Summer Internship Project
Report

Unsupervised Shape Recovery from 2D images applied to plastic packaging

Submitted by

Ayoub EL HOUDRI

CY Tech

Under the guidance of

Dr. Katharina Eissing

Digimind

Department of Research and Development

Digimind GmbH
Bismarckstraße 10-12
10625 Berlin

Summer Internship 2021

Department of Mathematics & Computer Science

CY Tech

Name of the guide
Prof. Pierre Plancoulaine

Date: August 20, 2021

Abstract

Included in this paper are accounts of my internship as a Data Scientist Intern undertaken in Digimind Labs in Berlin in the fulfillment of my first year of engineering studies at CY Tech. I was in charge of Shape reconstruction of plastic packaging using Computer Vision & Machine Learning methods as well as generating synthetic data using some of mesh and CAD manipulation libraries using Python as the main programming language, with emphasis placed on plastic bottles. The internship at Digimind Labs provided me with the opportunity to work with different people coming from different fields within the team. Further work at Digimind Labs included research in the field of Computer Vision especially in Shape Reconstructed AI based, presenting my recommendations for some algorithms improvement for better shape reconstruction from 2D images, and then implementing those ideas.

Contents

Chapter 1

Objective

The main objective of this internship project is to generate detailed 3D bottles from a single view 2D images. To accomplish this task the most used algorithms are AI based: using Graph Convolutional Neural Networks known in field as GCNNs. So as to do that, at first, we had to chose the most efficient model in term of high quality shape reconstruction from several models appearing in some research papers. After that we had to evaluate the pre-trained model's complexity and results using images of bottles scrapped from the web and preprocessed to be a ready-to-use input.

To improve the output shapes we focused on training the model on plastic bottles only instead of using the available pre-trained model which is trained not only on bottles but also on large categories of objects from ShapeNet [1] data set such as planes, cars, guns and many other shapes.

Besides to gaining a first professional experience, I was looking forward to improve my soft skills, to discover the teamwork culture, and to try a new experience of working abroad, in a technologically advanced country such as Germany.

Chapter 2

Introduction

In this report, I underline the several missions and problems I had to deal with during my internship at Digimind Labs. I will also talk about my motivation behind doing this internship and the reasons which pushed me to choose Digimind Labs to carry out my internship project.

In what comes, first of all, I will introduce the company and its field of activity, my career plan, and my work experience being in a team in a foreign company. Second of all, I will mention the technical work I had to achieve including all the algorithms and libraries used for that. Finally, I will indicate the future work ranging from generating data to training the algorithm.

At the end, I will present a conclusion as an overview of the technical and nontechnical skills I developed during my internship at Digimind Labs.

Chapter 3

The Startup: Digimind Labs

3.1 Introduction of Company

3.1.1 Activity field and goals

Digimind [2] is a startup based in Berlin, founded by Dr. Omar Fergani and Dr. Katharina Eissing in 2018. The company is developing an End-to-end AI platform to accelerate packaging innovation, considered the first company introducing a SaaS tool dedicated to the packaging industry. Its goal is to enable the transition of the packaging from a linear to a fully circular economy by 2025 and achieving this will be with leveraging Digimind's in-house AI software and material libraries, reducing material usage, increasing reuse potential, and enabling recyclability by design.

Through its platform Digimind Labs provides many services to its customers, like the following:

1. Analysis & Specification: Digimind supports customer at the start of its journey by analyzing his product and its life-cycle to gain all the necessary specifications. The specifications are translated into detailed engineering requirements for the product and manufacturing process. The requirements are specified to meet the high demands set by production, filling, transportation and handling as well as his production capacities.

2. Digitalization of the product: To help the customer to leverage the full power of Digimind Toolbox, the company digitalizes the product and performs using its computer topography technology.

3. AI Design & Light-weighting: Whether the costumer is looking for a completely new design or for the optimization of his existing design to make it more sustainable and cost saving while fulfilling all his performance requirements, Digimind can take this in hand.

4. Customer specific AI algorithm: Joined development for the customer's specific material based AI algorithm or a new machine learning model for his specific type of packaging.

5. Design testing & Verification: To ensure the customer's new or optimized design meets all the requirements set by production, filling, transportation and handling and his production capacities, Digimind tests and verifies the product performance with its GreenDigitalTwin technology. They also work with the customer in producing and testing a prototype before mass production.

6. Sustainability & Cost insights: With Digimind's ISO 14040/44 Life Cycle Assessment service, the insights needed on products environmental performance throughout its entire life-cycle are given to the customer. A Cost Insights Report provides him with an immediate overview of the cost savings achieved with new or optimized designs.

3.1.2 Overview of Digimind's activity

I present in Figure 1 an overview of the chain developed by Digimind and used to perform high quality pre-manufacturing tasks through its end-to-end platform.

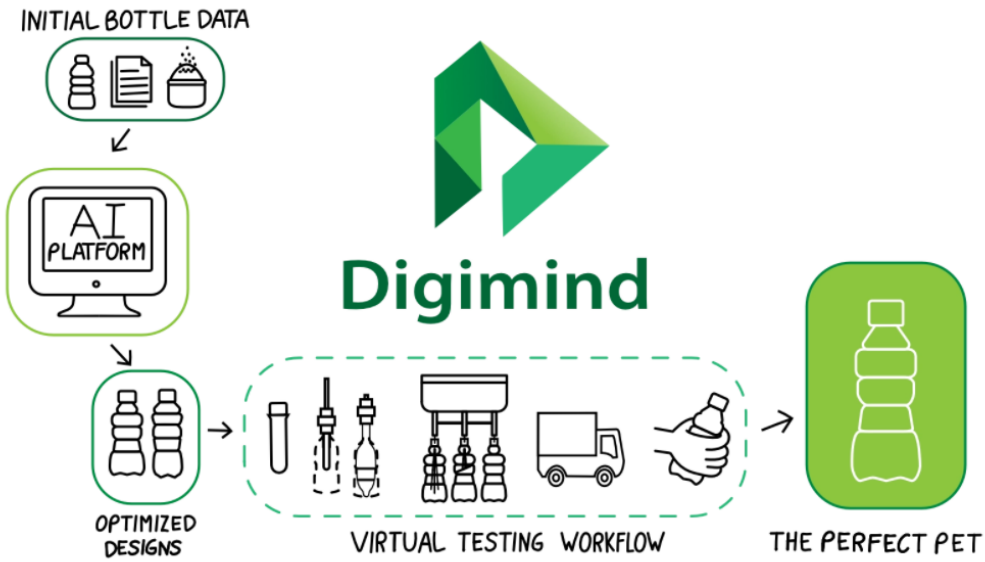


Figure 1: Overview of Digimind's design chain

The following Figure 2 shows how Digimind reduces the environmental impact by reducing the carbon footprint.

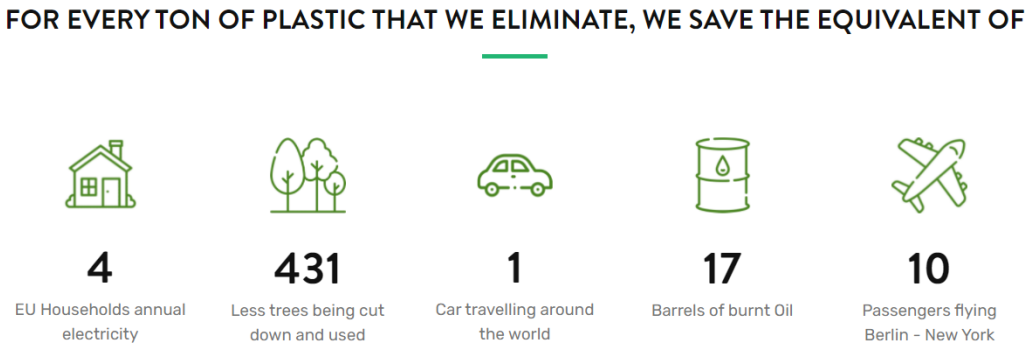


Figure 2: Environmental impact reduction by Digimind

3.2 Motivation behind choosing Digimind for internship

3.2.1 Career plan

My choice of Mathematics & Computer Science branch in the second semester of my first year of engineering at CY Tech [3] justifies my desire to become a Data Scientist. I acquired a lot of knowledge in the field at school and during this 3 months internship within the Digimind team.

For any kind of technical role, including data science, there are two paths: the management path and the individual contributor path.

The individual contributor path includes data scientists who work on core projects, contribute code, run analyses, and build ETL pipelines and machine learning models. The management path encompasses data scientists who manage people, scale data strategy, and work on fitting the pieces of a data organization together.

Both paths originate from the same journey from entry-level to the senior data scientist position, where they then diverge. As career advancement continues, individual contributors can decide to become managers or remain highly specialized data scientists.

