = PRODUCT_DEFINITION ( 'UNKNOWN', '', #63691, #126029 ) ;
```

Above is the respective product line. The number at the start of the line `#19786`, indicates that it is referenced by the NAUO entry from the example. Deleting such product line - given it is a leaf product - results in removing all parts referencing this product from the assembly.

Finally in **stage V** to isolate a single product representative, first, a ‘bloodline’ is generated. Bloodline describes the lineage from this parts product entry to the parent NAUO´s product entry recursively until the root which has an empty parent entry. All product lines not referenced in the bloodline are removed from the lines extracted in stage I. At this moment the picture for Figure 10 was taken. Only parts from the chosen product remain. Now the product representative can be any of the parts since all are on leaf levels. Therefore, every NAUO must be deleted from the lines except a single one. The result is one of the STEP files with a product representative.

4.2.2 Mesh Conversion

This work focuses on an MVCNN approach with enhanced views. Technological solutions on how to gain voxelized or point cloud data from 3D objects are discussed in literature. One solution encountered using the pyntcloud library presented a very appealing entry seeming simplistic and straightforward [34]. Upon further research the approach was adapted so only the idea for the mesh export via FreeCAD remained [35]. Instead of the .asc mesh format this solution was altered to use the common .stl mesh format since another available voxelization solutions working on the stl format was used in the follow-up view creation. The intermediate conversion into stl mesh format therefore eases the calculation of the views by reusing available technology.

With reference to Figure 13 this describes stage VI of the data preprocessing. Instead of manual export, the FreeCAD console was utilized, which proved to be especially helpful in this part of the project for its extensive set of transformation features streamlining the workflow. Before the export of solids, they are compared pairwise as mentioned in the previous sub-chapter. This allows to generate a material list of unique parts. In case the fastener recognition still needs a human verification, removal of duplicates should reduce the burden on the employee as much as possible.

Precision of the Mesh can be adapted by a parameter in the API command which was experimented with to find an ideal trade-off between resolution and runtime.

4.2.3 View Creation

3D object recognition by enhanced views in a MVCNN architecture is discussed in literature and considered state-of-the-art technology delivering competitive performance[6]. 3D component detection can specifically benefit from an X-Ray like view through a solid body to use the depth information for better shape distinction [12]. Looking at the picture to the right [Figure 14] deficiencies in perception of depth become apparent. Research has found that humans are able to detect depth by occlusion [16]. In this context the circle forming the head disk is occluded by the screw shank,

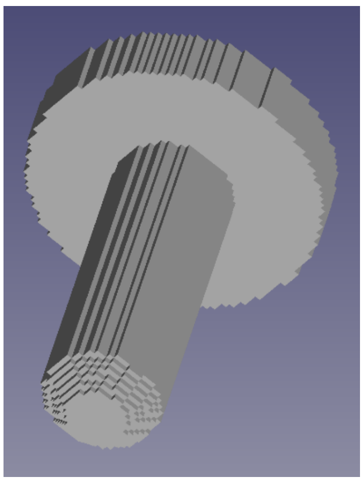
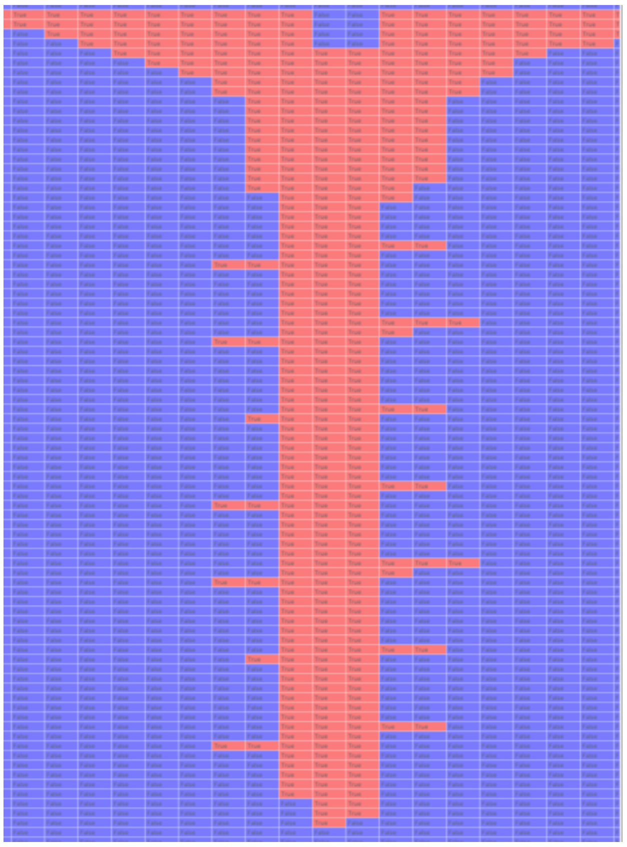


Figure 14: Voxelized screw

indicating that this part of the head is further away from the view than the shank. Apart from this the image only depicts greyscales that do not carry depth information. It is therefore based on experience to understand, that the head disk is neither oval shaped nor parallel to the screen while the pixels not portrait depth. Without depth information the machine learning model would need much more data to gain this experience. For this reason, advanced views containing depth information are used in this work. These enhanced views are derived as follows:



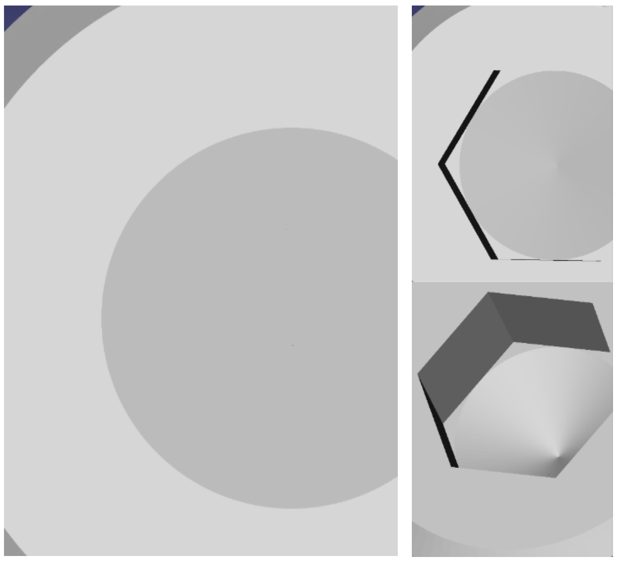


Figure 15: Head from above: socket invisible

The mesh of the component is transformed into a voxel-like 3D binary array that contains 1 wherever material is located and 0 where there is no material.

