# Physical Object Tracking with Machine Learning

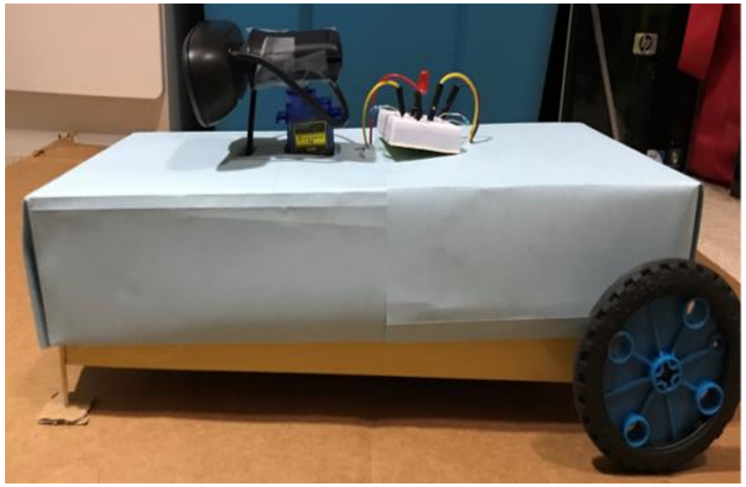
By: Cymberly Tsai

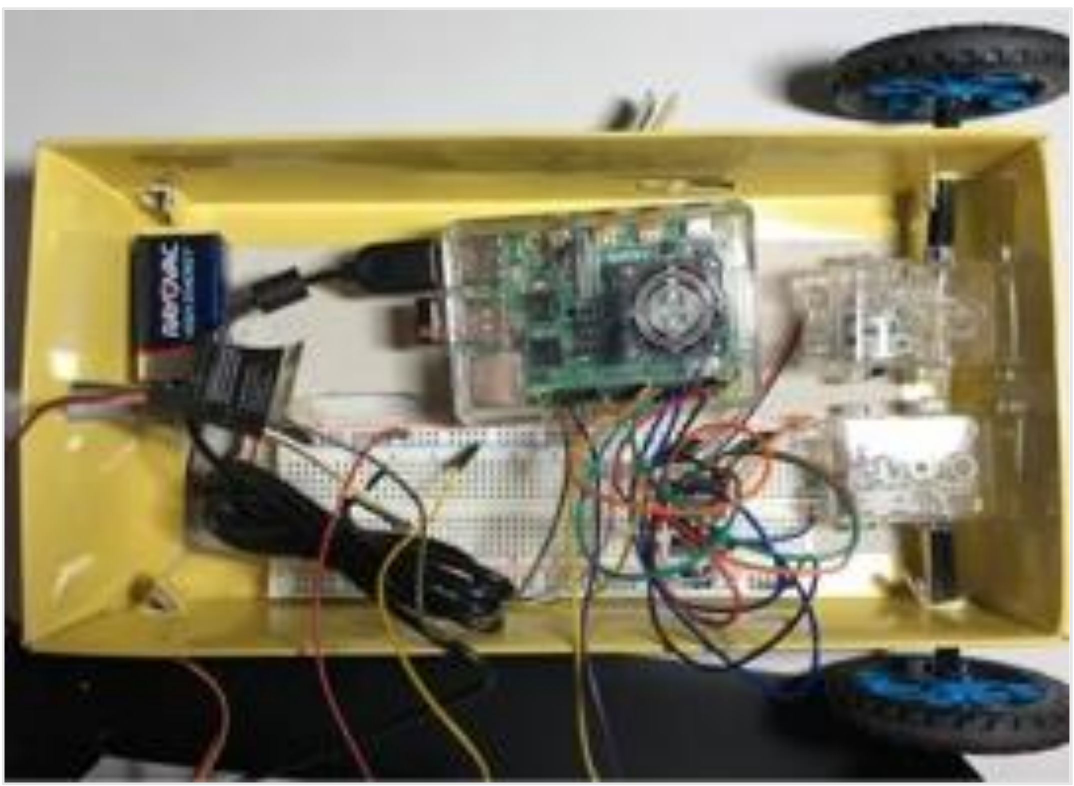
# Introduction

Object Detection

Machine Learning

- Real World Applications
  - Emergency services
  - Medical field
  - Industrial automation
  - Consumer production
- Engineering Goal: Create a machine
  - To detect and locate an item
  - To move toward the item
  - To stop when within 10 cm of the item and signal attached LED light