When it comes to my future choice, I found that being a data scientist manager suits me the most. Through my internship I developed tasks organization skills, communication skills, problems solving and hacking skills, which pushed me to be interested not only on the technical side of the job but also team management.

Digimind gave me a big opportunity to discover the job of data scientist and provided me with a large knowledge and tips to be successful in my future career.

3.2.2 Teamwork

As a data scientist intern at Digimind, I was in the R&D team focusing more on research. I had the opportunity to meet interns like me from all over the world coming from different backgrounds and fields of study such as: Computer Science, Mechanical Engineering, and Mathematical Modeling.

I'm very grateful to work in a heavy-weight team such as Digimind. Teamwork culture was omnipresent and mutual aid was ubiquitous: I learned a lot and helped a lot. I liked the possibility of collaboration between teams from the other departments such as business department, communication department and R&D department.

3.2.3 Internship abroad

Being a member of the Digimind team for 3 months was one of my greatest experiences. Digimind took the initiative by inviting all the team to Berlin to work face to face instead of keeping working fully remote.

I had the chance to meet directly with my colleagues. We had a great experience during our stay in Berlin.

Travelling to Germany, as a part of my internship, made me want to discover the startup's culture in other regions of the world. And I will be glad to retry the same experience abroad in other countries.

Chapter 4

Work Done

4.1 Pixel2Mesh Algorithm

4.1.1 Data and preprocessing

As a first step, I started testing Pixel2Mesh Implementation available on GitHub [4] that came with a research paper untitled Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images [5] , appeared in 2016. To achieve this task I needed at first one image of a bottle to be used as an input.

I scrapped some images using python BeautifulSoup library [6] for web scrapping. As a result I created my own data set of images of bottles that can be fed to the algorithm after being preprocessed.

From all the gathered images, I have chosen three images from the initial data set and the second step was preprocessing these images by resizing them and applying some filters on them such as: gray-scale filter and Gaussian blur filter for noise removal as well as reducing memory consumption.

The main used library for the preprocessing implementation is OpenCV [7], which contains ready-to-use functions for image processing. Below is the preprocessing code with all the function filters cited before as well as some functions for image reading and result plotting.

The following script is used to import some libraries and to define **loadImages** function to read the initial images from their paths.

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2

image_path = "/home/cytech/Desktop"

def loadImages(path):
    image_files = sorted([os.path.join(path, 'bottles', file)
    for file in os.listdir(path + "/bottles") if file.
   endswith('.png')])
    return image_files
```

The next script is defining two functions **display_one** and **display** to plot the image before and after processing it.

```
 # Display one image
def display_one(a, title1 = "Original"):
    plt.imshow(a), plt.title(title1)
    plt.xticks([]), plt.yticks([])
    plt.show()

# Display two images
def display(a, b, title1 = "Original", title2 = "Edited"):
    plt.subplot(121), plt.imshow(a), plt.title(title1)
    plt.xticks([]), plt.yticks([])
    plt.subplot(122), plt.imshow(b), plt.title(title2)
    plt.xticks([]), plt.yticks([])
    plt.show()
```

This part of the code is the main function of **processing**, it contains 3 steps: resizing, gray-scaling and blurring, it will be applied on a sample of three images from the data set.

```
def processing(data):

    # Loading image and getting 3 images from the data set to
    work with
    img = [cv2.imread(i, cv2.IMREAD_UNCHANGED) for i in data
   [:3]]
    print('Original size', img[0].shape)

```

```
    # Setting dim of the resize = (255,255)
    height = 220
    width = 220
    dim = (width, height)
    res_img = []
    for i in range(len(img)):
        res = cv2.resize(img[i], dim, interpolation=cv2.
   INTER_LINEAR)
        res_img.append(res) # res is the resized image

    # Gray-scaling filter
    gray_scale = []
    for i in range(len(res_img)):
        gray = cv2.cvtColor(res_img[i], cv2.COLOR_BGR2GRAY)
    gray_scale.append(gray) # gray is the gray scaled image

    # GaussianBlur filter (noise removal)
    no_noise = []
    for i in range(len(gray_scale)):
        blur = cv2.GaussianBlur(gray_scale[i], (5, 5), 0)
        no_noise.append(blur) # blur is the new image without
    noise

    # Display the original and the edited images
    for i in range(3):
      display (img[i], no_noise[i])
```

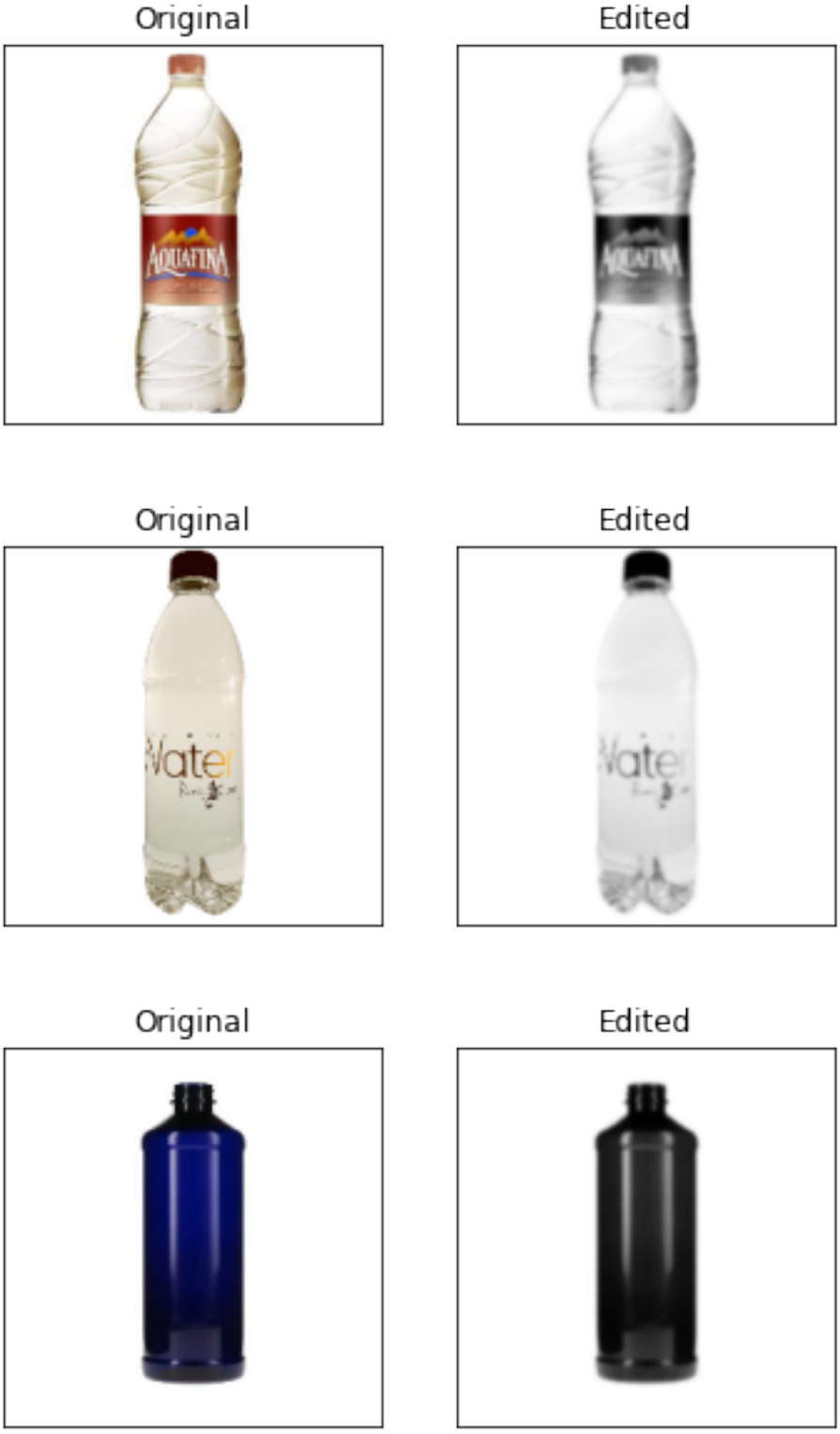
The last part is defining **main** as a function to run **processing** and to plot the Original and Edited images.