Figure 16: visible at angle

For this, an available solution was adapted [36]. Removing irrelevant factors like material and color allows the required abstraction so the focus remains clearly on geometry alone. Figure 17 demonstrates gained insight in a 2D slice of the resulting 3D array.

The binary 3D array really offers room for extracting further information. For this work depth data was of interest since there is need and related applications already employ

Figure 17: Slice of binary 3D array

3D object detection with depth data. Two types of depth data were generated: X-Ray like additive depth data and depth-camera like distance data. The binary 3D array is summed over the 3 axes x, y and z resulting in 2D views. This resembles the radiation passing through objects in an X-ray image. The sum on a pixel is higher where layers contain more material. More precisely, the resulting image looks like the negative of an X-ray image since denser material prevents more radiation from reaching the receptive layer and such pixel would have a lower value.

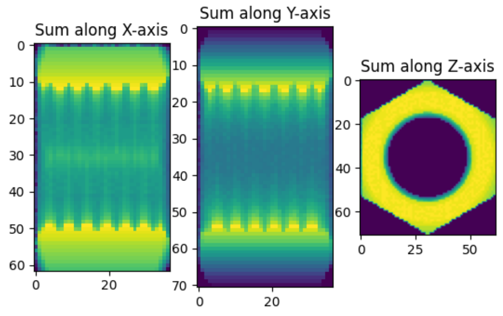


Figure 18: X-ray like depth view

Enhancing multi-views of a sample likely leads to better performance than compared to the regular multi-views. The problem with non-enhanced views of 3D models is that distance from the camera is not seen on a 2D screen. As a result, the deeper slot inside a screw head is invisible from straight above, similarly to how embossed or engraved font is only visible at an angle [Figure 15Figure 17]. Neither is the slot visible when seen from the tip side, nor when viewing the screw sideways in case of an unfortunate rotation (slot parallel with the view plane). When looking at socket head screws sideways the socket shape is even always invisible. In all imaginable scenarios the slot becomes visible on the X-ray image [Figure 19].

The X-ray view enables to recognize what driver a screw has even from the tip side since the sum makes the view on head side and tip side identical. On further inspection the views were even more enhanced by calculating depth-camera like views [Figure 20].

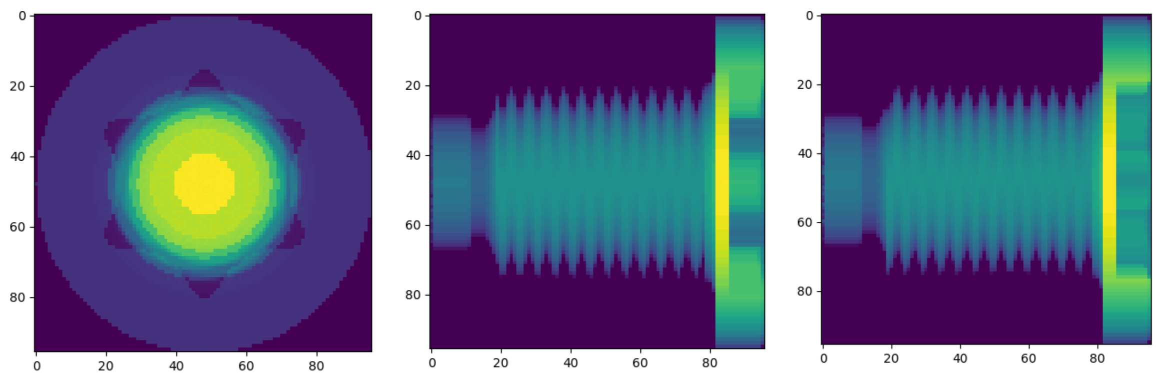


Figure 19: X-ray like views

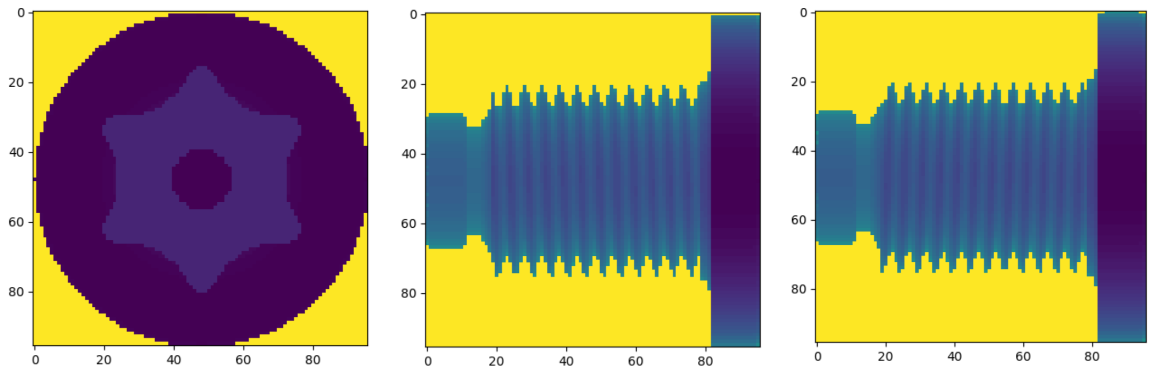


Figure 20: Depth-camera like views

Problem with the X-ray view is that due to summing the shank along the length of the screw it was not clear whether the drive is of type torx or torx-tr. Depth-camera style views are generated as follows: The head of the screw is found by observing the binary 3D array like a cube and calculating the weight between the front facing half of the cube and the weight of the half facing back. This is calculated along all 3 axis and the heaviest one is supposed to be the head. Of course, that is under the assumption that the head half contains more material than the tip half. Resultingly, the depth is aggregated by counting the 0-voxels from the cube face in a linear fashion until reaching the first 1.