# Materials

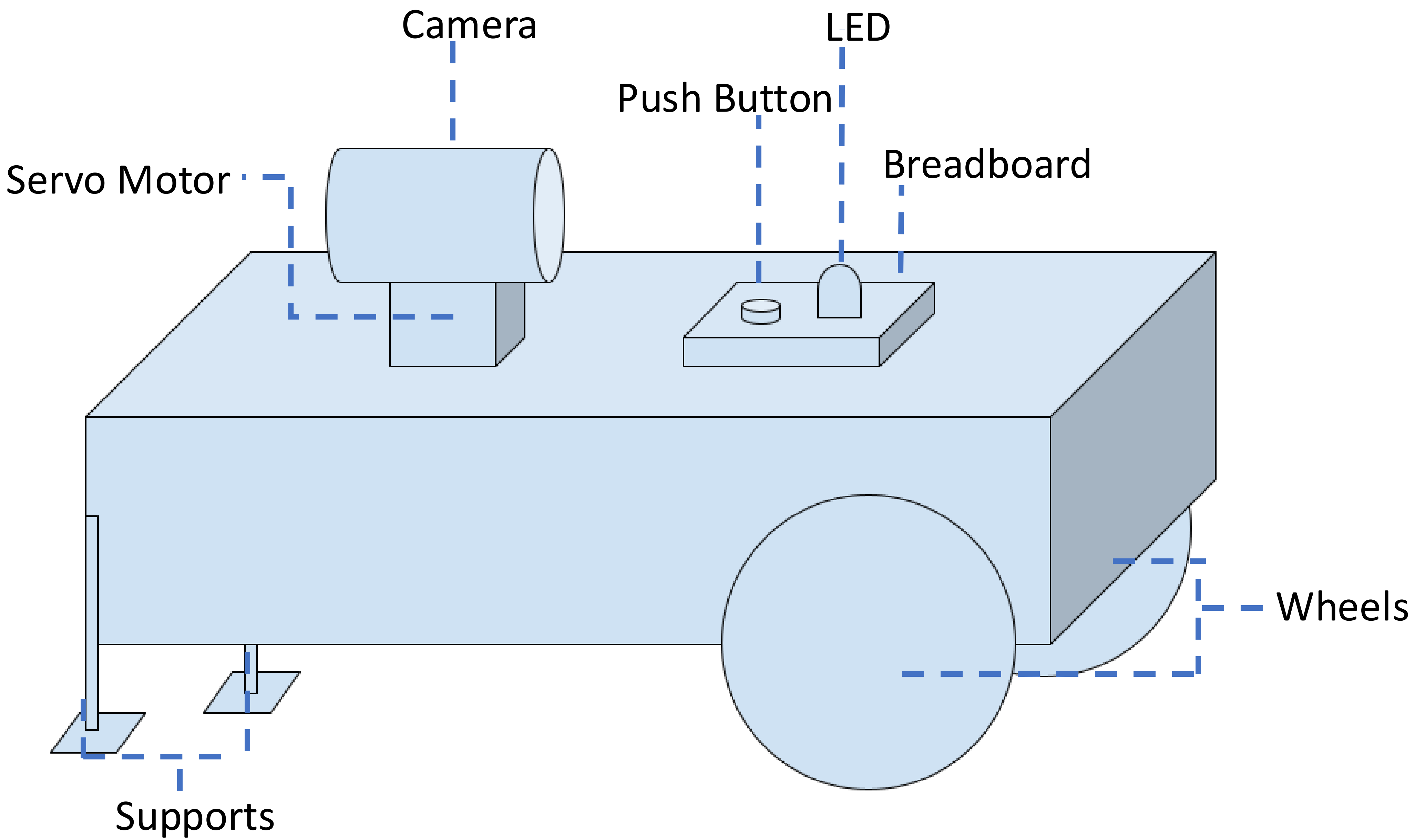
- DC Motors
- Motor Driver (L293D)
- Servo Motor (SG90)
- Wires
- Breadboard
- 9V Battery
- Push Button
- Led Light
- Resistors (330Ω)
- Raspberry Pi 4
- Power bank

