Experiment Name: Open AI & DALL-E - text to Image **Start at:3.1** **Finished at:3.13** **Operator: Zhang Mingyuan, Joyce**

Objective: using DALL-E to generate image based on text prompt

Method: using openai API

Procedure:

Step 1: set API key for the ChatGPT

```
import openai

# replace YOUR_API_KEY with your actual API key for the ChatGPT service
openai.api_key = "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"
```

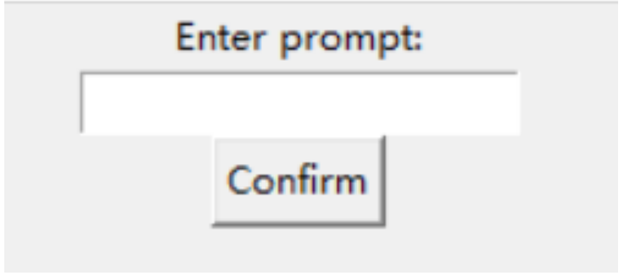
Step 2: use the DALL-E API to generate images based on input text:

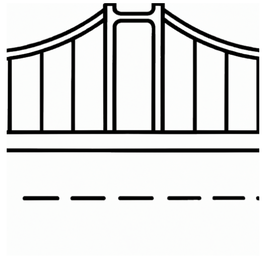
```
from requests.structures import CaseInsensitiveDict
import json

QUERY_URL = "https://api.openai.com/v1/images/generations"

def generate_image(prompt):
    headers = CaseInsensitiveDict()
    headers["Content-Type"] = "application/json"
    headers["Authorization"] = f"Bearer {openai.api_key}"

    model = "image-alpha-001"
    data = """
    {
        """
    data += f'"model": "{model}",'
    data += f'"prompt": "{prompt}",'
    data += """
        "num_images":1,
        "size":"512x512",
        "response_format":"url"
    }
    """
    resp = requests.post(QUERY_URL, headers=headers, data=data)

    if resp.status_code != 200:
        raise ValueError("Failed to generate image")

    response_text = json.loads(resp.text)
    return response_text['data'][0]['url']
```

Step 3: Call the generate_image function with a prompt, like "apple", and it will return a URL to an image of an apple generated by the DALL-E model. Then display the image in Python using a library like Pillow:

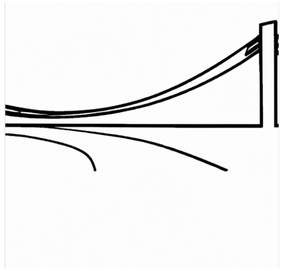
```
from PIL import Image
import requests

url = generate_image("a black line representing a bridge in white background")
img = Image.open(requests.get(url, stream=True).raw)
img.show()
```

Outcome:





“bridge”



“bridge”

Challenges:

Not a single line

Experiment Name: Open AI & Dalle E - edit image **Start at:3.12** **Finished at:3.20** **Operator: Zhang Mingyuan, Joyce**

Objective: using DALL-E to generate image based on text prompt

Method: using openai API

Procedure:

Step 1: set API key for the ChatGPT

Step 2: set the original image(need to be square)

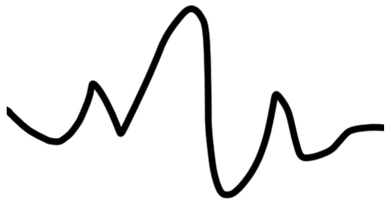
Step 3: edit the original image to generate a mask(alpha=0)

Notice: the transparent areas of the mask indicate where the image should be edited, and the prompt should describe the full new image, not just the erased area.

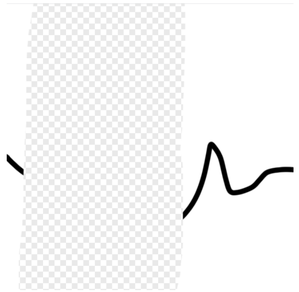
Step 4: use the DALL-E API to generate images based on the original image, mask and text prompt.