As discussed in the research section, the disadvantage of a small number of views chosen due to computational restrictions might be canceled out by prevention of random view angles.

4.3 Training & Validation

Two models were developed: Tensorflow was performing ideally.

4.4 Challenges & Solutions

4.4.1 Part isolation

Main technical challenge was the isolation of parts. In the beginning the attempt to remove parts was experimented on the file level by removing NAUO lines. This quickly succeeded and the experimental fashion of the approach brought profound insight into the structure of the STEP format starting from how parts are defined, the difference between leaf parts and parent components, extraction of relevant labels along the hierarchy all the way to inheriting geometries from products. Following through on this approach, however, part isolation became an inefficient hassle. This was due to the fact that the approach was based on files that were of great diversity. STEP files collected from various sources presented seemingly conflicting realities about the format and therefore the conclusion for the future must be that reliable information on the standard is to be harnessed primarily from official documentation.

The final solution is now utilizing tools for part extraction. Available implementations using OCC as well as FreeCAD scripting were evaluated and adapted. The solution to use those libraries effectively turned two months of effort into dust replacing it with a handful commands on the FreeCAD python console. After getting so excited and attached that felt sad. It has to be considered, however, how much understanding working on the STEP file brought. To be fair, these solutions were not even encountered in the beginning. Apparently, the necessary keywords to search for were not fully understood and embraced when the project started. Knowledge was built up along the process making it possible to judge existing approaches and adapt them according to the need.

In essence, the solution offering more reliable performance must be selected. In this instance, FreeCAD proved more reliably even employing automatic error-fixing to input files upon import.

4.4.2 Duplicate exclusion

Originating from the idea to simplify extraction of all parts by just extracting unique parts, the duplicate exclusion ironically ended up being the opposite: computationally expensive and multi-processing even induced more comparisons than necessary.

Surprisingly the solution was derived of knowledge from the lecture statistics. That aided in building a set of pairs and passing it to the multiprocessing pool in such a way enabling this calculation efficiently. Aside from this, shape duplicates were not calculated across different parts as discussed in chapter 4.2.1.

This results in duplicate parts wherever case parts in STEP file are not correctly linked.

4.4.3 Lacking prior expertise in CAD

Since my knowledge in CAD was non-existent, not only relating the STEP format but already the basic concepts like differences between assembly, product, part and solid and their constructed, use and representation was entirely new and unclear in the beginning. As a result reviewing API´s and libraries like OCC or Open3d [37] confused me and made me feel lost. FreeCAD overwhelmed me by the functionality, so I put the tools aside. I started analyzing a STEP file with the text editor attempting to follow the keywords and references. At the same time, I researched the theory and quickly succeeded in updating colors of faces within the step file, deleting parts and got more confident. The uncertainty from the start flipped into excitement. This should have been the right time to give the libraries for part extraction another chance. But I overestimated that I had already extracted parts and underestimated all the tedious edge case treatments, that were most probably handed just fine in already existing solutions. Nevertheless, looking back, I am very happy having taken this challenge to approach an entirely new field of CAD. This experience taught me new techniques, especially that I must not be too fearful in the future to use someone´s libraries. This is a big lesson for me since the potentially saved time could have allowed me to train more classifiers or collected a bigger dataset at least. The challenge to a very new and exciting field was in fact the reason I decided for this topic versus another one I had been already very familiar with.

4.4.4 Limited Computational Resources

While shape comparison, dataset generation and model training the code was found crashing on the basis of limits to hardware, especially the RAM was observed to be lacking. Solution to this was the separation into chunks, in a way that optimized the resource utilization.

5 Results

Validation of the classifier was performed on a variety of CAD assembles from the industrial background. A clear example is the sheet metal rack shown in **Fehler! Verweisquelle konnte nicht gefunden werden.** and **Fehler! Verweisquelle konnte n icht gefunden werden.** and the proposed classification result is displayed in the confusion matrix in Figure 21. In further unambiguous models the classifier generally performs with both precision and recall well over 95%. However, this data has to be interpreted carefully in context of part removal.