```
def main():

    # Calling global variable
    global image_path
    dataset = loadImages(image_path)
    print("List of files the first 3 in the folder:\n",
   dataset[:3])
    # sending all the images to pre-processing
    pro = processing(dataset)

main()
```

Below are the output with each input of the preprocessing script above.


Original
Edited
AQUAFINA
AQUAFINA
Original
Edited
Water
Water
Original
Edited

4.1.2 Pixel2Mesh result

The official implementation of Pixel2Mesh is available on GitHub [4] where the provided input is the preprocessed image of bottle. Here are some 3D shapes generated by the algorithm after giving it the edited images above in same order.

Figure 3: Original image - Edited input - 3D shape

Figure 4: Original image - Edited input - 3D shape

Figure 5: Original image - Edited input - 3D shape

We notice that the results are missing details. This is due to the pre-trained model which is normally trained on various categories of objects other than bottles, so to accomplish this task of reconstructing the 3D shape with more details and features, it is necessary to train the algorithm only on a large data set of bottle images and their 3D shapes, to make it adapted to bottles only.

So our goal now is to generate data in 3D form of bottles and use it to train the algorithm instead of using a pre-trained model. That is what we will see next.

4.2 Pygalmesh

4.2.1 Pygalmesh library

Pygalmesh [8] is a Python front-end to CGAL's 2D and 3D mesh generation capabilities. Pygalmesh makes it easy to create high-quality 2D, 3D volume meshes, periodic volume meshes, and surface meshes. We will use it to generate the 3D shape of bottles with rotational symmetry, using the function **RingExtrude**.

The reconstruction is done through 3 steps: contour coordinates extraction, data reduction and finally revolving the 2D polygon (contour) 360 degrees around the central axis. In the next section, I present the initial image as well as the scripts for each step and their output.

4.2.2 Implementation and results

Figure 6: Initial image

- **Step 1: Contour extraction**

```
1 import cv2 # Import OpenCV
2 import numpy as np # Import NumPy
3 import statistics # Import Statstics to use built-in
     avergae funtion
4 import pygalmesh # Import Pygalmesh
5 import matplotlib.pyplot as plt #Import Pyplot to plot
     the contour
6
```

```
# Read the image as grayscale
im = cv2.imread('CADQuery_Target-removebg-preview.png', 0)

# Run findContours
# Also, we want to find the best contour possible with CHAIN_APPROX_NONE
contours, hierarchy = cv2.findContours(im.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# Create an output of all zeroes that has the same shape as the input
# image
out = np.zeros_like(im)

# On this output, draw all of the contours that we have detected
# in white, and set the thickness to be 3 pixels
cv2.drawContours(out, contours, -1, 255, 3)

# Save the contour output
cv2.imwrite('bottle_edge.png',out)

#0.01 for normalization, to avoid having far points
#the i%10 to reduce the number of points: we take 112 points to reduce memory consumption
contours0 = [[c[0][0]*0.1,c[0][1]*0.1] for i,c in enumerate(contours[0]) if i%10==0]
contour0 #Normalized coordinates of the contour
```

Figure 7: Extracted contour

- **Step 2: Data reduction**

```
#X, Z are the sets of the contour's coordinates
X=[c[0] for c in contours0]
Z=[c[1] for c in contours0]

#We substract the mean of the x-coordianates from x-
   coordiantes, to center the bottle on the Z-axis
avg=statistics.mean(X)
X=list(X-avg)

contours1=[[X[i],Z[i]] for i in range(len(X))]

#In order to optimize the memory consumption we use just
   the points having a positive  x-coordinates
#(half the bottle)
contours_final=[]

for c in contours1:
    if c[0]>=0:
      contours_final.append(c)

#We plot the points to make sure that we deal only with
   half the bottle
X=[c[0] for c in contours_final]
Z=[c[1] for c in contours_final]

plt.plot(X, Z, 'o', color='black')
```

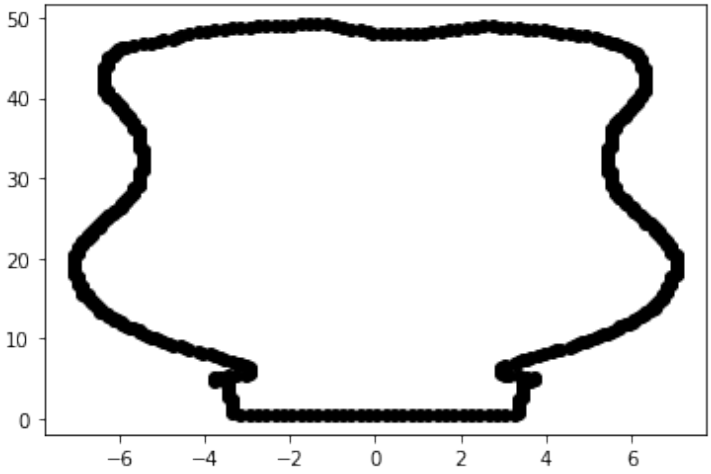


Figure 8: Before reducing the contour coordinates

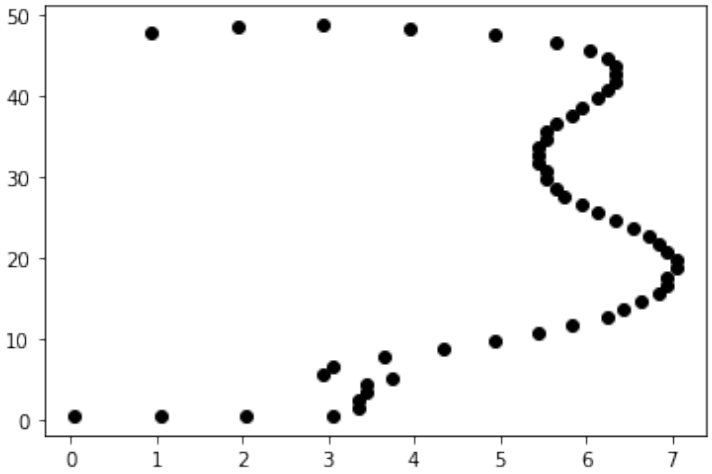


Figure 9: After reducing the contour coordinates

- **Step 3: Shape reconstruction**

```
p = pygalmesh.Polygon2D(contours_final)

max_edge_size_at_feature_edges = 0.1

domain = pygalmesh.RingExtrude(p,
    max_edge_size_at_feature_edges)

mesh = pygalmesh.generate_mesh(
    domain,
    max_cell_circumradius=0.1,
    max_edge_size_at_feature_edges=
    max_edge_size_at_feature_edges,
    verbose=False,
)

mesh.write("my_sweet_bottle.vtk")
```

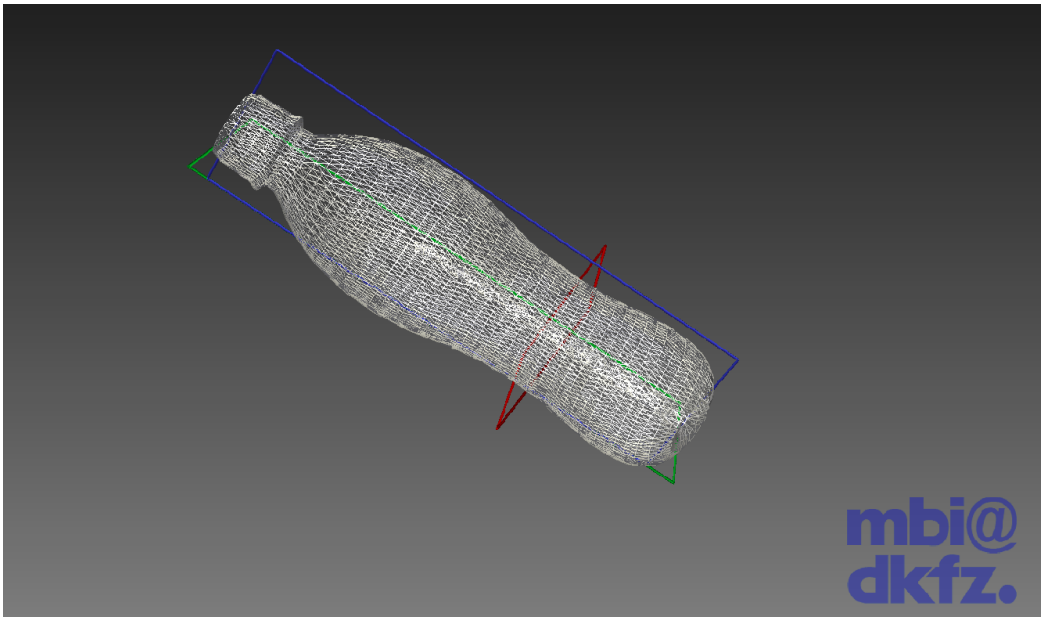


Figure 10: Final 3D shape

Applying this method to reconstruct the 3D shape of rotational symmetric bottles will help us in making a data set of this type of bottles and use them to train Pixel2Mesh algorithm.

4.3 CADQuery Library

4.3.1 Why CADQuery ?

CadQuery [9] is an intuitive, easy-to-use Python library for building parametric 3D CAD models. It has several goals:

- Build models with scripts that are as close as possible to how you would describe the object to a human, using a standard, already established programming language.
- Create parametric models that can be very easily customized by end users.
- Output high quality CAD formats like STEP and AMF in addition to traditional STL.

- Provide a non-proprietary, plain text model format that can be edited and executed with only a web browser.
- Unlike Pygalmesh library (seen in the section before), CADQuery can generate 3D shapes of rectangular bottles as well as bottles without rotational symmetry.

4.3.2 Implementation and results

The following implementation was used to create a parameterized 3D shape of a bottle having the most general shape possible. In what follows I will present the code as well as its result after varying some parameters.