- Logitech C310 Webcam

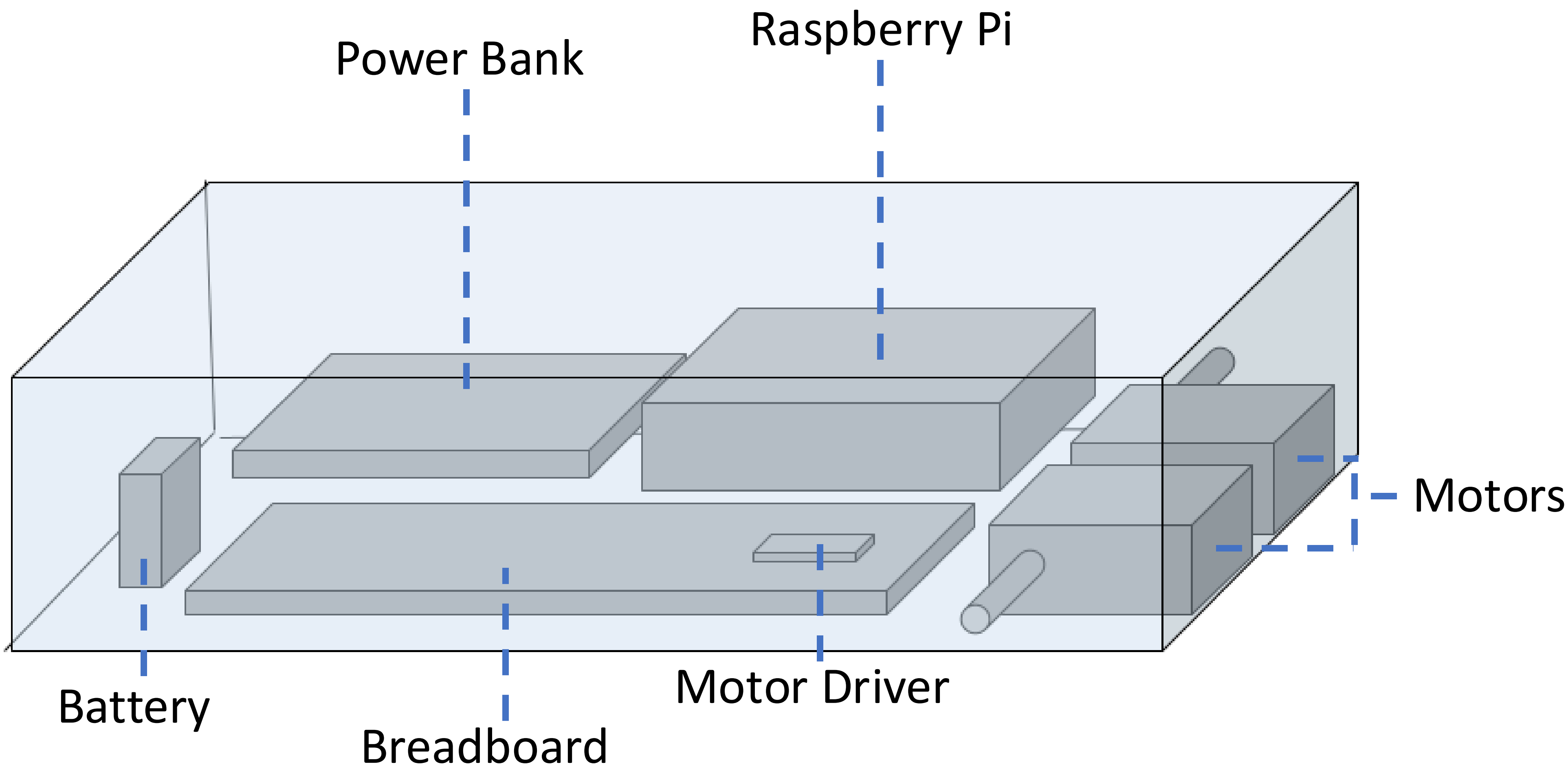
---------------------

- Python 3.7
- Open CV
- NumPy
- YOLOv3 -320

# Design

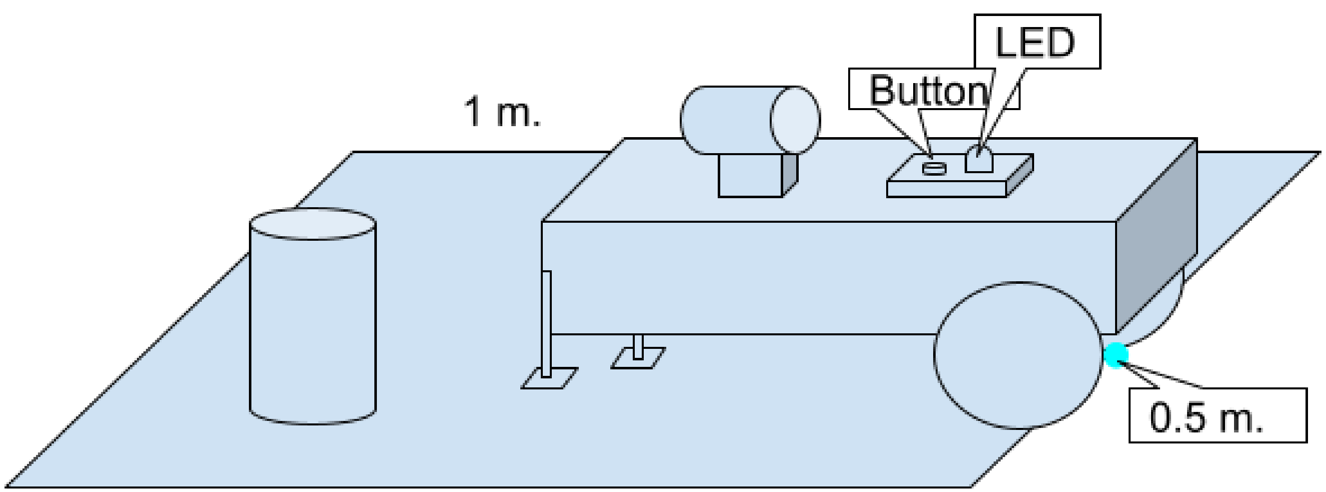

Camera
LED
Push Button
Breadboard
Servo Motor
Wheels
Supports

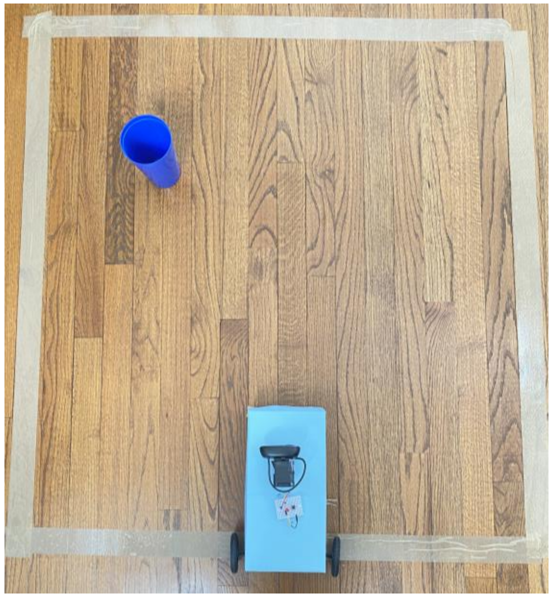
# Design


Power Bank
Raspberry Pi
Motors
Battery
Breadboard
Motor Driver

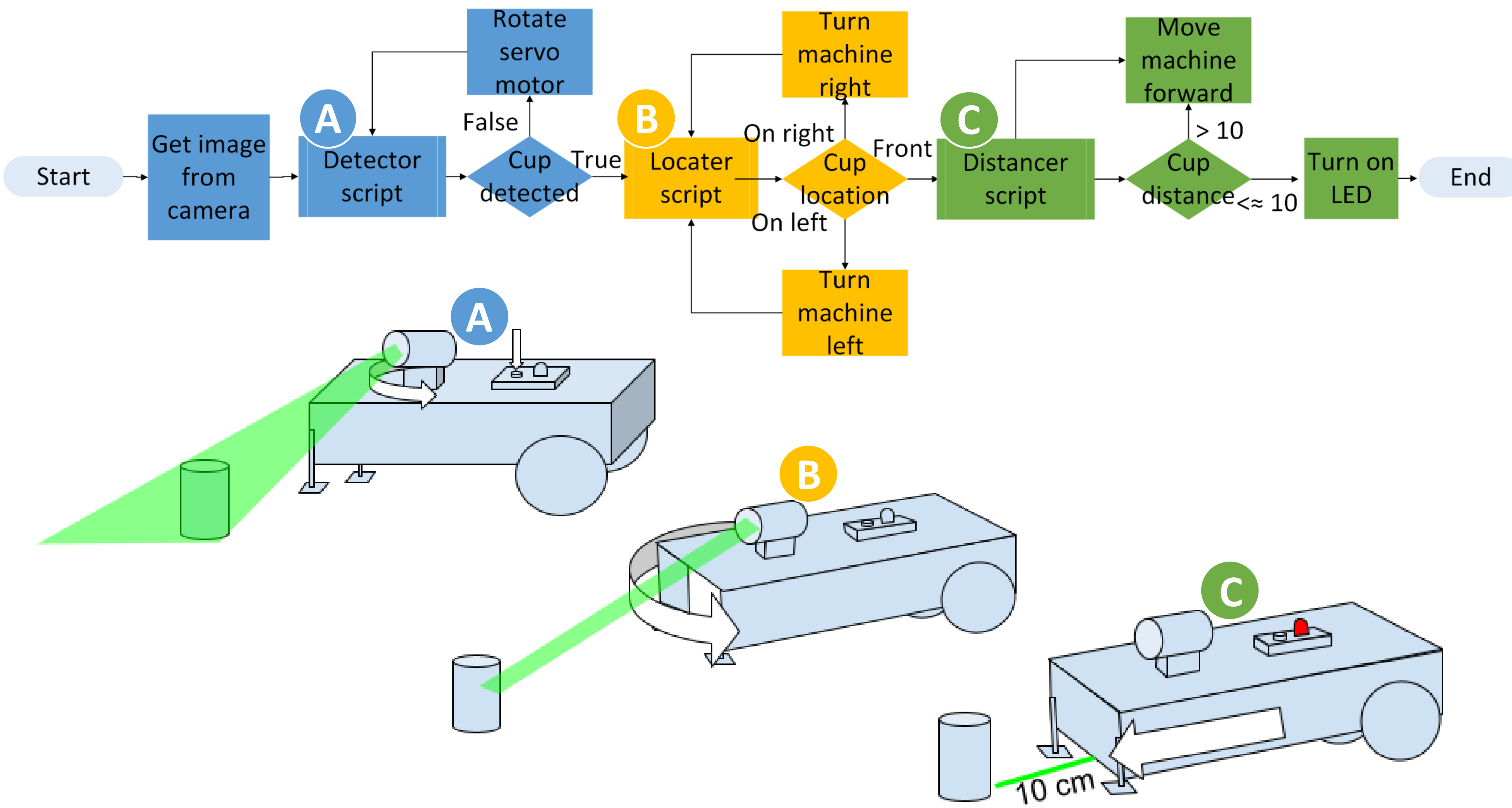
# Procedure

1. Place machine on edge of 1 m square
2. Place cup at random location in 1 m square
3. Initiate program with attached push button
4. Allow program to run and machine to move
5. Wait until LED light turns on
6. Record distance between cup and machine
7. Repeat steps 1 through 6 ten times



# Program Flowchart



# Detection

- You Only Look Once (YOLO Algorithm)
  - Pre-trained machine learning model
  - Object detection, classification, localization
  - Fast and accurate
    - 0.5 Mean Average Precision (mAP)
    - 2x faster than models of same mAP
- Process whole image at once (Fig. 1)
  - Divides image into cells (Fig.2)
  - Filter image through layers
  - Suppress redundant results
- Return results
  - Classification
  - Height and width
  - Center coordinate