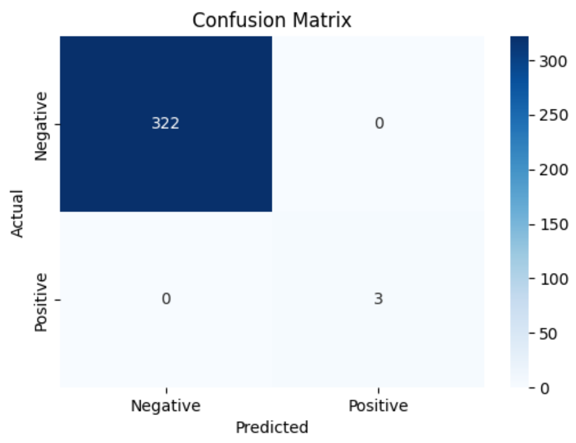


Figure 21: Classification Result on Validation Data

Namely the 322 negative samples are skewed in comparison to the three screws. In fact, the repetitiveness of parts in this model is high so that one should expect unique non-screw shapes only in the order of 20. The big gap is explainable and has to do with the fact how identical shapes are not recursively detected. As explained in section 184.2.1 parts are compared by links to the identical products and inside each part identical shapes are also found by pairwise comparison. As discussed, since the parts are not linked to the same product, they are not considered identical and shapes within different parts are not compared with one another due to computational complexity. The implemented solution specifically only computes similarity among shapes inside a single part. In this situation specifically the ball bearings are seemingly not linked to the same product and consist of subordinate shapes as visible in Figure 22. Meanwhile three different screws are identified correctly.

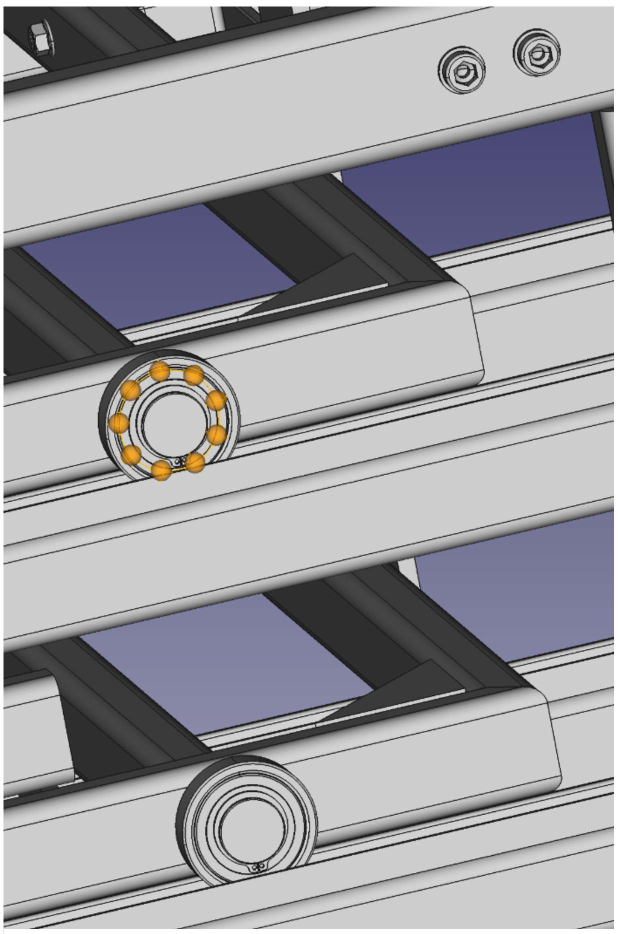


Figure 22: Multiple identical ball bearings containing sub-shapes

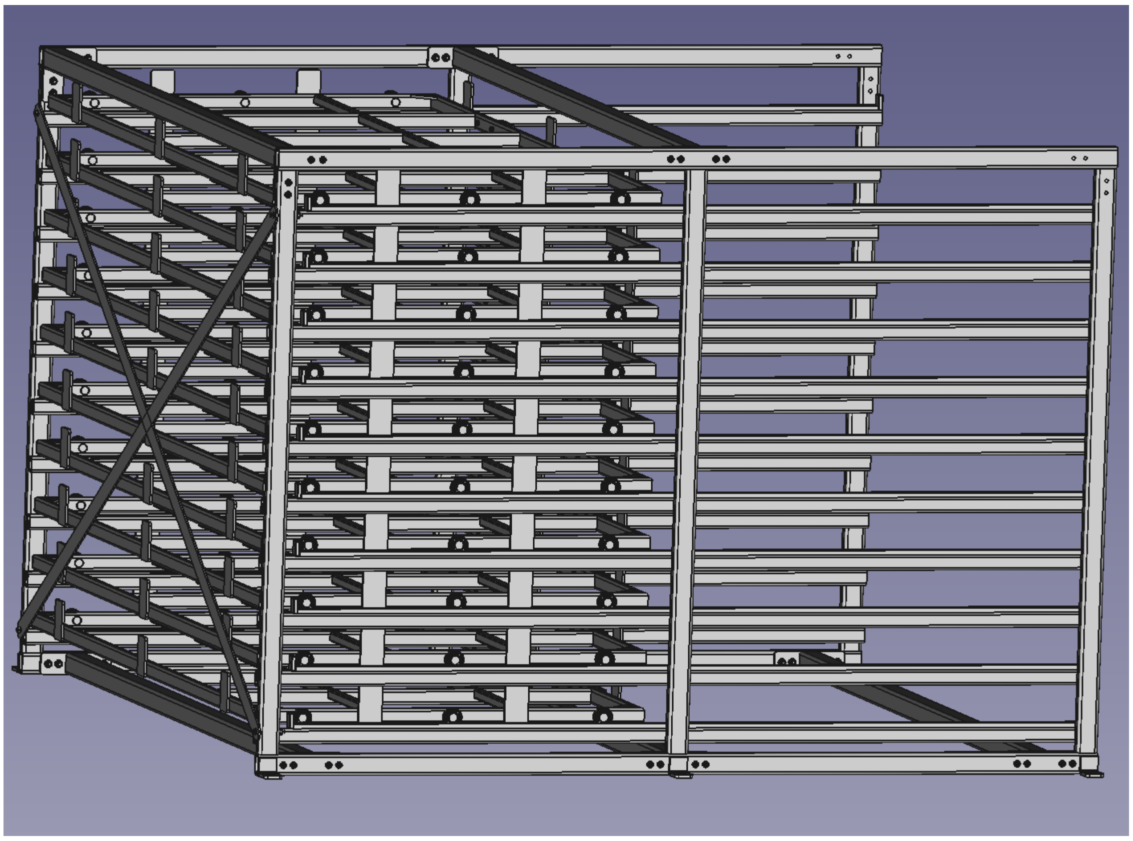


Figure 23: Sheet metal rack

Furthermore, it was observed that some non- UTF-8 characters in the STEP file lead to the pipeline being unable to extract parts correctly.

6 Discussion

After analyzing the results, some representable samples have to be discussed. Images are taken from the stl mesh not from the view the model saw making the classification to enable verification. In the case of the power socket, the step file is also shown to present the power socket assembly including the nut [Figure 26].

