Project Report

Course Name (WSQ)	***AI Engineer***
Product Name (Marketing & Sales)	***CCRS***
Module Name (WSQ)	**WSQ-Applied Python Programming (SF)**
Product Name (Marketing & Sales)	**WSQ-Applied Python Programming (SF)**

Student name	**Assessor name**	
Zhang Qiao		
Date issued	**Completion date**	**Submitted on**
2023 Aug 24	2023 Aug 24	2023 Aug 24

Project title	Real-time Object Detection and Tracking App

Learner declaration
I certify that the work submitted for this assignment is my own and research sources are fully acknowledged. Student signature: Zhang Qiao Date: 2023 Aug 24

Project Overview: Describe the Project along with Project Outcomes (Explain the Project in your own words in 15 – 20 lines)

The project is using opencv to build a GUI with function of loading images, resize and colour manipulation . Additional function such as different filters, object detection and tracking, and other image transformation functions have also been implemented.

The project is representing the skills of handling images with Python programming.

The outcome is a software with GUI built by python and demonstrating all the learned computer vision techniques.

1. **Project Technical Environment:** (Describe the Architecture with Tools used)

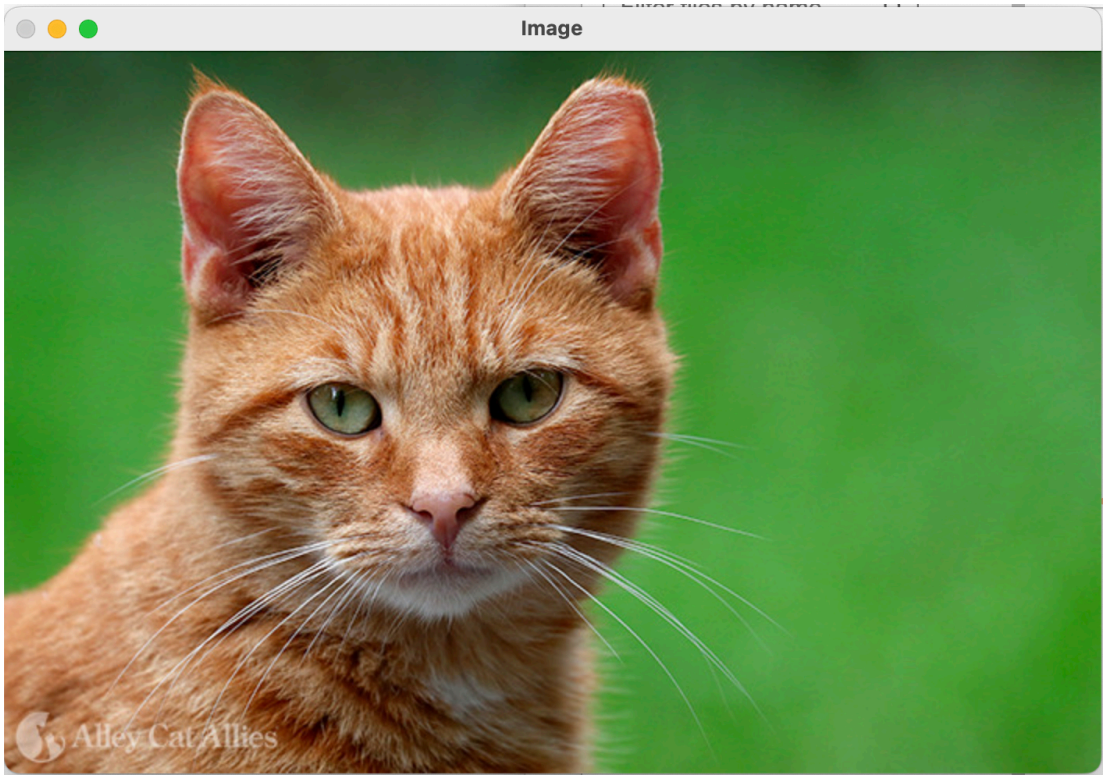
The project is mainly using opencv-python library operating on Jupyter notebook environment.

Other supportive libraries including pillow, wxpython (for GUI) and numpy have also been used.