Outcome:

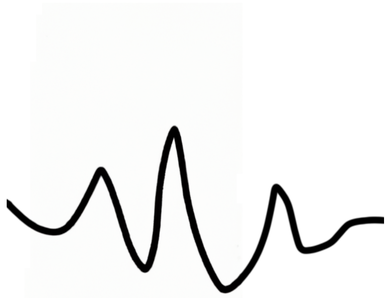
Original



Mask：



Generated



Prompt： “sharp”

Challenges: Finding the proper prompt to generate one single line

Experiment Name: ControlNet **Start at:3.20** **Finished at:3.24** **Operator: Zhang Mingyuan, Joyce**

Objective: using ControlNet to generate image based on image

Method: using ControNet API

Procedure:

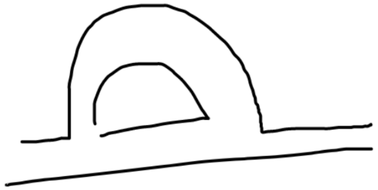
Step 1: sets the Replicate API token in the "REPLICATE_API_TOKEN" environment variable

Step 2: calls the replicate.run() function to run the image generation model.

Step 3: generate the image based on input image and prompt

Step 4: output the image

Outcome:



user input

generated

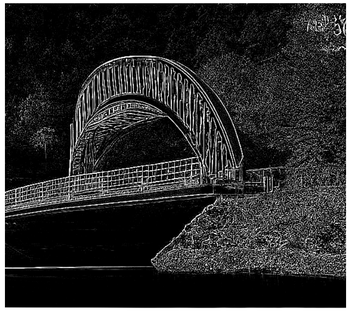


image processing

Challenges:

Need robotic arm trajectory planning

Experiment Name: Using brush to edit image **Start at:3.24** **Finished at:4.2** **Operator: Zhang Mingyuan, Joyce & Cheng Zhaolin, Sparrow**

Objective: allow user to edit an image using brush

Method: using ControNet API

Procedure:

1. Import the necessary libraries: tkinter, PIL (Python Imaging Library), openai, requests, BytesIO
2. Define a class called "App" that takes the "master" (i.e., root) as a parameter
3. Inside the class constructor (init), initialize the instance variables:
   - brush_size (set to 20)
   - color (set to 'white')
   - last_x, last_y (set to None)
   - Load the original image and convert it to RGBA format
   - Create a Draw object for the image
   - Get the width and height of the image
   - Create a canvas with the same dimensions as the image and pack it into the master
   - Create a PhotoImage object from the image and create an image item on the canvas using the PhotoImage
   - Bind the canvas to the paint and reset methods using the '<B1-Motion>' and '<ButtonRelease-1>' events, respectively
   - Create a button widget labeled "确认" that calls the save method when clicked and pack it into the master
4. Define a paint method that takes an event as a parameter
   - If last_x and last_y are not None, create a line on the canvas from the last_x, last_y coordinates to the current event.x, event.y coordinates, using the brush size and color
   - Draw a line on the image from the last_x, last_y coordinates to the current event.x, event.y coordinates, using a fill color of (0, 0, 0, 0) (i.e., transparent) and the brush size
   - Set last_x and last_y to the current event.x, event.y coordinates
5. Define a reset method that takes an event as a parameter
   - Set last_x and last_y to None
6. Define a save method
   - Save the image to a file named "E:/mask/mask.png"
   - Call the display_generated_image method
7. Define a display_generated_image method
   - Set the OpenAI API key
   - Call the openai.Image.create_edit method with the original and mask images, a prompt, number of images to generate, and desired size
   - Extract the URL of the generated image from the API response
   - Use requests library to download the image data from the URL
   - Save the image data to a file named "E:/generated/generated.png"
   - Open the file and display the generated image in a window
8. Create a tkinter root window and an instance of the App class using the root as a parameter
9. Call the mainloop method of the root window to run the GUI application.