6.1 False Positives

Figure 24 depicts a ‘lead screw’ that has been extracted from a workbench clamp CAD assembly and was classified into the screw class. Even though the name suggests so and its geometry and function feature clear similarities with that of a screw, this is not a standard fastener.

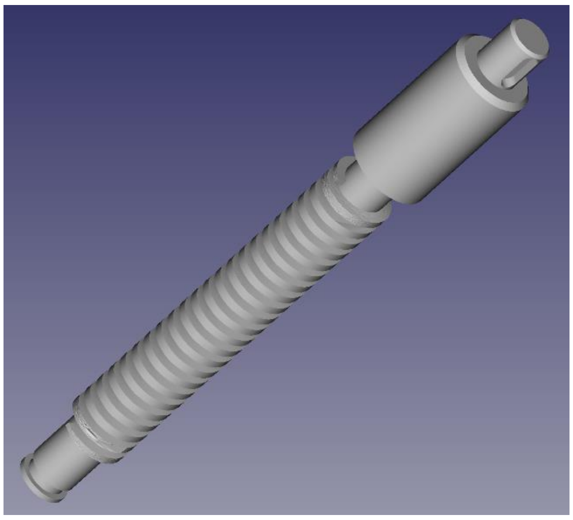


Figure 24: Workbench clamp

Figure 25 was falsely classifies as screw, even it does not look like a screw from this perspective. But when rotated along its length the object in fact comes very close to resembling a screw.

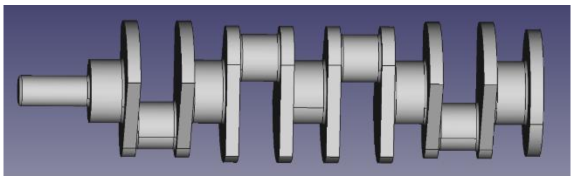


Figure 25: Motor crankshaft

For the model to classify these two rather big components as screws, it seems the classifier has learned some general idea about screw geometries. The “head” area of these objects, however, does not feature any of the common driver shapes.

Figure 26 on the left presents a geometry very much resembling a screw, while the label ‘power socket’ in the classification result csv file already gives away the misclassification. The right image shows the head-like portion with an unusual driver shape. Of course this part could be considered a screw which is apparent by by threads and the nut fitted to fasten the power socket to the device chassis. However, it cannot be considered a standard component and cannot be driven by a regular screwdriver.

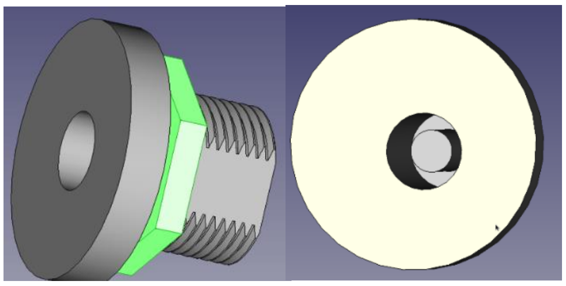


Figure 26: Threaded Power Socket with Counter Screw Nut

On Figure 27 a part of the exhaust pipe of a vehicle is displayed. Again, it can be seen how such a shape from limited views can be classified as screw.