2. **Design the Model:** (Explain the training model which you are designing using)

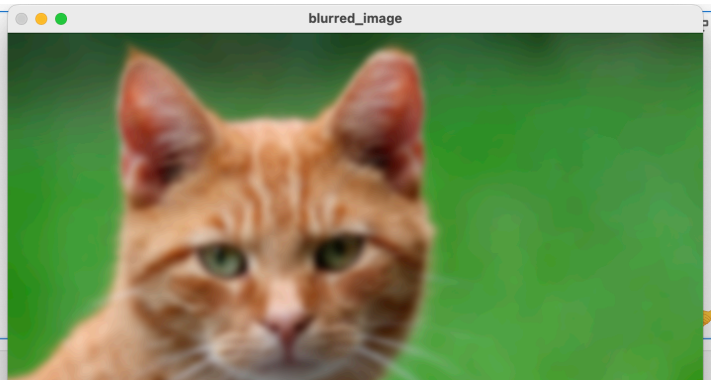
This project is loading yolov3.weights for object detection, there is no training model.

Activity 1 & 2:

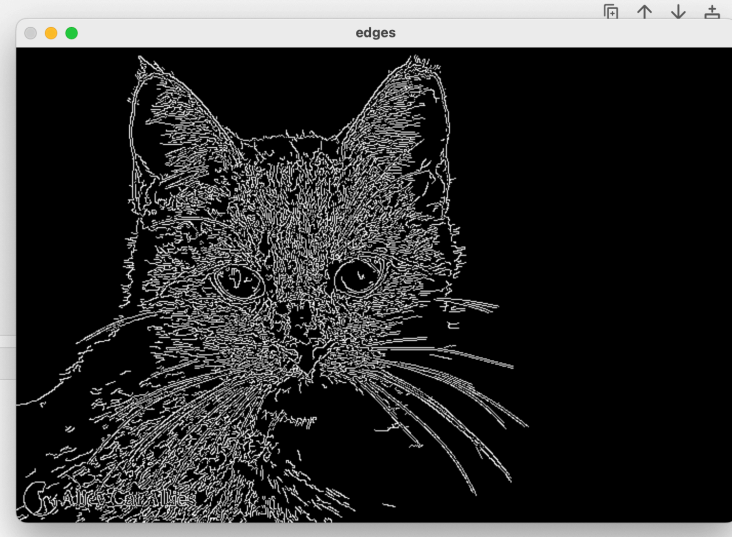



```
import cv2
img = cv2.imread('../images/cat.jpg')

cv2.imshow("Image",img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
image = cv2.imread('../images/cat.jpg')

cropped = image[0:300, 0:300]

gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#Gaussian Blur
blurred_image = cv2.GaussianBlur(image , (21,21) , 0)
#Canny
edges = cv2.Canny(image , 120 , 220)

#Color Manipulation - Invert color
inverted_image = 255 - image

show_img({'blurred_image':blurred_image})
```



```python
image = cv2.imread('../images/cat.jpg')

cropped = image[0:300, 0:300]

gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#Gaussian Blur
blurred_image = cv2.GaussianBlur(image , (21,21) , 0)
#Canny
edges = cv2.Canny(image , 120 , 220)

#Color Manipulation - Invert color
inverted_image = 255 - image

show_img({'edges':edges})
```