Outcome:

Challenges:

The size of the brush and the size of the edited area would affect the effect of the DALL-E generation

Experiment Name: Capture image by camera **Start at: 3.1** **Finished at: 3.2** **Operator: Cheng Zhaolin, Sparrow**

Objective: Camera and the line on the foam

Method: python

Procedure:

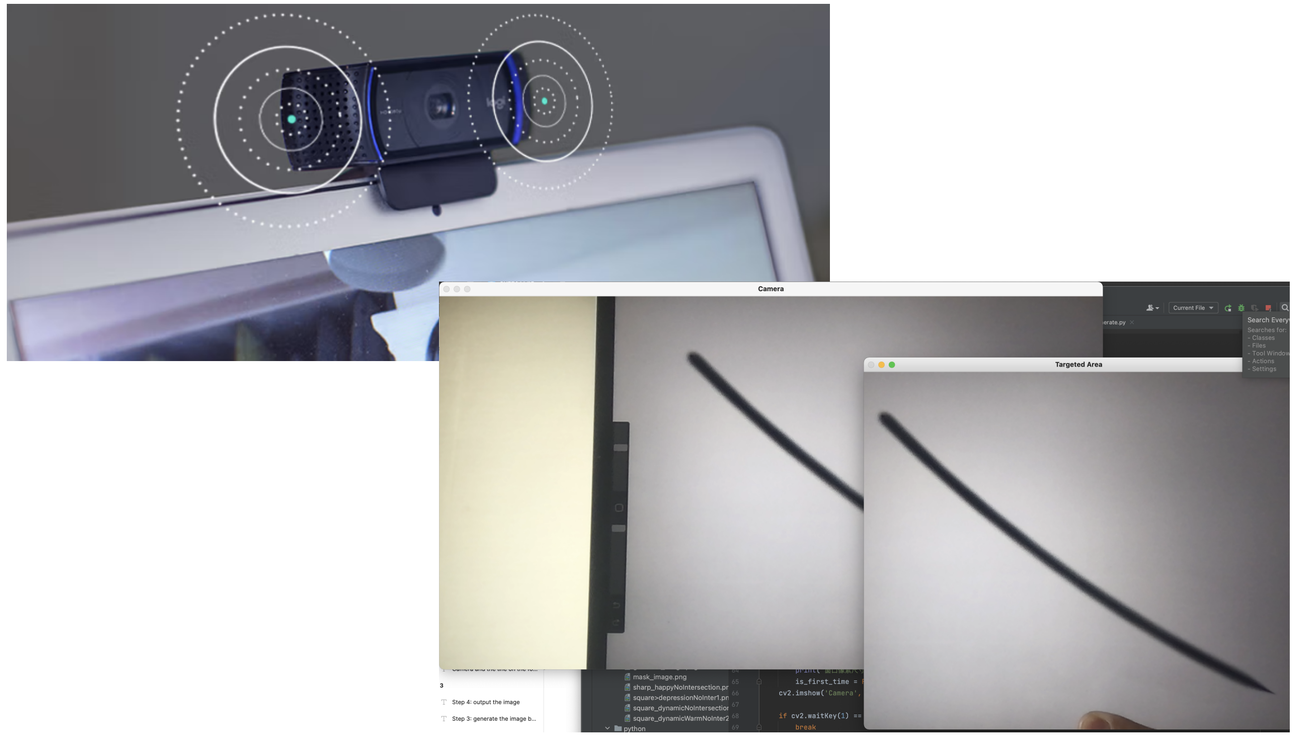
Step 1: Obtain the video stream

```python
import cv2
import os
save_dir = "../../output/"
if not os.path.exists(save_dir):
    os.makedirs(save_dir)
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Cannot open camera")
    exit()
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
print("Resolution: {}x{}".format(width, height))
cv2.namedWindow('Camera', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Camera', width, height)
is_first_time = True

while True:
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
```

Step 2: limit the working area in the camera and capture by ‘enter’ key

```python
    x, y, w, h = 450, 30, 1000, 1000
    roi = frame[y:y+h, x:x+w]
    cv2.imshow('Targeted Area', roi)
    if is_first_time:
        x, y, width, height = cv2.getWindowImageRect("Targeted Area")
        print("窗口像素尺寸为：{} x {}".format(width, height))
        is_first_time = False
    cv2.imshow('Camera', frame)

    if cv2.waitKey(1) == 13: # 13 is the ASCII code for the Enter key
        break

cap.release()
cv2.destroyAllWindows()
cv2.imwrite(os.path.join(save_dir, "captured_image.png"), roi)
```

Outcome:



Challenges:

Camera devices are very important. Some devices need to be configured to adapt to computers, while others are not compatible with MacOS at all.