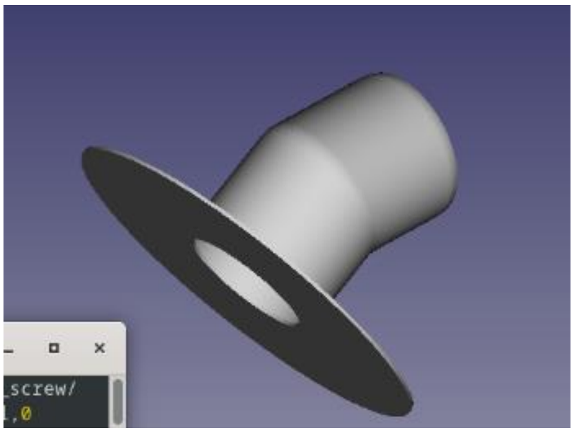

```
.screw/
1,0
```
Figure 27: Exhaust pipe

Figure 28 depicts an IO-pin from an electrical connection. Such an object clearly features strong resemblance in geometry with a square-headed bolt [Table 1], even while the “head” is slightly off-centered.

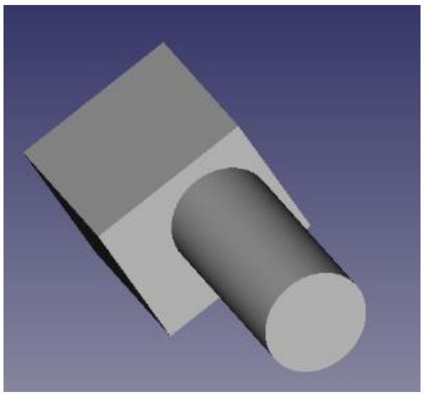


Figure 28: Electrical IO Pin

6.2 False Negatives

For some reason a single totally normal screw was never being classified as screw. This might point to over training.

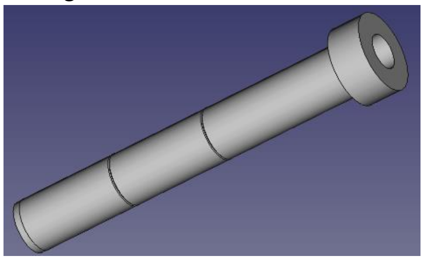


Figure 29: Pin Tube

6.3 Implications

6.3.1 View Padding

Analyzing these falsely identified samples, this can lead to question the decision to scale the objects to the window-size without respect to aspect ratio in the view-creation process. The MVCNN usually accepts a square image so padding is generally common practice. This, however, was seen unfit during design of the view generation process as the resolution was already chosen fine enough to safely recognize the geometric shape of screws, specifically driver shape. Padding a 2D image would keep the aspect ratio at the cost of a lower resolution of the axis of the smaller dimension. This resolution loss depends on the asymmetry of the two dimensions, which is usually very high in screws [Figure 31]. The top two pictures showcase the side view difference with padding and without padding, bottom two images show the respective head views comparing padding with scaling. In the original approach views where not padded in order to prevent resolution especially of the head view. When reviewing the decision, padding can possibly be applied only to those views with a aspect ratio except 1. When looking straight onto the head the aspect ratio of such a view would be roughly square shaped. Hence, no or little padding would be needed in such case and the resolution would not be impacted severely in respect to the head view. It can be expected to be utilizable for detection of the driver classification.

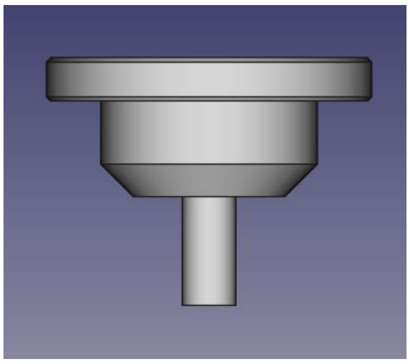


Figure 30: Funnel-shaped spray nozzle

6.3.2 Normalizing Rotation

Shape rotation is already employed before view extraction, but only to compare shapes and exclude duplicate shapes. View data is not normalized with respect to rotation. This was intended on design due to the way how angled views are better able to detect shapes and maybe more importantly, since there was no real way of knowing in which direction the design was created. The results especially failed samples have given further insight that there might be more and less fortunate views. Partly rotation normalization is handled selecting those views that have higher weight to accurately view the head [see topic View Creation], this does not exclude slightly tilt views.

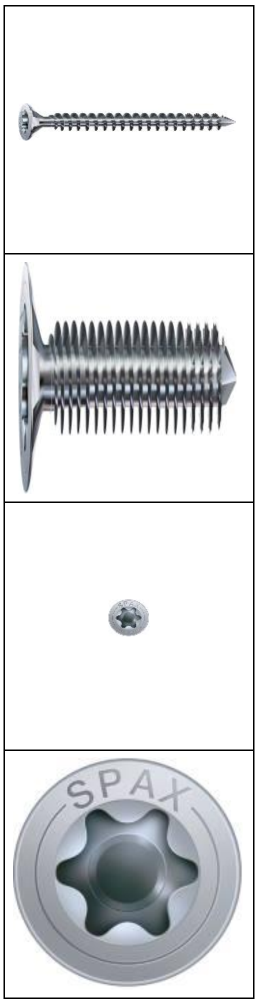


Figure 31: View Padding Concept, after [26]

6.3.3 Pre-Trained Model

EfficientNet and Resnet have been attempted. Applying YOLOv7 object classification on the views before concatenation of the data

6.3.4 Reducing views only to focus on head-shape

6.3.5 Enhancement of views by utilization of RGB color spectrum:

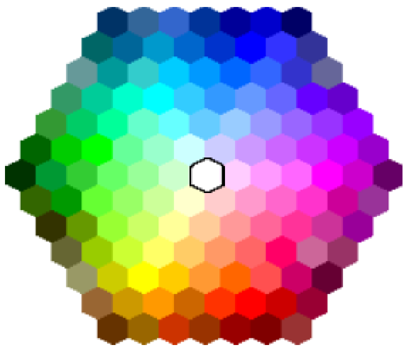


Figure 32: color picker

With regard to lacking in 2D view of 3D objects Depth information can be incorporated into the views via the color channels of the image. Tilt angles of faces are neither uniquely displayable nor distinguished by a grayscale gradient. Since only grayscale is used in this image, the angled light sources give a limited insight into the three-dimensional shape displayed. This brings up the idea of arranging three virtual light sources, a red, a green and a blue in the room in such a way that the intensity of the mixture of light on any pixel gives additional information compared to the grayscale shadows in the available form. The color palette Figure 32, after [38] offers more than only intensity of light and could probably encode depth and orientation of a surface clearer.

