ELL715: Assignment 3

- Aditya Singh (2020EE10461)
- Harsh Swaika (2021EE11052)
- Ananya M (2020MT10787)
- Pramod Prasad (2022BSZ8403)
- Sarthak Srivastava (2020EE10550)

1. Design the **Interactive Interface**: Devise an intuitive user interface that allows users to upload images and interactively mark areas of their choice, they may assume some object is present. Bonus: Provide options for drawing, outlining, or applying brush strokes.

2. Implement **Background Subtraction**: Develop an algorithm that intelligently subtracts the background of the image based on user markings.

3. Innovative **Object Highlighting**: Design a creative way to highlight the segmented objects within the image.

4. Real-time **Object Detection**: Integrate real-time object detection using pre-trained models or custom algorithms. As users interact with the image, the system should instantly identify objects based on their markings.

In [1]:
```
!pip3 list
```

```
Package       Version
------------- --------
numpy         1.25.2
opencv-python 4.6.0.66
pip           23.2.1
setuptools    58.0.4
```

In [2]:
```
!pip3 install --upgrade pip
```

```
Requirement already satisfied: pip in ./env/lib/python3.9/site-packages (2
3.2.1)
```

In [3]:
```
!pip3 install opencv-python==4.6.0.66
```

```
Requirement already satisfied: opencv-python==4.6.0.66 in ./env/lib/python
3.9/site-packages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in ./env/lib/python3.9/site-p
ackages (from opencv-python==4.6.0.66) (1.25.2)
```

In [4]:
```
import cv2 as cv
import numpy as np
```

In [5]:
```
cv.__version__
```

Out[5]:
```
'4.5.5'
```

In [6]:
```
print(cv.EVENT_MOUSEMOVE)
print(cv.EVENT_LBUTTONDOWN)
print(cv.EVENT_LBUTTONUP)
```

```
0
1
4
```

In [7]:
```
!wget --no-clobber -O "coco.names" "https://opencv-tutorial.readthedocs.i
```

```
File ‘coco.names’ already there; not retrieving.
```

In [8]:
```
!wget --no-clobber -O "yolov3.weights" "https://pjreddie.com/media/files/
```

```
File ‘yolov3.weights’ already there; not retrieving.
```

In [9]:
```
!wget --no-clobber -O "yolov3.cfg" "https://opencv-tutorial.readthedocs.i
```

```
File ‘yolov3.cfg’ already there; not retrieving.
```

In [10]:
```
img_path="n03384352_forklift.JPEG"
window_name="window"
img = cv.imread(img_path)
```

In [11]:
```
# https://docs.opencv.org/4.8.0/db/d5b/tutorial_py_mouse_handling.html
drawing = False
last = (0, 0)
def interactive_interface(event, x, y, flags, param):
    global drawing
    global last
    if event == cv.EVENT_LBUTTONDOWN:
        drawing = True
        last = (x, y)
    if event == cv.EVENT_LBUTTONUP:
        if drawing:
            cv.line(img, (x,y), last, (0, 255,0), 5)
        drawing = False
    if event == cv.EVENT_MOUSEMOVE:
        if drawing:
            cv.line(img, (x,y), last, (0, 255, 0), 5)
            cv.circle(img, (x,y), 6, (0, 255, 255), -1)
        last = (x, y)
```

In [12]:
```
# grabCut tutorial https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut
def background_subtraction(window_name, img):
    cv.setWindowTitle(window_name, "Select rectangle and press space")
    rect = cv.selectROI(window_name, img)
    cv.setWindowTitle(window_name, "Background is subtracted. z to reset"

    mask = np.zeros(img.shape[:2], np.uint8)
    bgdModel = np.zeros((1,65), np.float64)
    fgdModel = np.zeros((1,65), np.float64)

    cv.grabCut(img, mask, rect, bgdModel, fgdModel, 5, cv.GC_INIT_WITH_RE

    mask2 = np.where((mask == 2) | (mask == 0), 0, 1).astype('uint8')
    img = img * mask2[:, :, np.newaxis]

    return img
```