Experiment Name: Initial analyze images by OpenCV **Start at: 3.10** **Finished at: 3.15** **Operator: Cheng Zhaolin, Sparrow**

Objective: Images captured by camera

Method: Using functions in OpenCV library

Procedure:

Step 1: GaussianBlur() the image for better condition towards binary image.

```
# 对灰度图像进行高斯模糊
img = cv2.GaussianBlur(img, (15, 15), 0)
borderType = cv2.BORDER_CONSTANT
dst = cv2.copyMakeBorder(img, 20, 20, 20, 20, borderType, None, (255,255,255))
dst = cv2.GaussianBlur(dst, (5,5), 100)
dst = dst[20:-20, 20:-20]
```

Step 2: Enhance the contrast of image

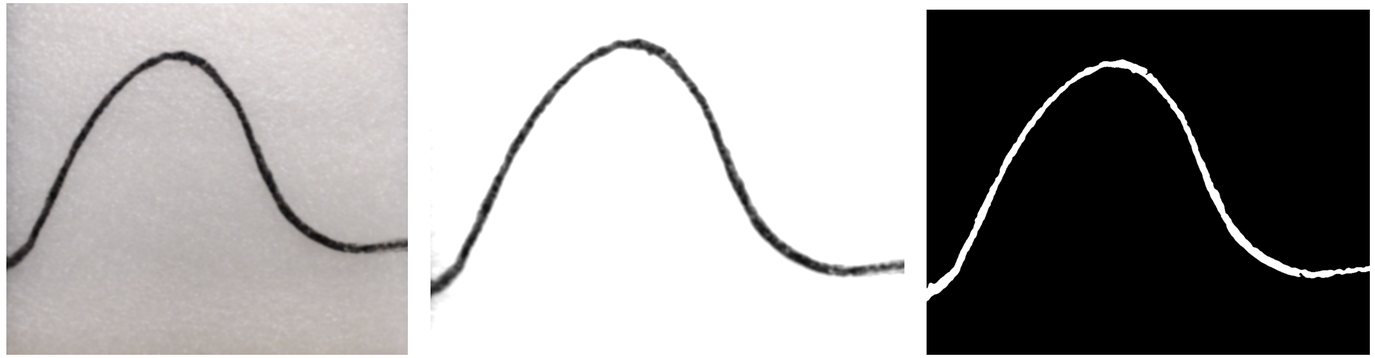
```
# define the contrast and brightness value
contrast = 1.6# Contrast control ( 0 to 127)
brightness = 1 # Brightness control (0-100)

# call addWeighted function. use beta = 0 to effectively only
# operate on one image
out = cv2.addWeighted(gray, contrast, gray, 0, brightness)

# Create a CLAHE object with desired parameters
clahe = cv2.createCLAHE(clipLimit=1.0, tileGridSize=(8,8))

# Apply CLAHE to the image
img_clahe = clahe.apply(out)
```

Step 3: Build binary image for skeletonization in the next phase

```
_, binaryImg = cv2.threshold(img_flipped, 127, 255, cv2.THRESH_BINARY_INV)
_, binaryImg2 = cv2.threshold(img_flipped, 127, 255, cv2.THRESH_BINARY)
cv2.imshow('bi-img', binaryImg)
cv2.waitKey(3000)
cv2.destroyAllWindows()
```

Outcome:

Gaussian Blur



Challenges:

It involves many functions and parameters. In real working environments, if the ambient light cannot be maintained, parameters also need to be modified accordingly to adapt to the environment

Experiment Name: Initial analyze images by OpenCV **Start at: 3.15** **Finished at: 3.25** **Operator: Cheng Zhaolin, Sparrow**

Objective: binary image

Method: Skeletonization / KNN

Procedure:

Step 1: Skeletonize the image by cv2.ximgproc.thinning(binaryImage) and find every coordinates of all non-zero pixels in the skeleton.

```
skeletonImg = cv2.ximgproc.thinning(binaryImg)

try:
    coor3ds = cv2.findNonZero(skeletonImg)  # Find the coordinates of all non-zero pixels in the skeleton
    # assume points is a 3-dimensional numpy array
    coords = np.squeeze(coor3ds, axis=1)

    num_points = 120
    indices = np.linspace(0, len(coords) - 1, num_points, dtype=np.uint32)
    points = coords[indices]
    blank_image = np.zeros_like(skeletonImg)
    for point in points:
        x, y = point
        cv2.circle(blank_image, (x, y), 1, (255, 255, 0), -1)

    for i in range(len(points)):
        x, y = points[i]
        print("Point", i + 1, ":", "(", x, ",", y, ")")
```