```
import cadquery as cq

#upper_part + bottom_part
def bottle(distZ, distX, bandlengh, bandwidth, boxlengh,
    boxwidth, boxheight):

    s = cq.Workplane("ZX")

    sPnts = [
    (distZ*4,distX),
    (distZ*2,2*distX+0.1),
    (0,distX*3)
    ]

    r = s.lineTo(distZ*5,0).lineTo(distZ*5,distX).lineTo(
    distZ*4+0.01,distX)
    r = r.spline(sPnts,includeCurrent=True).close()

    result=r.revolve(axisStart=(0,0), axisEnd=(1,0),clean=
    True)
    result = result.faces(">Z").shell(0.05)
    global upperpart
    upperpart=result.translate((0,0,2*(boxheight + bandlengh)
    -0.4))

    #bottompart1
    s = cq.Workplane("XY" ).box(boxlengh, boxwidth, boxheight
    ).edges("|Z").fillet(boxlengh // 2)
    #bottompart2
    q=s.faces(">Z").workplane(centerOption="CenterOfMass").
    circle(bandwidth/2).extrude(bandlengh + .3,True)
    #bottompart3:
    q=q.faces(">Z").workplane(centerOption="CenterOfMass").
    box(boxlengh, boxwidth, boxheight + .6).edges("|Z").fillet
```