In [13]:
```
def object_highlighting(img):
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    _, binary = cv.threshold(gray, 0, 255, cv.THRESH_BINARY + cv.THRESH_O
    contours, heirarchy = cv.findContours(binary, cv.RETR_TREE, cv.CHAIN_
    highlighted_image = np.zeros_like(img)
    cv.drawContours(highlighted_image, contours, -1, (0, 255, 0), 2)
    return highlighted_image
```

In [14]:
```
# YOLO Tutorial https://opencv-tutorial.readthedocs.io/en/latest/yolo/yol
classes = open('coco.names').read().strip().split('\n')
np.random.seed(0)
colors = np.random.randint(0, 255, size=(len(classes), 3), dtype='uint8')

net = cv.dnn.readNetFromDarknet('yolov3.cfg', 'yolov3.weights')
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)

layer_names = net.getLayerNames()
layer_names = [layer_names[i-1] for i in net.getUnconnectedOutLayers()]
blob = cv.dnn.blobFromImage(img, 1/255.0, (416, 416), swapRB=True, crop=F
r = blob[0, 0, :, :]

net.setInput(blob)
yolo_outputs = net.forward(layer_names)
```

In [15]:
```
# YOLO Tutorial https://opencv-tutorial.readthedocs.io/en/latest/yolo/yol
def object_detection(event, X, Y, flags, param):
    cv.setWindowTitle(window_name, f"Hover to detect. ({X}, {Y})")
    boxes = []
    confidences = []
    classIDs = []
    h, w = img.shape[:2]
    for output in yolo_outputs:
        for detection in output:
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]
            if confidence > 0.3:
                box = detection[:4] * np.array([w, h, w, h])
                (centerX, centerY, width, height) = box.astype("int")
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))
                if X >= x and X <= x + width:
                    if Y >= y and Y <= y + height:
                        box = [x, y, int(width), int(height)]
                        boxes.append(box)
                        confidences.append(float(confidence))
                        classIDs.append(classID)
    indices = cv.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
    if len(indices) > 0:
        for i in indices.flatten():
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            color = [int(c) for c in colors[classIDs[i]]]
            cv.rectangle(img, (x, y), (x + w, y + h), color, 3)
            text = "{}: {:.4f}".format(classes[classIDs[i]], confidences[
            cv.putText(img, text, (x, y - 5), cv.FONT_HERSHEY_DUPLEX, 0.6
```

In [16]:
```
cv.startWindowThread()
cv.namedWindow(window_name)
```

How to use interactive interface

Press the following keys on keyboard for each function

- `d` for drawing. `left-click` then drag to draw
- `r` for selecting
- `b` for background subtraction, using **grabCut** method. Select rectangle then press `space`
- `h` for object highlighting, using **contours**
- `o` for object detection. using **YOLOv3** algorithm. Hover over the objects to detect them
- `z` for reset
- `q` for quit