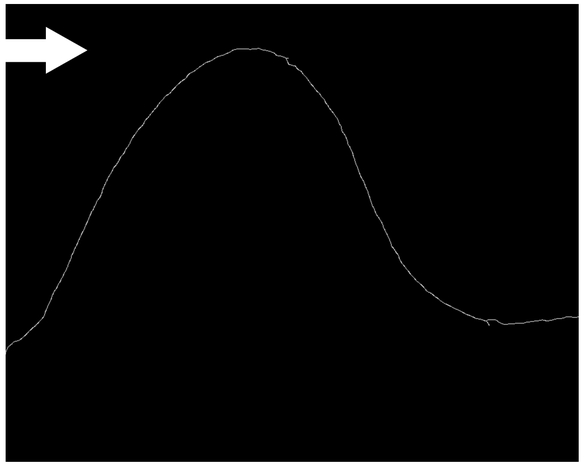
Step 2: And use KNN to order every points form the left to right.

```
left_endpoint = points[np.argmin(points[:, 0], axis=0), :]

ordered_points = []
ordered_points.append(left_endpoint)
# 从左端点开始，沿着曲线有序地遍历每个点
current_point = left_endpoint

points_copied = points
len_points = len(points)
####### 怎么去除重复点
while len(ordered_points) < len_points:
    # 如何确保每一个点不重复的遍历，这个很重要
    # 统一不同维度的数组 eg.points == [current_point]
    index_to_remove = np.where(np.all(points == [current_point], axis=1))
    points = np.delete(points, index_to_remove, axis=0)

    # 计算当前点到所有其他点的距离，除了当前点相同坐标的点外。
    distances = distance.cdist(points, [current_point], 'euclidean')
    # distances[np.where(distances == 0)] = np.inf  # 设置为inf，避免选择自身

    # 找到最近的点并添加到有序列表中，引用最近点的x值
    nearest_point = points[np.argmin(distances), :]

    ordered_points.append(nearest_point)
    # 将新的最近点作为下一个起点
    current_point = nearest_point

for ordered_point in ordered_points:
    print(ordered_point)
```

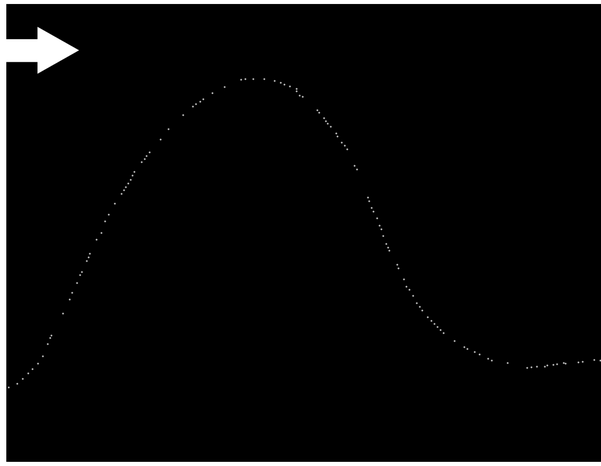
Outcome:

BinaryImage

Skeleton of the line



Draw circles on the coordinates of the points



Challenges:

This phase involves tons of parameters and strong logical structures are required. Main obstacles are understanding every lines of code and build the functional process better logically. Most problems encountering during coding can be solved by AI assistance, except this part. (CV(Computer vision) has been developed for many years, but still lacks technology and technicians.) Therefore, this section of all is the most challenging.