Fig. 1

Fig. 2

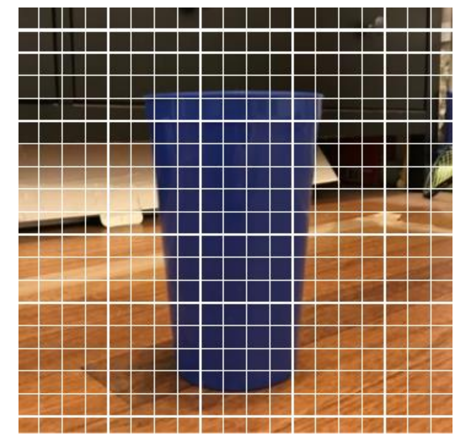


Fig. 3

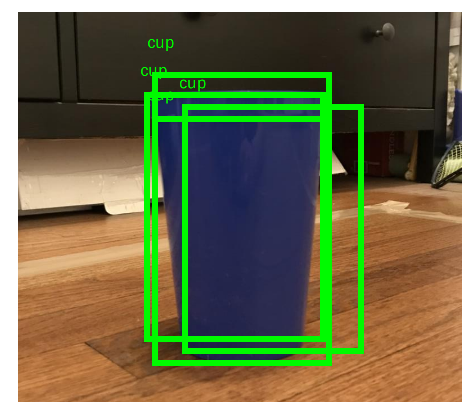
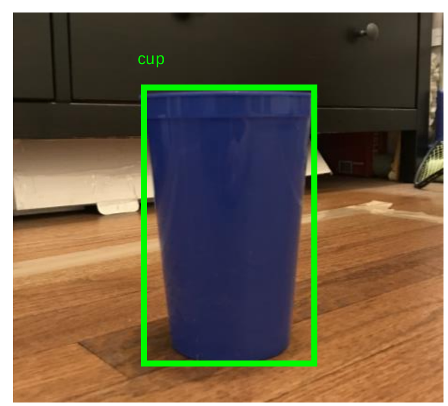



Fig. 4



# YOLO Algorithm Implementation Script

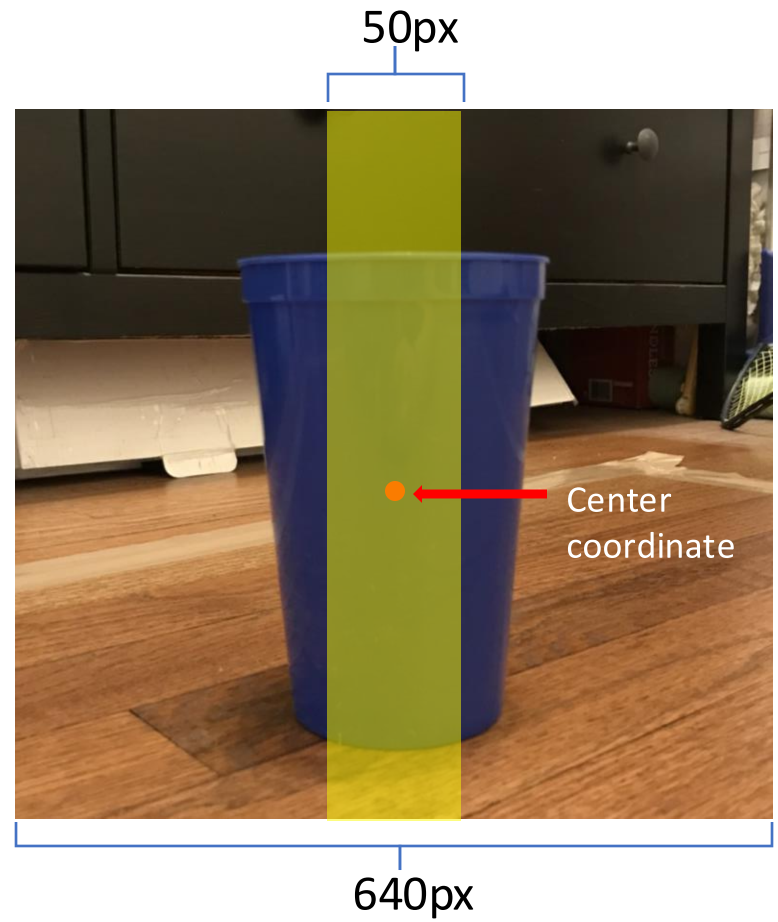
```
# Get image and run model
def snap():
  # Take image from camera
  cam = cv2.VideoCapture("/dev/video0")
  success, img = cam.read()
  # Convert image and input to model
  nImg = cv2.dnn.blobFromImage(...)
  net.setInput(nImg)
  # Get and return model outputs
  outLayeri = []
  for n in net.getUnconnectedOutLayers():
    outLayeri.append(net.getLayerNames())
   outputs = net.forward(outLayeri)
   cv2.waitKey(0)
   return [outputs, nImg]
```

```
# Run functions and return results
def pops():
  snapped = snap()
  crackled =crackle(snapped[0],snapped[1])
  if len(crackled) > 0:
    for t in crackled:
      if t[0] == "cup":
       return crackled[crackled.index(t)]
  else:
    return ["no cup :("]
```