```
    (boxlengh // 2)
28      #bottompart4:
29      q=q.faces(">Z").workplane(centerOption="CenterOfMass").
    circle(bandwidth/2).extrude(bandlengh,True)
30      #bottompart5:
31      q=q.faces(">Z").workplane(centerOption="CenterOfMass").
    box(boxlengh, boxwidth, boxheight).edges("|Z").fillet(
    boxlengh // 2)
32      q=q.faces("<Z").fillet(0.5)
33      global bottompart
34      bottompart=q.faces(">Z").shell(0.05)
35
36      return  upperpart,  bottompart
37
38  bottle (0.5, 0.5, 3, 1.2*2, 3, 3, 2.4)
```

The script above gives the following result.

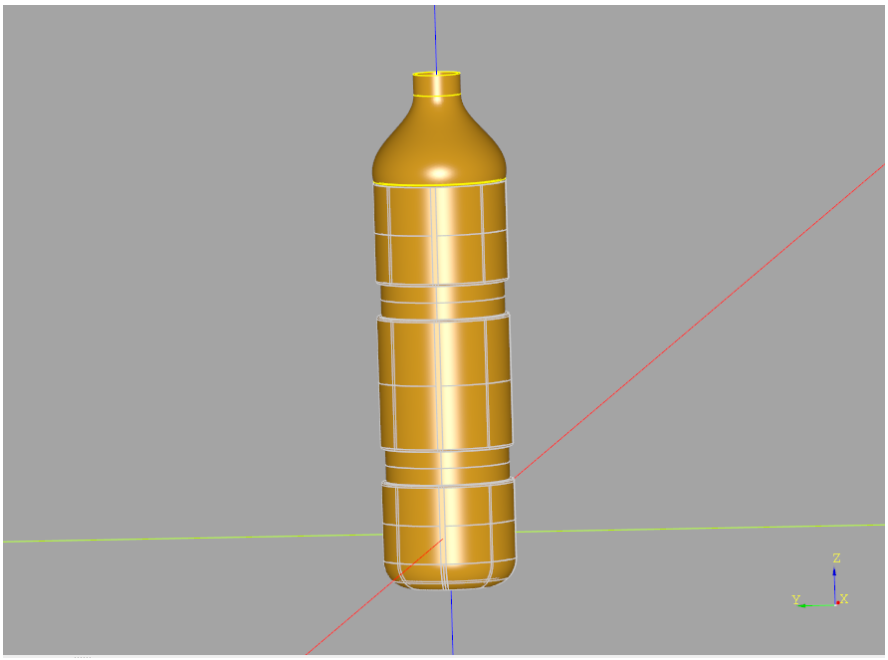


Figure 11: Output (view 1)

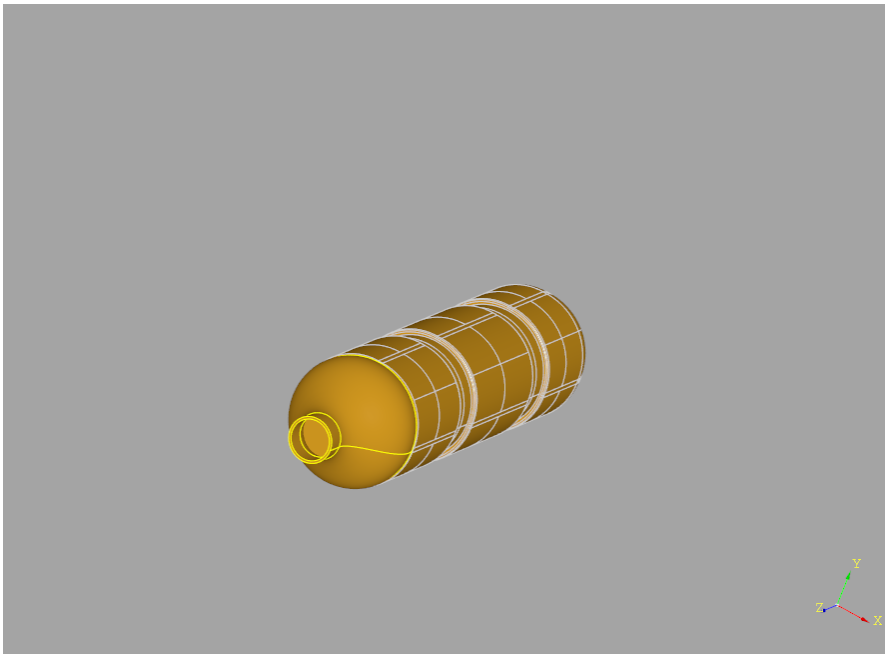


Figure 12: Output (view 2)

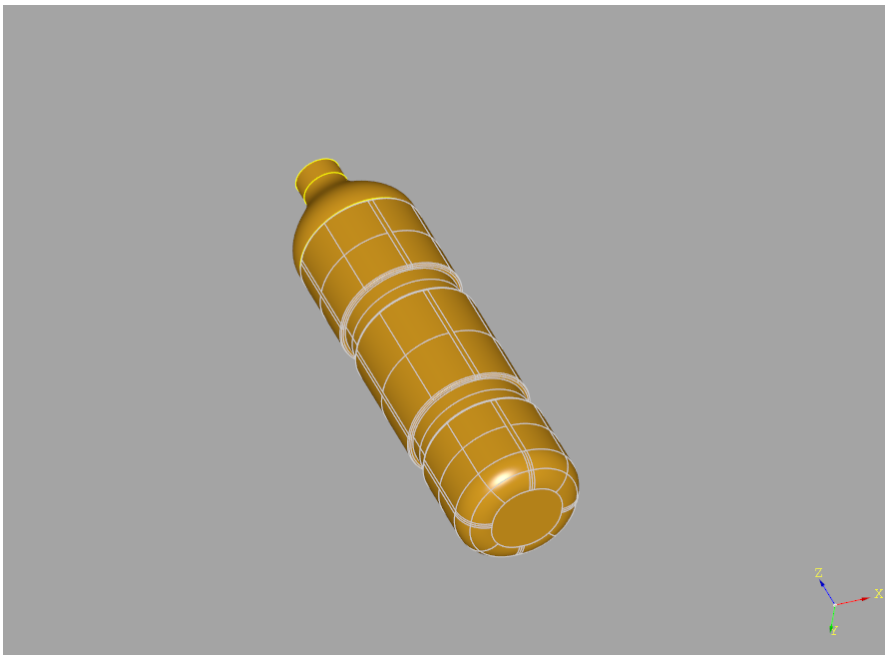


Figure 13: Output (view 3)

4.3.3 Construction blocks

The CADQuery bottle is made from two parts: the upper-part which is the curvy part with the cap at the top, and the bottom-part which is the cylindrical. In the following, I will explain how the two parts of the parameterized bottle are made and define each parameter and how its variation changes the form of the bottle.

- **Upper-part**

  As a first step, I have set a workplane that should be following 2 axes placed in order: Z and X, and then the plan is named s as we see in line 6. This plan will serve as a basis for building blocks one on top of the other until the construction of the entire bottle, part by part which will be explained with more details after.

  As a second step, line 8, sPnts contains all the coordinates of 3 points in the workplane (the 2D plan ZX), the first coordinate is following the Z axis and the second is following the X axis (that's why the order of the axes while defining the workplane is important): the parameters which intervene here are distZ and distX multiplied afterwards by coefficients like 4,2 and 3. distZ represents a distance along the Z axis and distX a distance along the X axis the coefficients only allow to have a larger shape of the upper-part: The multiplication by coefficients is linear and does not affect the shape, they could be even deleted since distZ and distX are local variables.

  Plotting r in line 15 after using the spline function gives the result in the next page.

  The spline function interpolates the 3 points, and the close function closes the spline. The next result is given after setting the parameters distZ and distX to 0.5.

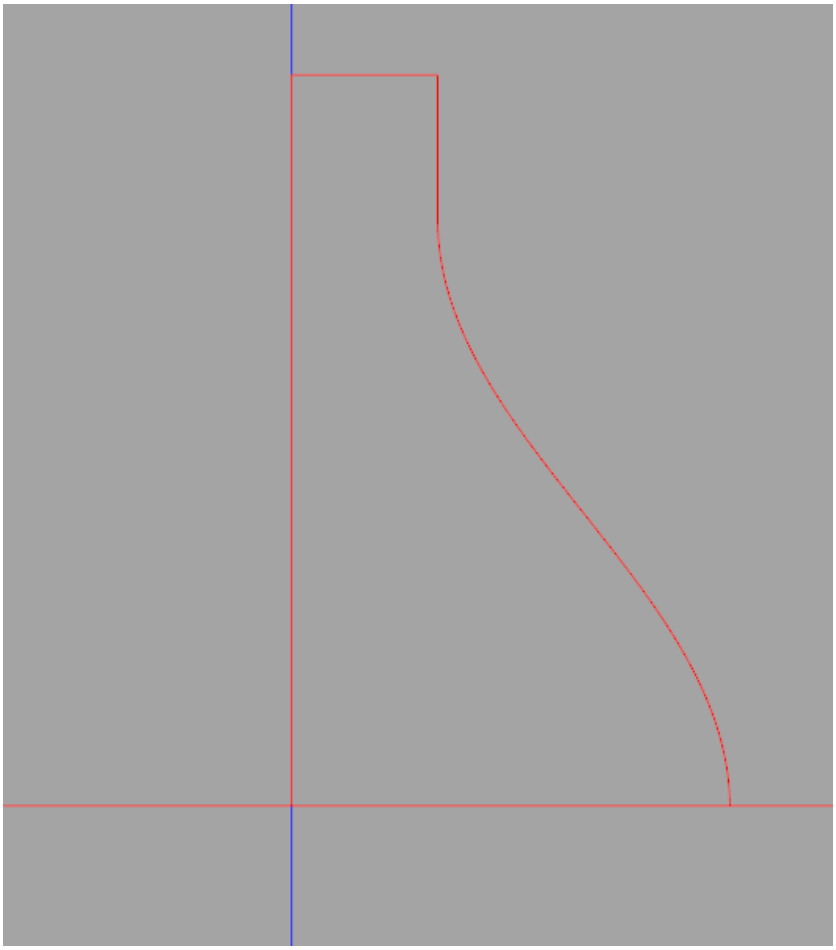


Figure 14: sPnts plot with distZ=0.5, distX=0.5

The revolve function is used after that on this 2D sketch, and takes as variables axisstart and axisend, to make sure that the rotation will be around the Z axis, and takes also a boolean variable: clean set to True, to clean the 2D sketch. As a result we get the following.