7 Conclusion & Future Work

This thesis concludes that machine learning is able to aid component recognition in CAD assemblies given in STEP file to a meaningful level of confidence. This was shown especially for screw fasteners driven by head drives with the aid of a custom and non-pre-trained MVCNN network.

Strength of the approach are the detection and automatic removal of multiple identical parts and shapes, while weaknesses are mainly the lack of training samples and potential need for normalization or further training data.

Could be easily expanded to accept a variety of different input file types, weight and dimension data could be extracted to features. That might help the training.

Multi-layered recognition should be attempted on the basis of detecting whether a shape is a screw, detecting head and consequently the driver shape and ultimately the threading.

Affidavit

8 References

[1] Z. Jia, A. Bhatia, R. M. Aronson, D. Bourne, and M. T. Mason, "A Survey of Automated Threaded Fastening," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 1, pp. 298–310, 2019, doi: 10.1109/TASE.2018.2835382.

[2] A. Neb, I. Briki, and R. Schoenhof, "Development of a neural network to recognize standards and features from 3D CAD models," *Procedia CIRP*, vol. 93, pp. 1429–1434, 2020, doi: 10.1016/j.procir.2020.03.010.

[3] B. Bigoney, "Automatic Drilling and Fastening System for Large Aircraft Doors," in *SAE Technical Paper Series*, 2019.

[4] M. Alemanni, F. Destefanis, and E. Vezzetti, "Model-based definition design in the product lifecycle management scenario," *Int J Adv Manuf Technol*, vol. 52, 1-4, pp. 1–14, 2011, doi: 10.1007/s00170-010-2699-y.

[5] M. J. Pratt, "A New ISO 10303 (STEP) Resource for Modeling Parameterization and Constraints," *Journal of Computing and Information Science in Engineering*, vol. 4, no. 4, pp. 339–351, 2004, doi: 10.1115/1.1814383.

[6] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller, "Multi-View Convolutional Neural Networks for 3D Shape Recognition," [Online]. Available: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Su_Multi-View_Convolutional_Neural_ICCV_2015_paper.pdf

[7] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A Deeper Look at 3D Shape Classifiers," Sep. 2018. Accessed: Sep. 28 2024. [Online]. Available: https://arxiv.org/pdf/1809.02560

[8] A. Teichman and S. Thrun, "Practical object recognition in autonomous driving and beyond," in *Advanced Robotics and its Social Impacts*, 2011, pp. 35–38.

[9] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015, pp. 922–928.

[10] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/papers/Qi_PointNet_Deep_Learning_CVPR_2017_paper.pdf

[11] Q. Yu, C. Yang, H. Fan, and H. Wei, "Latent-MVCNN: 3D Shape Recognition Using Multiple Views from Pre-defined or Random Viewpoints," *Neural Process Lett*, vol. 52, no. 1, pp. 581–602, 2020, doi: 10.1007/s11063-020-10268-x.

[12] K. J. Liang *et al.,* "Toward Automatic Threat Recognition for Airport X-ray Baggage Screening with Deep Convolutional Object Detection," Dec. 2019. Accessed: Sep. 28 2024. [Online]. Available: https://arxiv.org/pdf/1912.06329

[13] V. Riffo, I. Godoy, and D. Mery, "Handgun Detection in Single-Spectrum Multiple X-ray Views Based on 3D Object Recognition," *J Nondestruct Eval*, vol. 38, no. 3, 2019, doi: 10.1007/s10921-019-0602-9.

[14] Y. You *et al.,* "Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving," Jun. 2019. Accessed: Sep. 30 2024. [Online]. Available: https://arxiv.org/pdf/1906.06310

[15] S. Mangold, C. Steiner, M. Friedmann, and J. Fleischer, "Vision-Based Screw Head Detection for Automated Disassembly for Remanufacturing," *Procedia CIRP*, vol. 105, pp. 1–6, 2022, doi: 10.1016/j.procir.2022.02.001.

[16] L. H. Finkel and P. Sajda, "Object Discrimination Based on Depth-from-Occlusion," *Neural Computation*, vol. 4, no. 6, pp. 901–921, 1992, doi: 10.1162/neco.1992.4.6.901.
[17] Twin Creeks Log Home Supply, *Spax PowerLag Screws - T-Star Pancake Head.* [Online]. Available: https://www.twincreeksloghomes.com/products/spax-powerlag-log-and-timber-screws-xtm-pancake-head (accessed: Sep. 30 2024).