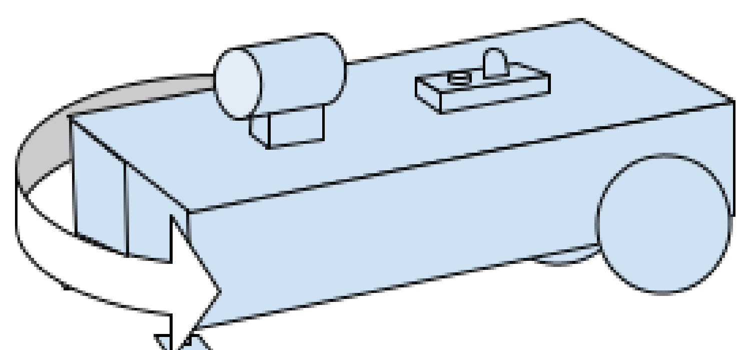
```
# Filter and format model outputs
def crackle(outs, img):
  # Variables for results
  boxi = []
  idi = []
  confi = []
  midi = []
  # Loop through outputs
  for a in outs:
    for b in a:
    # Filter out low confidence score
      scores = b[5:]
      classid = np.argmax(scores)
      con = scores[classid]
      if con > 0.4
        # Format valid outputs
        (…)
  # Filter out redundant results
  keep =
  cv2.dnn.NMSBoxes(boxi, confi, 0.4, 0.3)
  trueOut = []
  for k in keep:
    b = boxi[k[0]]
    x, y, w, h = b
    trueOut.append(...)
  return trueOut
```

# Locating

- Determines location of cup
- Given center X coordinate
- Compare coordinate to image
  - Image 640px wide
  - 295px to 345px center range
- Machine turns toward cup

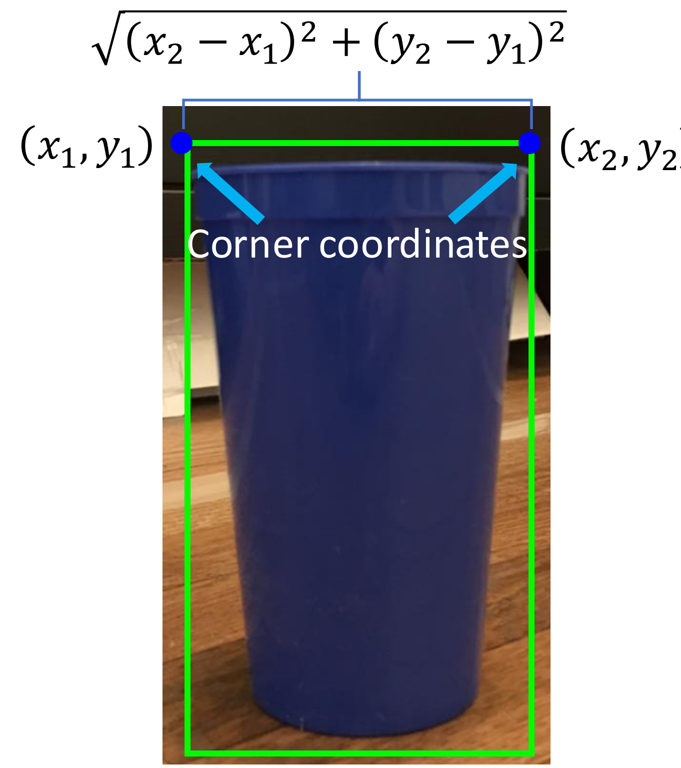



```
# Locating function
def find(x):
  if x > 345:
    return ‘l’
  elif x < 295:
    return ‘r’
  else:
    return ‘s’
```