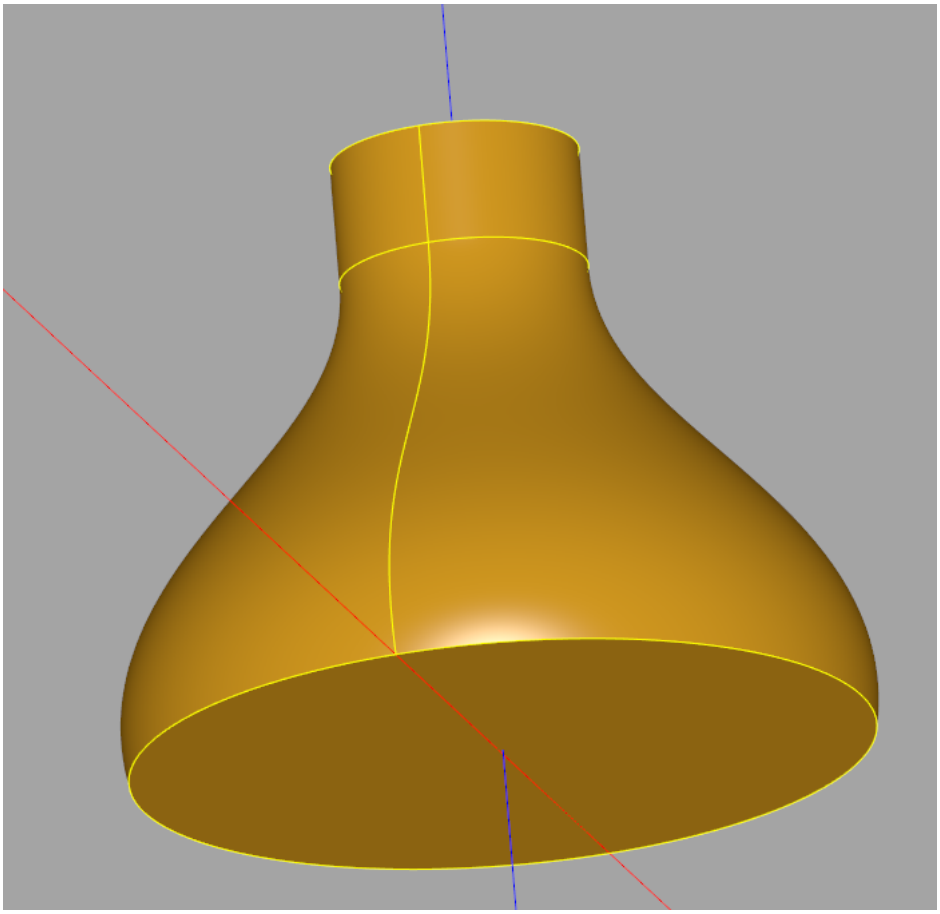


Figure 14: Upper-part with distZ=0.5, distX=0.5 (view 1)

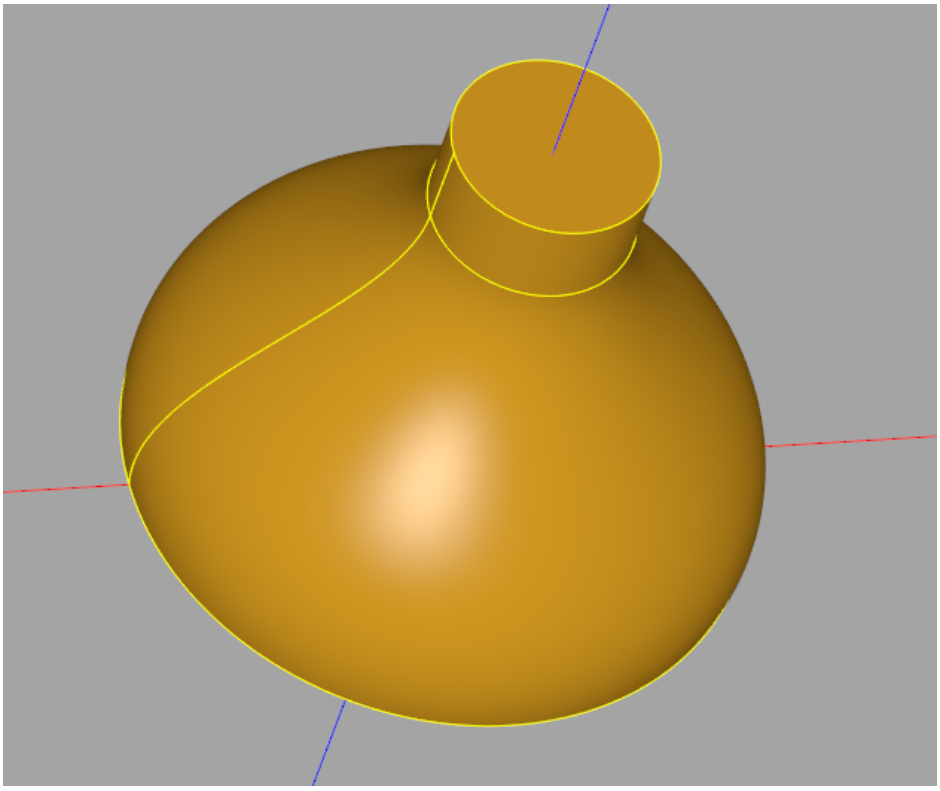


Figure 15: Upper-part with distZ=0.5, distX=0.5 (view 2)

I applied a shell function to this part with a thickness = 0.05, starting from the upper face (faces (“>Z”) in line 18) to the bottom face, creating the bottle’s wall by emptying the inside of the shape above. And gives the next result.

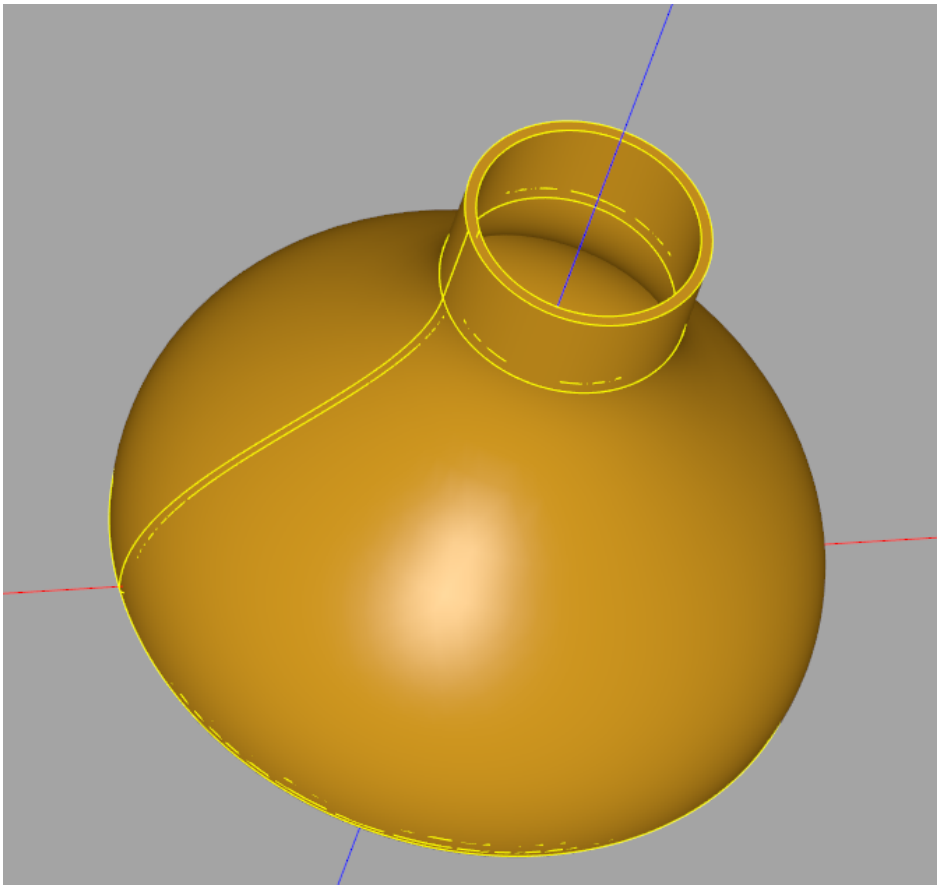


Figure 16: Upper-part after applying shell function with distZ=0.5, distX=0.5

Below, we find the results for other values of the parameters distZ and distX.

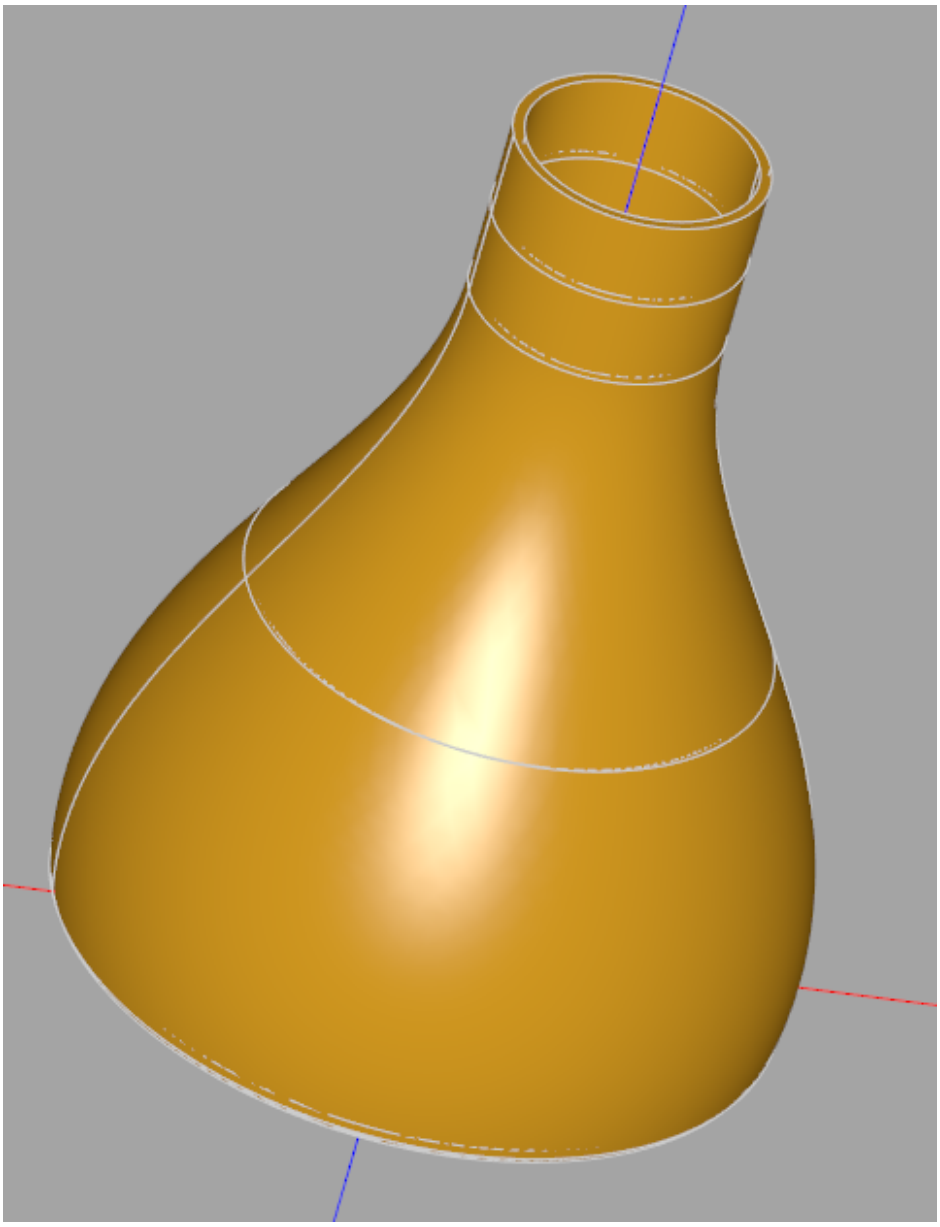


Figure 17: Upper-part after applying shell function with distZ=0.8, distX=0.5

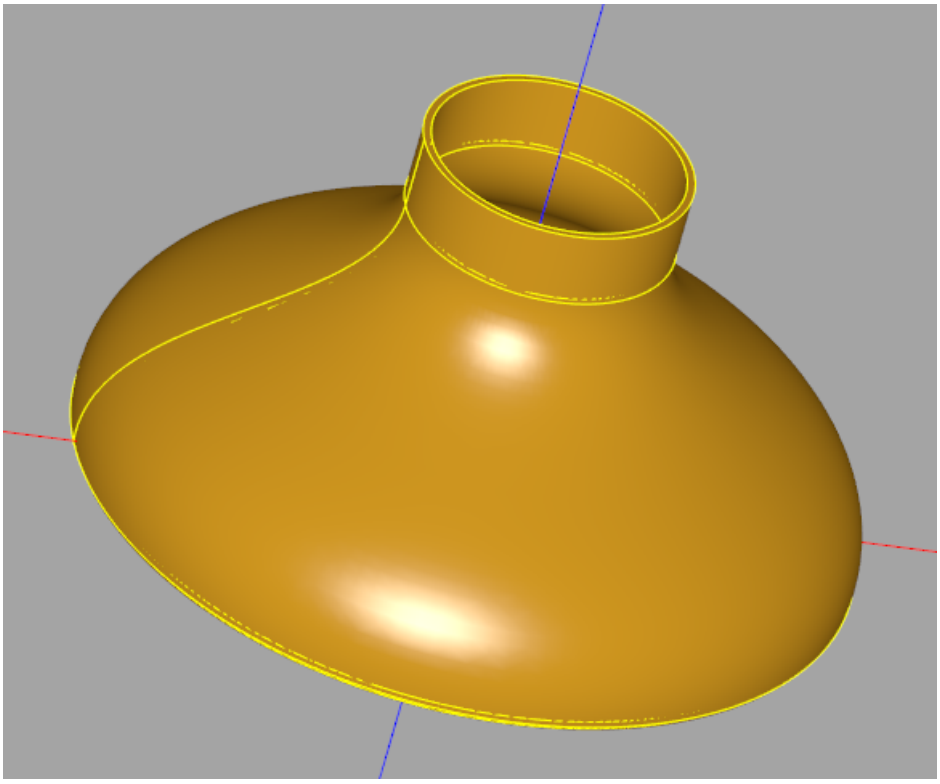


Figure 18: Upper-part after applying shell function with distZ=0.5, distX=0.8

In line 20 appears the translate function which is for assembly. It will be discussed in detail later.

- **Bottom-part**

  The bottom-part of the bottle is made of 5 blocks, starting from the bottom to the top.

  Part 1

  A workplane is defined first in line 23, which is XY this time. On this plan I build a box using a box function which takes the variables boxlengh, boxwidth, and boxheight.

  The curvy feature of the box's edges along the Z axis is made through the class function fillet which takes as a parameter the value boxlengh//2 which defines the radius of "the curvy box" (fillet (0) makes no changes).

Part 2

In line 25, I made an extruded circle with a radius = bandwidth / 2. s.faces(“>Z”) which means that the circle is built on the upper-face (along the Z axis) of the box (Part1). In this step, workplane (centerOption = ”CenterOfMass”) defines a new workplane set to the upper-face of the box (Part1) and means that the circle will have as center the center of mass of the upper face of the box. The next step is extruding the circle by a distance = bandlengh + 0.3 to make a cylindre, where 0.3 is a correction value: I had to add it because I noticed that both bands don’t have the same height even by using the same bandlengh value for both of them.

Part 3

This part is a box as well, and constructed the same way as the Part1. The difference is that it’s built on the circle (Part2) by defining a new workplane as it’s done in the part before.

0.6 is added to the boxheight, because the box in the middle is supposed to be longer than the others at the end with a length of 0.6 more.

Part 4