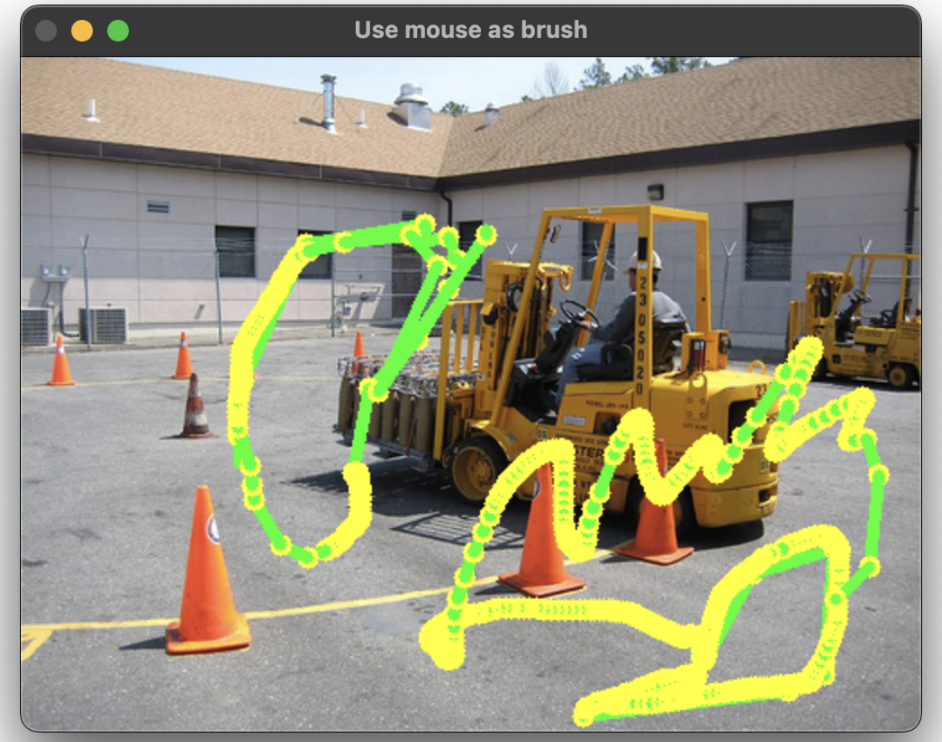
In [17]:
```
drawingMode = False
objectMode = False

INSTRUCTIONS = "q:quit z:reset d:draw b:background h:highlight o:objdetec
cv.setWindowTitle(window_name, INSTRUCTIONS)

while True:
    cv.imshow(window_name,img)

    code = cv.waitKey(1)

    if code == ord('d'):
        if not drawingMode:
            cv.setMouseCallback(window_name, interactive_interface)
            drawingMode = True
            cv.setWindowTitle(window_name, "Use mouse as brush")
        else:
            drawingMode = False
            cv.setWindowTitle(window_name, INSTRUCTIONS)

    if code == ord('r'):
        # just for demonstration of ROI, use 'b' for actual bg subtractio
        # use space or enter to finish selection, use key c to cancel sel
        cv.selectROI(window_name, img)

    if code == ord('b'):
        img = background_subtraction(window_name, img)

    if code == ord('h'):
        img = object_highlighting(img)
        cv.setWindowTitle(window_name, "Highlighted Contours. z to reset"

    if code == ord('o'):
        if not objectMode:
            cv.setMouseCallback(window_name, object_detection)
            objectMode = True
            cv.setWindowTitle(window_name, "Hover to detect.")
        else:
```

```
            cv.setMouseCallback(window_name, lambda *args: None)
            objectMode = False
            cv.setWindowTitle(window_name, INSTRUCTIONS)
            img = cv.imread(img_path)

    if code == ord('q'):
        break

    if code == ord('z'):
        img = cv.imread(img_path)
        cv.setWindowTitle(window_name, INSTRUCTIONS)
        cv.setMouseCallback(window_name, lambda *args: None)
        drawingMode = False
        objectMode = False
        drawing = False
cv.destroyAllWindows()
cv.waitKey(1)
```