[18] *square head bolt - Pesquisa Google.* [Online]. Available: https://www.google.com/search?client=firefox-b-d&q=square+head+bolt#imgrc=002G09-OzpZFhM&imgdii=i6GpK9xhj01wpM (accessed: Sep. 30 2024).
[19] *4.8 X 25 PPA2DX S50 DIN7504 - PAN HEAD SCREW, SS, 4.8MM X 25MM.* [Online]. Available: https://de.farnell.com/en-DE/tr-fastenings/4-8-x-25-ppa2dx-s50-din7504/pan-head-screw-ss-4-8mm-x-25mm/dp/3442481 (accessed: Sep. 30 2024).
[20] *Accessories / Shoulder Screw.* [Online]. Available: https://www.alphanovatech.com/en/cat_c_sse.html (accessed: Sep. 30 2024).
[21] *Fastener Guides - Screws Drive Styles | Fastener SuperStore | Bulk Industrial Fasteners.* [Online]. Available: https://www.fastenersuperstore.com/fastener-guides/screws-drive-styles (accessed: Sep. 30 2024).
[22] *Fastener Guides - Screws Head Styles | Fastener SuperStore | Bulk Industrial Fasteners.* [Online]. Available: https://www.fastenersuperstore.com/fastener-guides/screws-head-styles?srsltid=AfmBOorldqM1mW95BHaMv9hyfDttMBeKjbtY8CK6nJuBWRhNDsxesP2s (accessed: Sep. 30 2024).
[23] The Engineering Choice, *Different Types Of Threads And Their Uses.* [Online]. Available: https://www.theengineeringchoice.com/types-of-threads/ (accessed: Sep. 30 2024).
[24] JakeSales.com, *#9 Silver Star Stainless Steel TRIM HEAD Screw Torx/Star Head - Stainl.* [Online]. Available: https://jakesales.com/products/9-silver-star-stainless-steel-trim-head-screw-torx-star-head-stainless-steel-trim-head-wood-screws-grade-305-stainless-steel-torx-star-drive-screws (accessed: Sep. 30 2024).
[25] *Construction Screws | Heavy Duty Timber Screws | Fastco | Fastco.* [Online]. Available: https://www.fastco.co.uk/forgefast-construction-screw-wafer-head-tan-8-x-160mm-pack-of-30.html (accessed: Sep. 30 2024).
[26] Der Schraubenladen Online Shop, *Spax Universalschraube, Vollgewinde, Senkkopf, WIROX T-STAR plus kaufen.* [Online]. Available: https://shop.der-schraubenladen.de/Spax-Universalschraube-Vollgewinde-Senkkopf-WIROX-T-STAR-plus/SW11030.715 (accessed: Sep. 30 2024).
[27] EFC International, *Item # 24A000312-302, SINULOC&#174; Machine Screws On EFC International.* [Online]. Available: https://www.efc-intl.com/item/thread-forming/sinuloc-machine-screws/24a000312-302 (accessed: Sep. 30 2024).
[28] "Datasheet - ForgeFast Construction Screw," [Online]. Available: https://www.fastco.co.uk/amfile/file/download/file/72/product/2879/
[29] *#00 x 3/8" U Drive Self-Tapping Screws / Round Head / Steel / Zinc.* [Online]. Available: https://www.fastenersuperstore.com/products/100272/self-tapping-screws?pid=18363 (accessed: Sep. 30 2024).
[30] GitHub, *[Problem] Assembly: duplicate parts are renamed · Issue #12139 · FreeCAD/FreeCAD.* [Online]. Available: https://github.com/FreeCAD/FreeCAD/issues/12139 (accessed: Sep. 27 2024).
[31] N. Ojal, B. Giera, K. T. Devlugt, A. W. Jaycox, and A. Blum, "A universal method to compare parts from STEP files," *J Intell Manuf*, vol. 33, no. 7, pp. 2167–2178, 2022, doi: 10.1007/s10845-022-01984-3.

[32] *Earth Globe World Logo, earth, globe, logo png | PNGEgg* (accessed: Sep. 28 2024).

[33] *Meshes, black, png | PNGEgg* (accessed: Sep. 30 2024).

[34] V. Sawant, "3D Cad to Binary Voxel Numpy Array - Analytics Vidhya - Medium," *Analytics Vidhya*, 22 Apr., 2021. https://medium.com/analytics-vidhya/3d-cad-to-binary-voxel-numpy-array-b538d00d97da (accessed: Sep. 28 2024).

[35] *GitHub - dcunni9/STP-STEP-to-STL-Python-Converter: Simple multithreaded STL to STP Python Converter based on Freecad.* [Online]. Available: https://github.com/dcunni9/STP-STEP-to-STL-Python-Converter/tree/main (accessed: Sep. 29 2024).

[36] GitHub, *voxelization to 3D numpy.ndarray · pyvista/pyvista · Discussion #5652.* [Online]. Available: https://github.com/pyvista/pyvista/discussions/5652 (accessed: Sep. 28 2024).

[37] I. Nikolov, "How to Voxelize Meshes and Point Clouds in Python - Towards Data Science," *Towards Data Science*, 31 May., 2022. https://towardsdatascience.com/how-to-voxelize-meshes-and-point-clouds-in-python-ca94d403f81d (accessed: Sep. 28 2024).

[38] *HTML Color Picker.* [Online]. Available: https://www.w3schools.com/colors/colors_picker.asp (accessed: Sep. 28 2024).