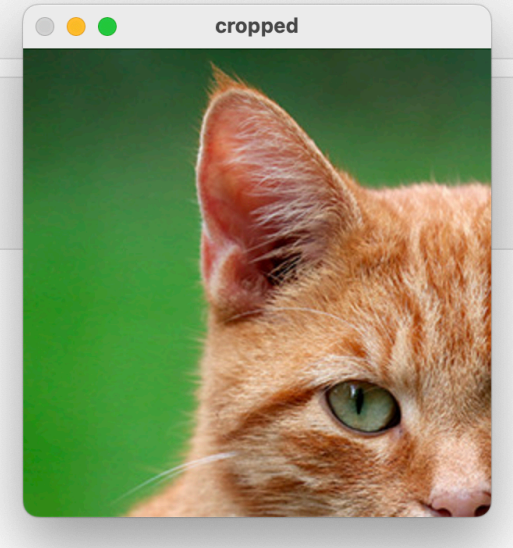


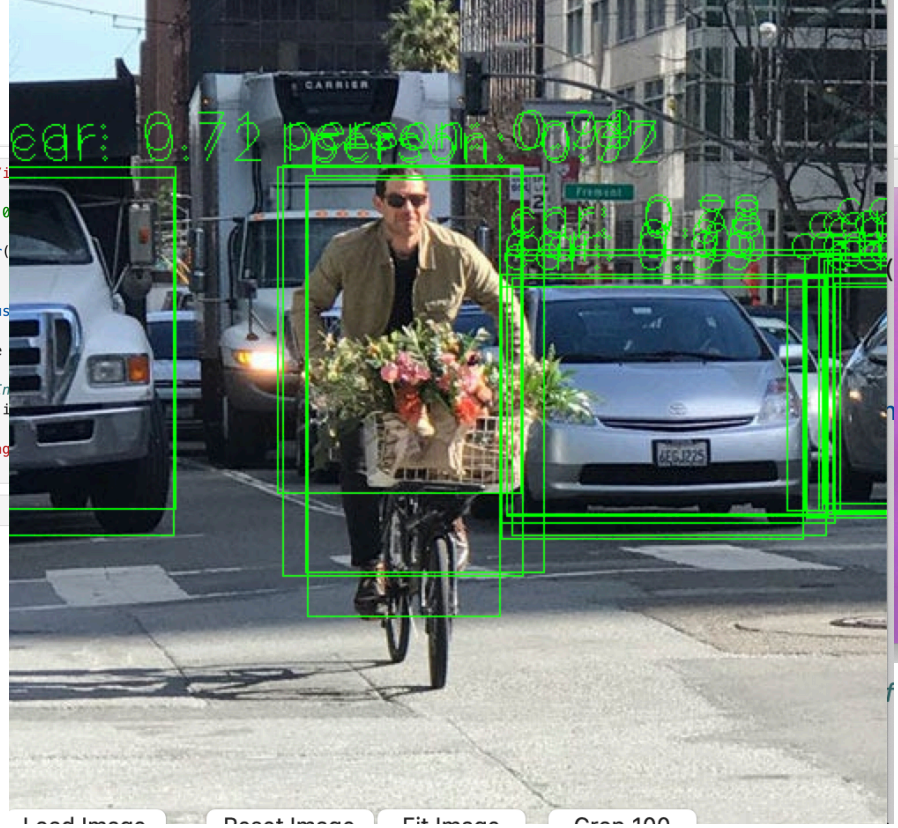

```
cv2.waitKey(1)
```

```python
image = cv2.imread('../images/cat.jpg')

cropped = image[0:300, 0:300]

gray_img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

show_img({'gray_img':gray_img})
```


```
    cv2.destroyAllWindows()
    cv2.waitKey(1)
```

```python
[*]: image = cv2.imread('../images/cat.jpg')

cropped = image[0:300, 0:300]
show_img({'cropped':cropped})
```