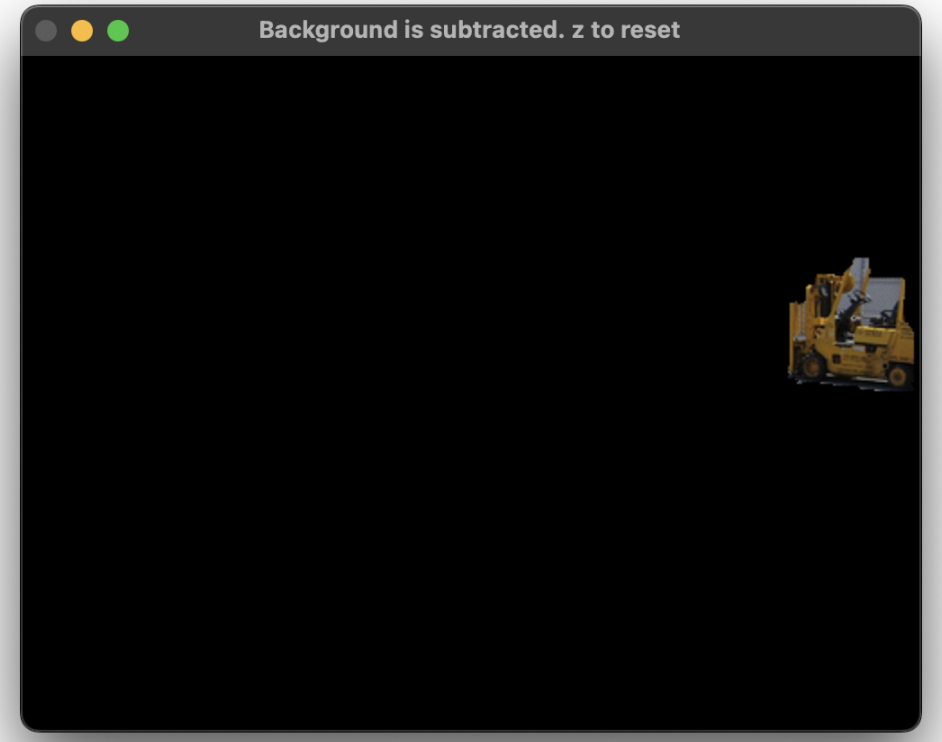
Out[17]: -1

Results

Interactive Interface

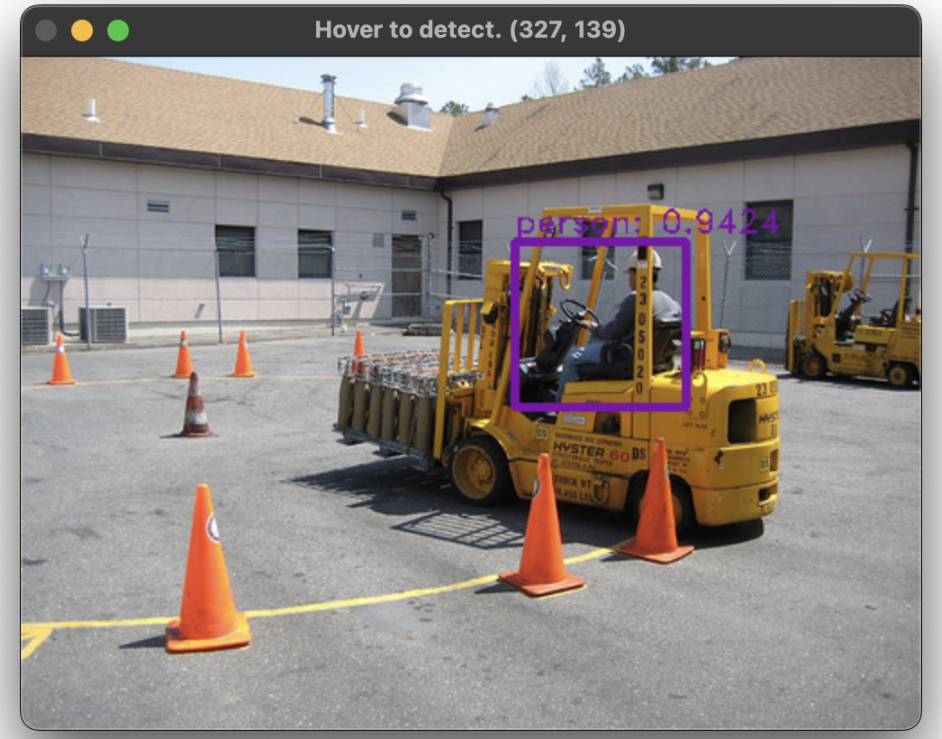


Background Subtraction



Object Highlighting

Object Detection



Conclusion

- We have been able to implement an interactive interface using OpenCV's `namedWindow` and `setMouseCallback`
- We can draw with mouse like a paint brush
- We can subtract background using grabCut algorithm
- We can highlight objects on basis of their contour
- We can detect objects on hovering them using YOLOv3