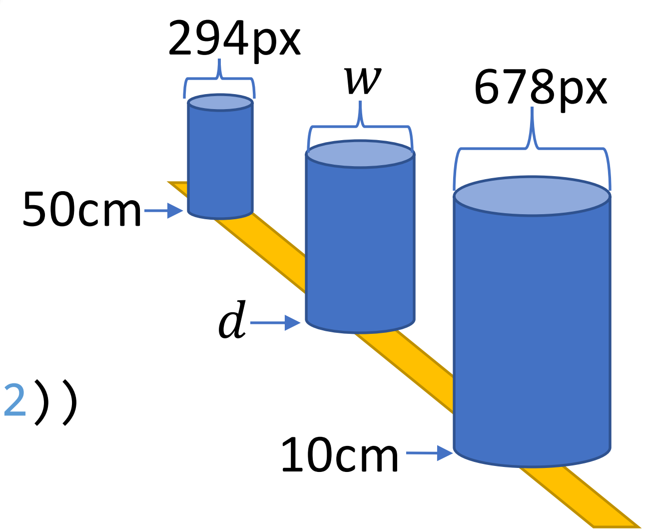


# Distancing

- Determines distance to cup
- Given corner coordinates: $(x_1, y_1)$, $(x_2, y_2)$
- Equation for width to distance
  - $w$: Width of cup in pixels
    - $w = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
  - $d$: Distance to cup in centimeters
    - $d = 9.32d^2 - 1.82d + 9.36$
- Machine moves to cup




```
# Data and equation
w = [...]
d = [...]
eq = numpy.polyfit(w, d, 2)
# Distancing function
def measure(x1, y1, x2, y2):
  pixi = int(math.sqrt((y2 - y1)**2+(x2-x1)**2))
  centi =eq[0]*pixi**2 + eq[1]*pixi+ eq[2]
  return centi
```



# Main Script Pseudocode

```
import scripts and modules
define variables
set up hardware
def servoTest()
  Rotate servo motor
def motorTest()
  Control motors

def detectorTest():
  Run detector script
  While cup not in image:
    Run servoTest()
    Run detection script
  For servo position to center
    Turn with motorTest()
  Rotate servo to center
```

```
def letsGo():
  Rotate servo to right
  Run detectorTest()
  Run locater script
  While cup not in image center:
    Turn with motorTest()
    Run detectorTest()
    Run locater script
  Run detectorTest()
  Run distance script
  While cup over 10cm away:
    Move forward with motorTest()
    Run detectorTest()
    Run distance script
  Turn on LED light

While True:
   If button pressed
      Run letsGo()
```

# Conclusion

| **Trial Number** | **Distance (cm)** |
|---|---|
| 1 | 7.5 |
| 2 | 4.2 |
| 3 | 0 |
| 4 | 10.8 |
| 5 | 0 |
| 6 | 0 |
| 7 | 3 |
| 8 | 6.5 |
| 9 | 8.3 |
| 10 | 0 |
| **Average** | **4.03** |

## Engineering goal achieved

- Distance within 10cm.
- Zeros indicate contact
- Single outlier

## Possible limitations

- Time consuming
- Inconsistent
- Poor turning mechanism

## Future improvements

- Larger data collection
- Different detection model
- Additional motors

## Further investigation

- More trials
- Larger field
- Multiple objects

# Bibliography


- Bandyopadhyay, Hmrishav. "Yolo: Real-Time Object Detection Explained." V7 Labs, 19 Mar. 2022, https://www.v7labs.com/blog/yolo-object-detection. Accessed 30 Mar. 2022
- Brownlee, Jason. "4 Distance Measures for Machine Learning." Machine Learning Mastery, Mar. 2020, machinelearningmastery.com/distance-measures-for-machine-learning/. Accessed 4 Nov. 2021.
- Gupta, Manish. "Yolo – You Only Look Once: A state of the Art Algorithm for Real-Time Object Detection System." Towards Data Science, 30 May 2020, https://towardsdatascience.com/yolo-you-only-look-once. Accessed 4 Nov 2020
- Gupta, Shubham. "Introduction to Object Detection." Hacker Earth, 7 Aug. 2018, www.hackerearth.com/ blog/developers/introduction-to-object-detection/. Accessed 4 Nov. 2021.
- "Raspberry Pi Documentation." Raspberry Pi, www.raspberrypi.com/documentation/computers/os.html. Accessed 4 Nov. 2021.
- Raspberry Pi Pin Layout. Raspberry Pi, www.raspberrypi.com/documentation/computers/os.html. Accessed 4 Nov. 2021.
- Redmon, Joseph and Ali Farhadi. "Yolov3: An Incremental Improvement." ArXiv, 2018.
- Saha, Sumit. "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 Way." Towards Data Science, 15 Dec. 2018, towardsdatascience.com/ a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Accessed 4 Nov. 2021