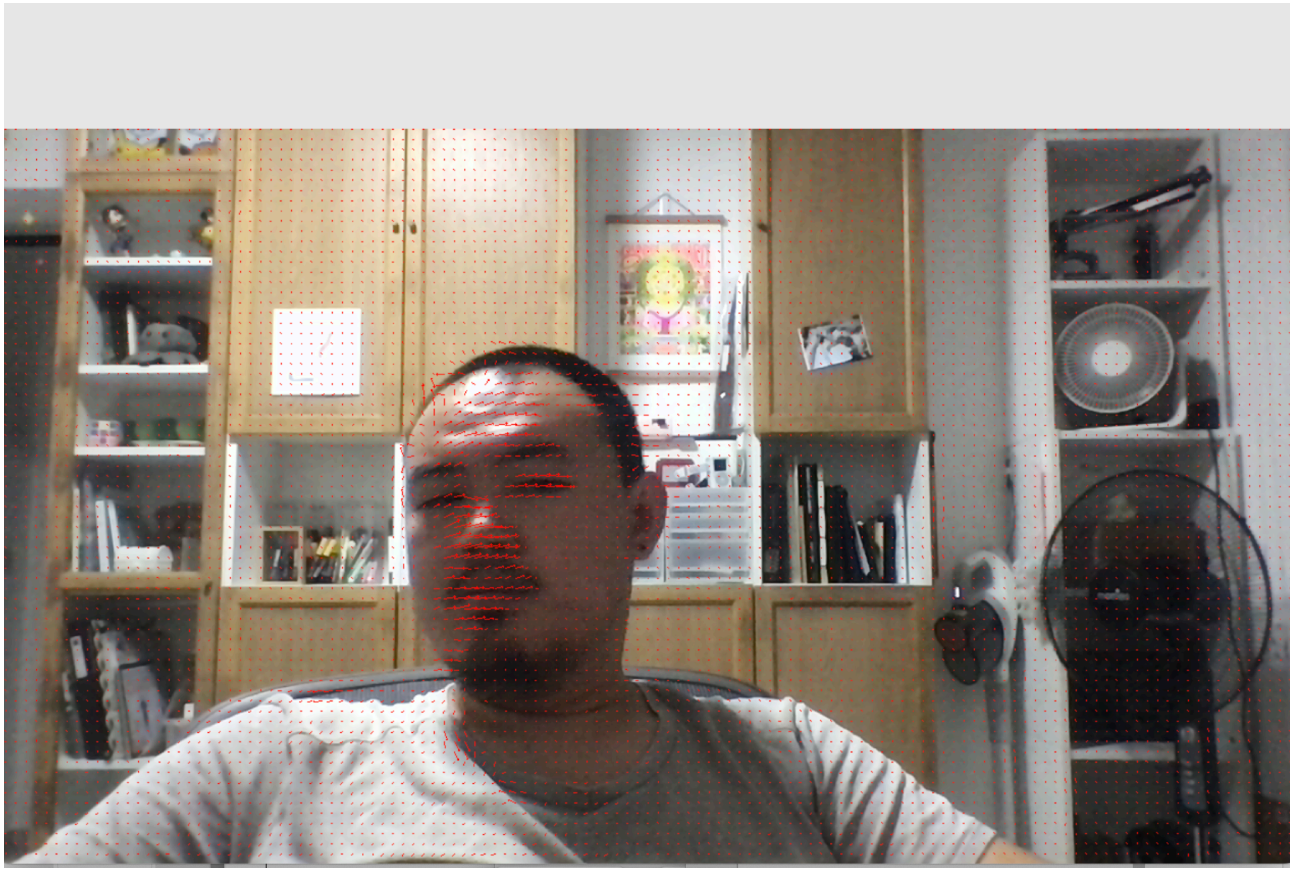




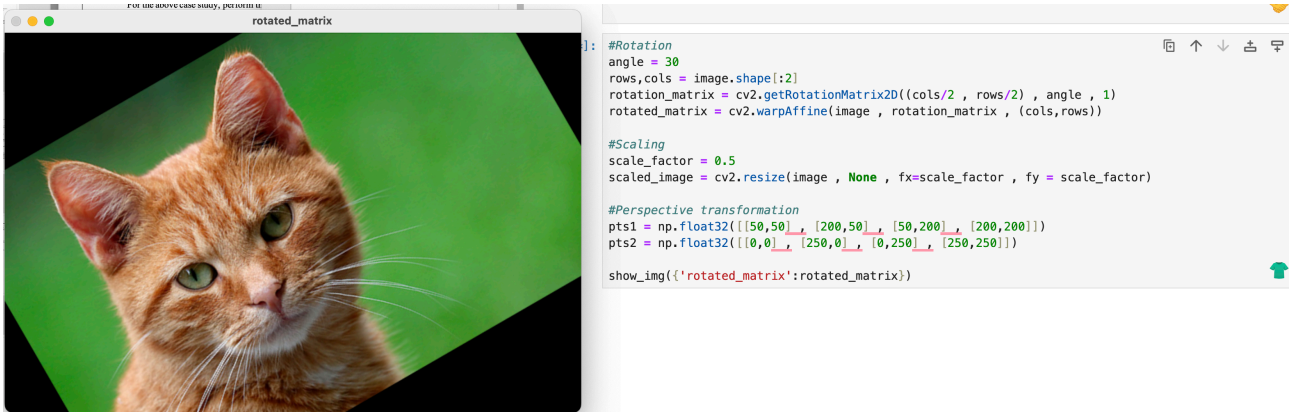
[1]:
```python
import cv2
import numpy as np

# Capture video feed
cap = cv2.VideoCapture(0)

# Read the frame
ret1, frame1 = cap.read()
prev_gray = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)

while cap.isOpened():
    ret, frame2 = cap.read()
    current_gray = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)

    # Calculate optical flow
    flow = cv2.calcOpticalFlowFarneback(prev_gray, current_gray, None, 0.5, 3, 15, 3, 5, 1.2, 0)

    # Draw motion vectors
    motion_image = np.copy(frame2)
    for y in range(0, motion_image.shape[0], 10):
        for x in range(0, motion_image.shape[1], 10):
            dx, dy = flow[y, x]
            cv2.arrowedLine(motion_image, (x, y), (int(x+dx), int(y+dy)), (0, 0, 255), 1)

    prev_gray = current_gray

    # cv2.imshow('Video', frame1)
    # cv2.imshow('Gray Video', prev_gray)
    cv2.imshow('Optical Flow', motion_image)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
cv2.waitKey(1)
```