Experiment Name: Integration with xArm7 **Start at: 3.25** **Finished at: 4.5** **Operator: Cheng Zhaolin, Sparrow**

Objective: Coordinates of points, robot arm

Method: Spatial mapping

Procedure:

Step 1: Preset of xArm7

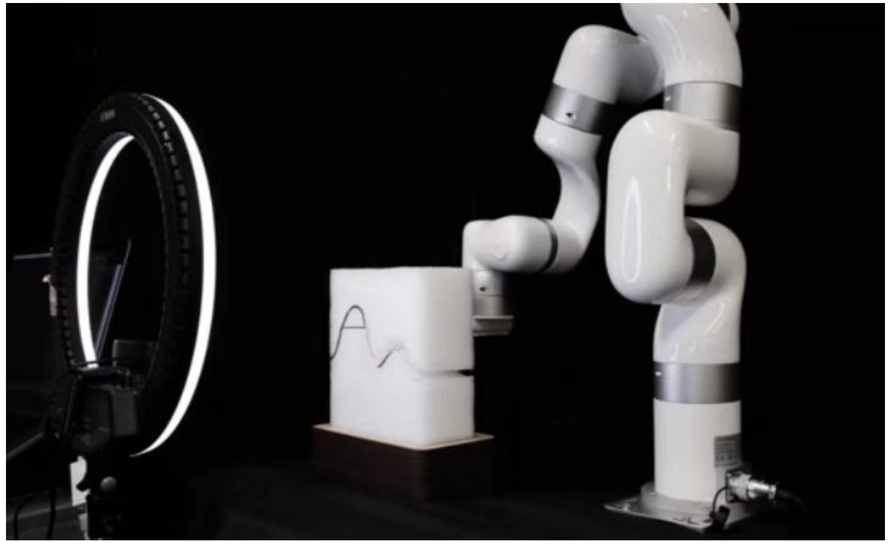
```
#!/usr/bin/env python3
# Software License Agreement (BSD License)
#
# Copyright (c) 2018, UFACTORY, Inc.
# All rights reserved.
#
# Author: Vinman <vinman.wen@gmail.com> <vinman.cub@gmail.com>

"""
Start xArm system
"""
if len(sys.argv) >= 2:
    ip = sys.argv[1]
else:
    try:
        from configparser import ConfigParser

        parser = ConfigParser()
        parser.read('../robot.conf')
        ip = parser.get('xArm', 'ip')
    except:
        ip = input('Please input the xArm ip address:')
        if not ip:
            print('input error, exit')
            sys.exit(1)
########################################################

arm = XArmAPI(ip, is_radian=True)
arm.motion_enable(enable=True)
arm.set_mode(0)
arm.set_state(state=0)

# arm.reset(wait=True)
```

Step 2: Calculate the ratio between two systems, including how robot arm move during working

```
k = 200 / 512  # preset

######################
# initial(退刀)
arm.set_position(x=196, y=-250, z=120.5, roll=-180, pitch=0, yaw=0, speed=100, is_radian=False, wait=True)
print(arm.get_position(), arm.get_position(is_radian=False))

#  > 推刀 > working
first_z = ordered_points[0][1]
zArm_first = 265 - k * first_z
arm.set_position(x=196, y=0, z=zArm_first, roll=-180, pitch=0, yaw=0, speed=70, is_radian=False, wait=True)
print(arm.get_position(), arm.get_position(is_radian=False))

# questions
# 1. 先计算好数集?
# roi_size / xarm_size

path_feedback = np.zeros_like(skeletonImg)  ####

for i in range(len(ordered_points)):
    x_img, y_img = ordered_points[i]
    cv2.circle(path_feedback, (x_img, y_img), 2, (255, 255, 0), -1) ####
    cv2.imshow('Coords to Path', path_feedback)
    cv2.waitKey(1)  # Wait for 1 millisecond to update the window
    print("Coordinate", i + 1, ":", "(", x_img, ",", y_img, ")")

    xArm = k * x_img + 216
    # + offset_xarm
    zArm = 265 - k * y_img
    # + offset_zarm
    # 每次循环都需要计算，是否可以降低效率
    arm.set_position(x=xArm, y=0, z=zArm, roll=-180, pitch=0, yaw=0, speed=8, is_radian=False, wait=True)
    print(arm.get_position(), arm.get_position(is_radian=False))

# > 退 > initial(退刀)
last_x = ordered_points[-1][0]
last_z = ordered_points[-1][1]
xArm_last = k * x_img + 216
zArm_last = 265 - k * y_img
arm.set_position(x=xArm_last + 50, y=0, z=zArm_last + 50, roll=-180, pitch=0, yaw=0, speed=5, is_radian=False, wait=True)
print(arm.get_position(), arm.get_position(is_radian=False))
```

Outcome:

Image data mapping onto the xArm7



Challenges:

Mapping needs to consider spatial relationship between image axis and xArm axis. Also the spatial coordinate of foams is important. We should weigh the pros and cons, and consider how to minimize work scope and risks during user experience.(eg. how to avoid the heating bars get in touch with user when system is uncontrollable.)