This part is made the same way as the circle (Part2), it is built on the box (Part3).

Part 5

This is the ending box: made the same way as the starting box (Part1), with exactly the same proportions.

Curvy feature of the bottle’s bottom and shell

To add the curvy feature to the final shape, I used fillet function with a value of 0.5 (fillet (0) keeps the bottom rectangular). And faces (“<Z”) means that the feature is added at the bottom of the bottle oriented along the Z axis from the start.

Here is the result after making all the parts and the features mentioned before.

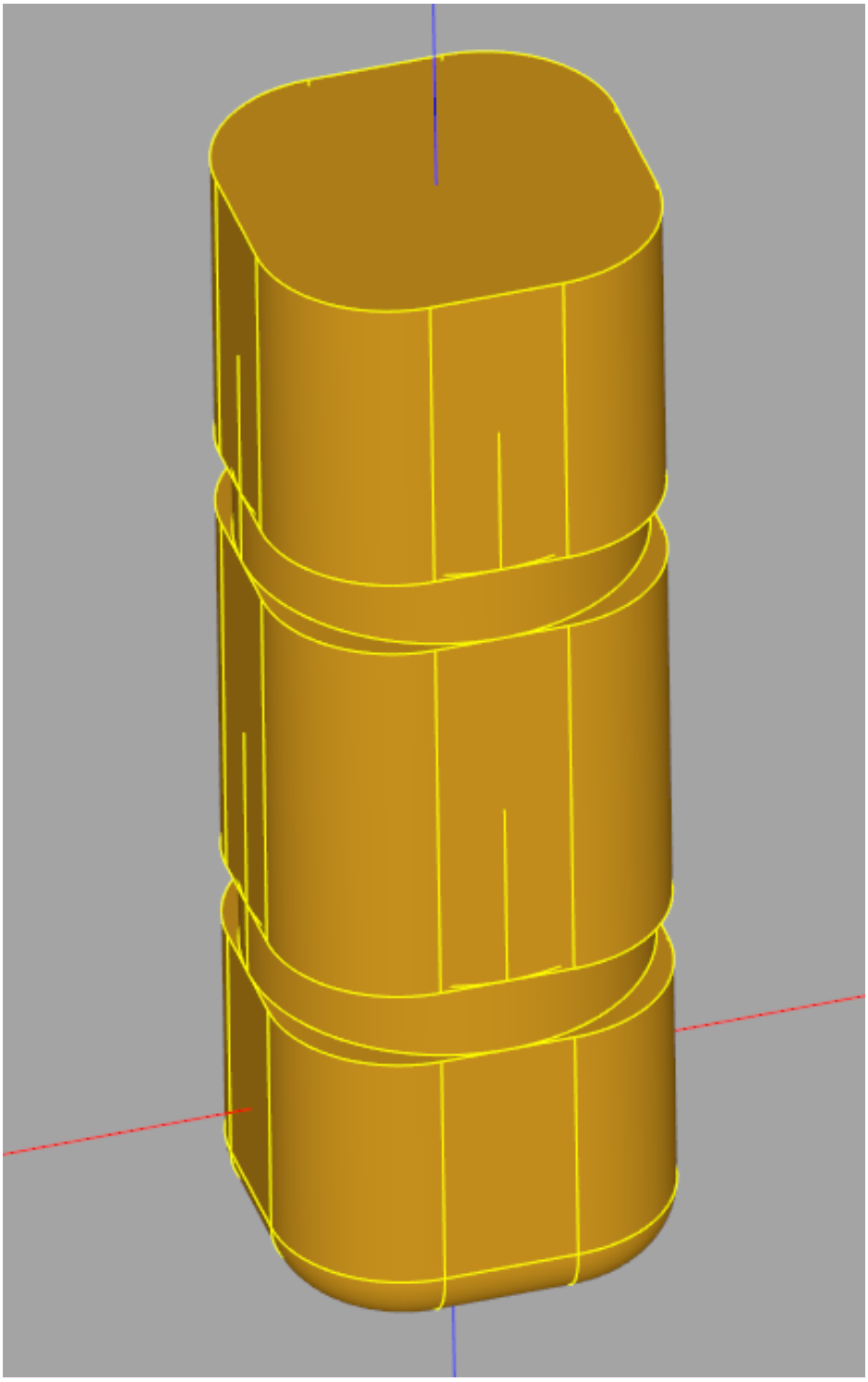


Figure 19: Bottom-part with bandlengh=1.8, bandwidth=3, boxlengh=3, boxwidth=3, boxheight=2.4 (view 1)

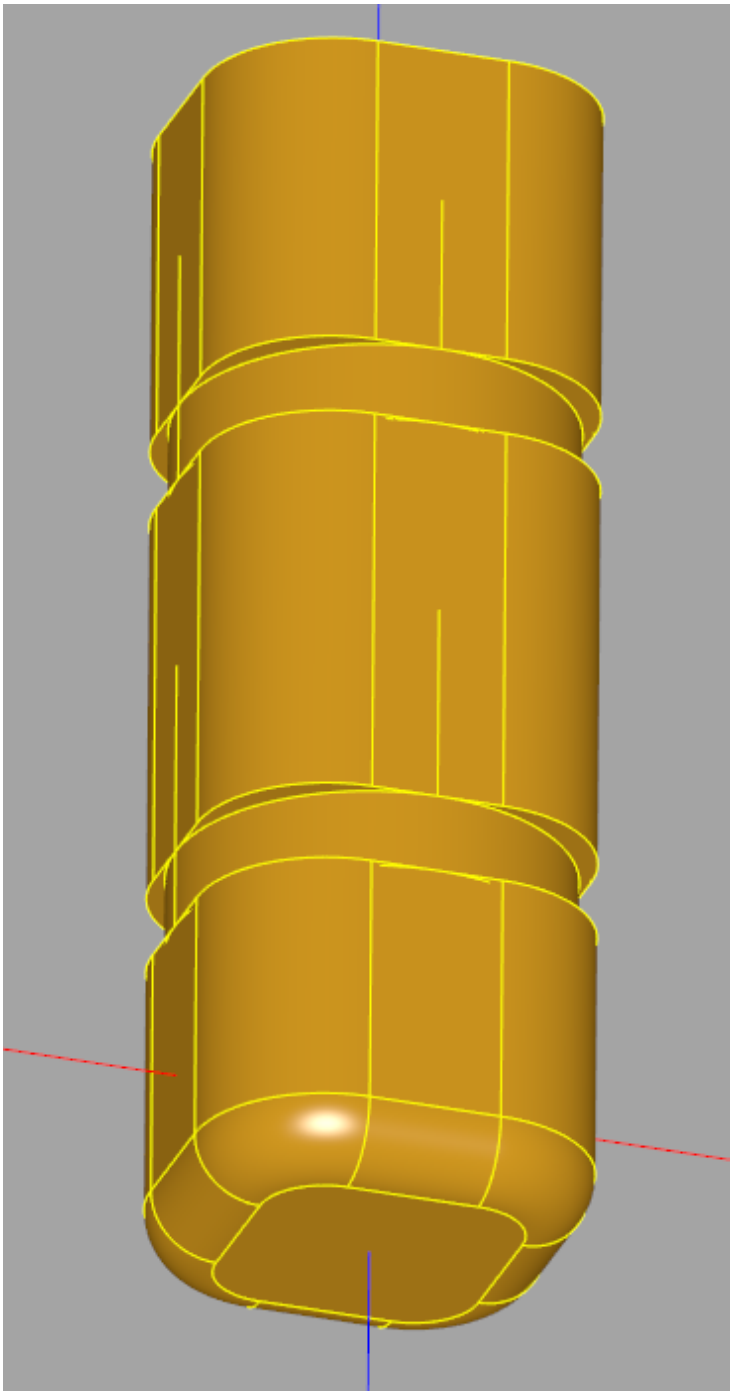


Figure 19: Bottom-part with bandlengh=1.8, bandwidth=3, boxlengh=3, boxwidth=3, boxheight=2.4 (view 2)

After applying the same shell function explained for the upper-part, we get the following result.

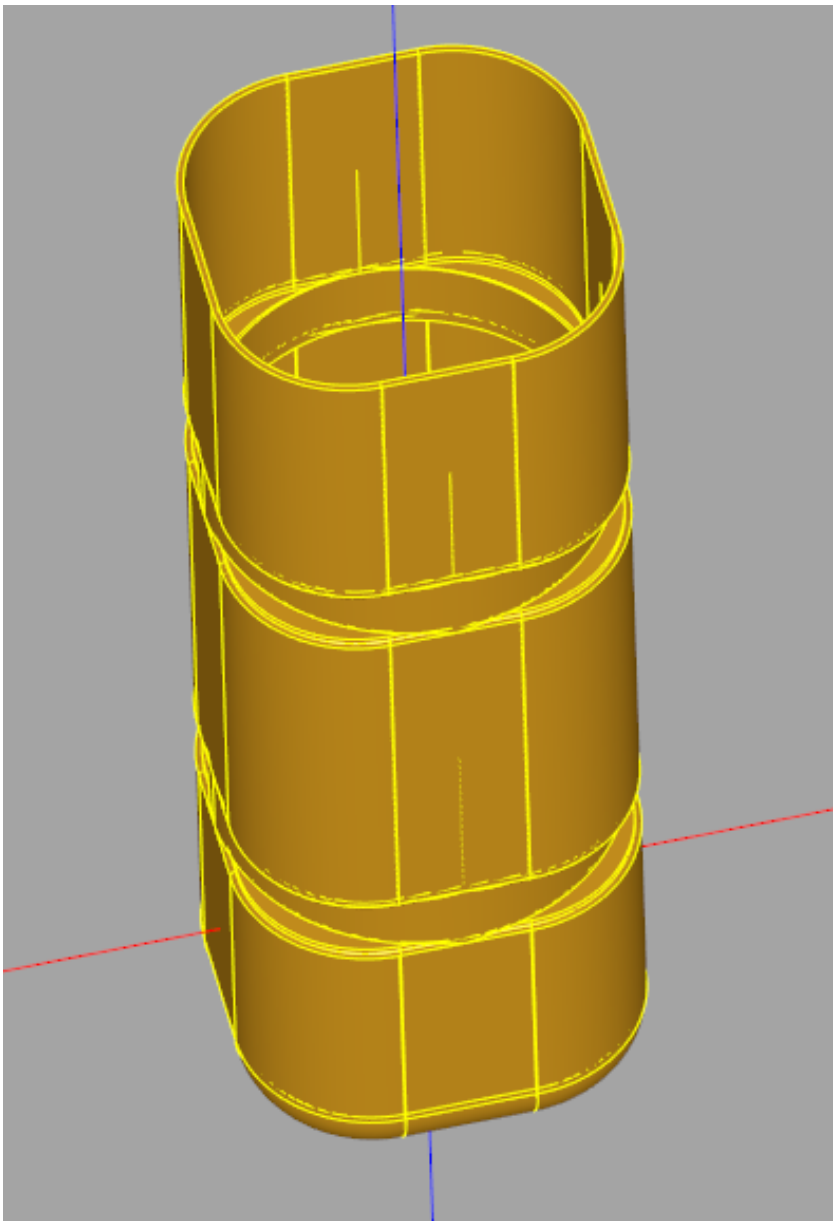


Figure 19: Bottom-part after applying shell function with bandlengh=1.8, bandwidth=3, boxlengh=3, boxwidth=3, boxheight=2.4

Translate function, used in line 20 translates the upper-part following the Z axis with a distance = the height of all the bottom-part. As a result the upper-part and the bottom-part are assembled and gives the result in Figure 11, Figure 12 and Figure 13 for the values: bandlengh=1.7, bandwidth=2.5, boxlengh=2.5, boxwidth=2.5, boxheight=1.9.

Chapter 5

Future Work

5.1 Generating Data

5.1.1 Synthetic Data based on parametric design

The code above should be improved by parameterizing the number of features as well as the curvature of the upper-part's edges. Another code should be added to construct the cap which will be assembled with the upper-part. It will be also better if the curvature of the upper-part is controlled through a parameter.

As a result, we will have a strong script for generating different kinds and shapes of bottles with many other features. Then in the end, we can create a fairly complete data set containing various bottles (images associated to their 3D shapes).

5.2 Training Pixel2Mesh

5.2.1 Training set

The data set mentioned in the section before will be used to train the model and to make it specialized in bottle reconstruction only. It is supposed to give better results than the pre-trained model.

Training will take place in AWS [10] servers since it offers high level facilities to train AI models.

5.2.2 Training the model using AWS ML