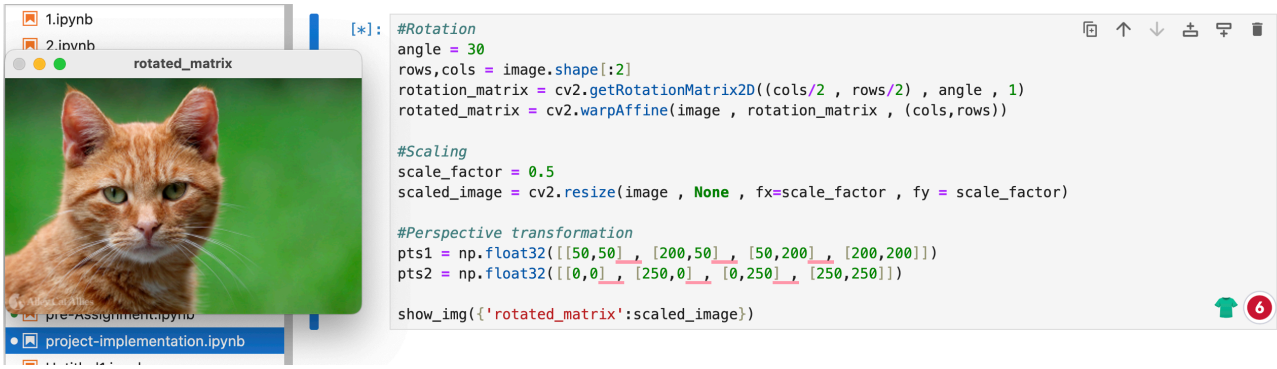
Activity 3 & 4

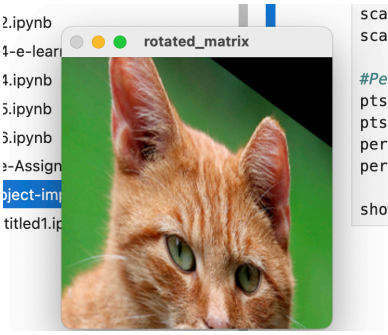
```
def OnDetect(self, event):
    # Prepare input image for YOLO model
    img = self.img_raw

    height, width = img.shape[:2]

    # img = img.astype(np.float32)

    blob = cv2.dnn.blobFromImage(img, 1/255, (416, 416), swapRB=True, crop=False)
    self.net.setInput(blob)

    # Get detection results
    outs = self.net.forward(self.net.getUnconnectedOutLayersNames())

    # Process detection results
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = scores.argmax()
            confidence = scores[class_id]
            if confidence > 0.7:  # Set a confidence threshold
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                # Draw bounding box and class label
                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 1)
                label = f'{self.classes[class_id]}: {confidence:.2f}'
                cv2.putText(img, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 1)

    self.ShowCV2Image(img)
```