Experiment Name: Cutting speed of the robotic arm

Start at: 4.01

Finished at: 4.05

Operator: SHIU Sheung Ting Ramona, Liang Jiacheng Robin

Objective: Find out the most suitable moving speed when cutting foam

Method: Trial and errors

Procedure:

Step 1: Make a 3d print piece to hold the hot wire

Step 2: Connect the piece to the robotic arm

Step 3: Cut some foam pieces to try the cutting

Step 4: Try the speed

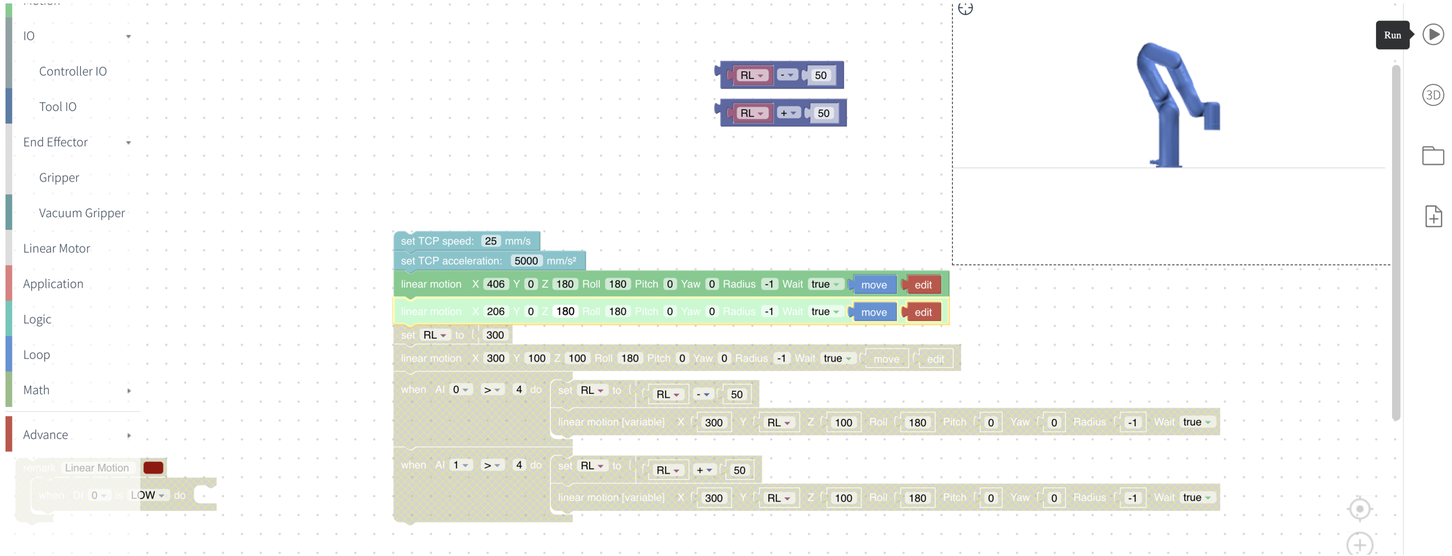
- Tried 200 mm/s at first, it cuts too fast that it is very difficult to hold the foam tight and stable, so the speed to cut is reduced for several attempts
- 200/150/100/50 mm/s is still too fast
- So then the speed is reduced to 25 mm/s, it gives a tidy cut with a not too fast speed
- bended

Outcome:

Speed 50/100/200 mm/s is too fast
We cannot hold the foam tight while moving this fast

Speed 25mm/s below is the best with a tidy cut

At a later stage the hot wire bended
To be safe, speed 5 mm/s is the best working speed to cut the foam



Challenges:

This phase involves tons of parameters and strong logical structures are required. Main obstacles are understanding every lines of code and build the functional process better logically. Most problems encountering during coding can be solved by AI assistance, except this part. (CV(Computer vision) has been developed for many years, but still lacks technology and technicians.) Therefore, this section of all is the most challenging.