Typically, the process from conceptualizing to producing ML models is complex and time consuming. Training the model requires handling large amounts of data, choosing the best algorithm, managing compute capacity during training, and then deploying the model in a production environment. Amazon SageMaker alleviates this complexity by making it easier to build and deploy ML models. Once the right algorithms and frameworks are picked from the wide array of choices available, Amazon SageMaker manages all the underlying infrastructure to train the model at petabyte scale and deploy it to production. This will allow Digimind to train Pixel2Mesh on the generated data in a secure environment without any memory or complexity constraints.

Chapter 6

Conclusion

In conclusion, the internship was a useful experience. I have found out what my strengths and weaknesses are. I gained new knowledge and skills and met many new people. I achieved many of my learning goals, however for some the conditions did not permit to achieve them as I wanted.

At last this internship has given me new insights and motivation to pursue a career in Data Science & AI research. To prepare myself for my future career I can improve several things: I can work on my communication skills so that I am able to present and express myself more confidently. I could perform certain tasks in research better if I had more experience in the research methodologies and for that I tried to write this report in LaTeX in a way to make it the same way other thesis or research papers are made.

Acknowledgment

I cannot express enough thanks to the Digimind team for their continued support and encouragement: My supervisor **Dr. Katharina Eissing**, as well as the founder **Dr. Omar Fergani**, the great Software Engineer **Aimad Harilla** for our collaboration, and the manager **Duygu Duener**, who was in charge for my trip to Berlin. I offer my sincere appreciation for the learning opportunities provided within the team.

Finally, to my supportive parents and friends. Many thanks to **Léa** for her help in reviewing the entire report. Your encouragements are much appreciated and duly noted.

References

[1] ShapeNet Data Set: https://shapenet.org/

[2] Digimind Labs Web Site: https://digimindlabs.de/

[3] CY Tech Web Site: https://cytech.cyu.fr/en

[4] Pixel2Mesh Implementation: https://github.com/nywang16/Pixel2Mesh

[5] *Pixel2Mesh: Generating3D Mesh Models from Single RGB Images*

[6] BeautifulSoup Library: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[7] OpenCV Library: https://opencv.org/

[8] Pygalmesh Library: https://github.com/nschloe/pygalmesh

[9] CADQuery Library: https://cadquery.readthedocs.io/en/latest/

[10] Amazon Web Services - Machine Learning (AWS ML): https://aws.amazon.com/machine-learning/?nc1=h$_ls$