Activity 5

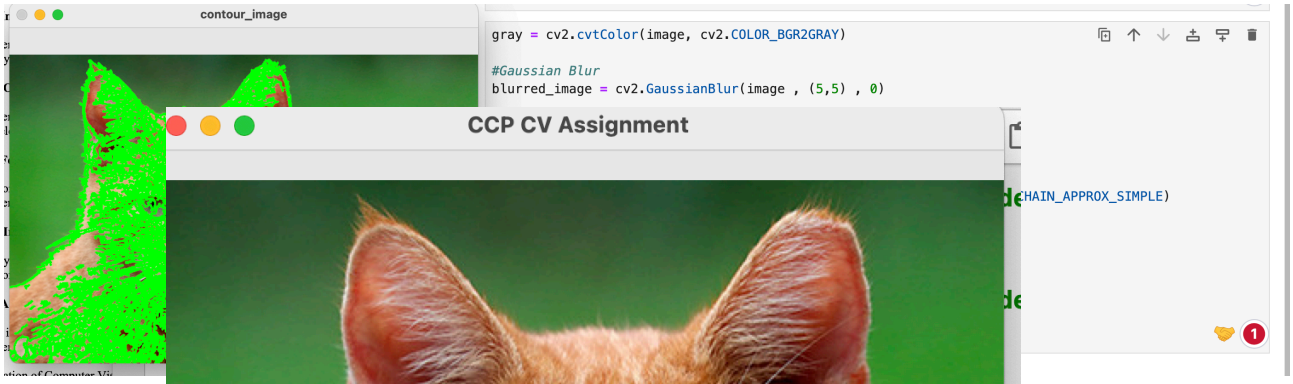

rotated_matrix
#Rotation
angle = 30
rows,cols = image.shape[:2]
rotation_matrix = cv2.getRotationMatrix2D((cols/2 , rows/2) , angle , 1)
rotated_matrix = cv2.warpAffine(image , rotation_matrix , (cols,rows))
#Scaling
scale_factor = 0.5
scaled_image = cv2.resize(image , None , fx=scale_factor , fy = scale_factor)
#Perspective transformation
pts1 = np.float32([[50,50] , [200,50] , [50,200] , [200,200]])
pts2 = np.float32([[0,0] , [250,0] , [0,250] , [250,250]])
show_img({'rotated_matrix':rotated_matrix})


1.ipynb
rotated_matrix
project-implementation.ipynb
[*]:
#Rotation
angle = 30
rows,cols = image.shape[:2]
rotation_matrix = cv2.getRotationMatrix2D((cols/2 , rows/2) , angle , 1)
rotated_matrix = cv2.warpAffine(image , rotation_matrix , (cols,rows))
#Scaling
scale_factor = 0.5
scaled_image = cv2.resize(image , None , fx=scale_factor , fy = scale_factor)
#Perspective transformation
pts1 = np.float32([[50,50] , [200,50] , [50,200] , [200,200]])
pts2 = np.float32([[0,0] , [250,0] , [0,250] , [250,250]])
show_img({'rotated_matrix':scaled_image})




```
scale_factor = 0.5
scaled_image = cv2.resize(image , None , fx=scale_factor , fy = scale_factor)

#Perspective transformation
pts1 = np.float32([[50,100] , [200,30] , [50,200] , [200,200]])
pts2 = np.float32([[0,0] , [100,0] , [0,100] , [100,150]])
perspective_matrix = cv2.getPerspectiveTransform(pts1,pts2)
perspective_transformed_image = cv2.warpPerspective(image, perspective_matrix , (250,250))

show_img({'rotated_matrix':perspective_transformed_image})
```

Activity 6




```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#Gaussian Blur
blurred_image = cv2.GaussianBlur(image , (5,5) , 0)
```

[1]:
```
import wx
import cv2
from PIL import Image
import imutils
import numpy as np
import time
```

[2]:
```
class Frame(wx.Frame):
    def __init__(self, parent, ID=-1, title="CCP CV Assignment"):
        super().__init__(parent, ID, title)

        # Main window holds all the content
        self.main_window = wx.FlexGridSizer(rows=2, cols=1, hgap=10, vgap=10)

        # Contraollers holds all buttons and controls
        self.controllers = wx.FlexGridSizer(rows=4, cols=4, hgap=2, vgap=2)

        # Image Viewer
        self.img_viewer = wx.StaticBitmap(self)

        # Add image viewer and controllers to main window
        self.main_window.Add(self.img_viewer,0,0)
        self.main_window.Add(self.controllers,0,0)

        self.img_viewer.SetMinSize((500,500))
        self.controllers.SetMinSize((500,200))

        # Controller Buttons
        btn_load_img = wx.Button(self, label='Load Image')
        btn_load_img.Bind(wx.EVT_BUTTON, self.OnBrowse)

        btn_reset = wx.Button(self, label='Reset Image')
        btn_reset.Bind(wx.EVT_BUTTON, self.OnReset)

        btn_resize = wx.Button(self, label='Fit Image')
        btn_resize.Bind(wx.EVT_BUTTON,
```