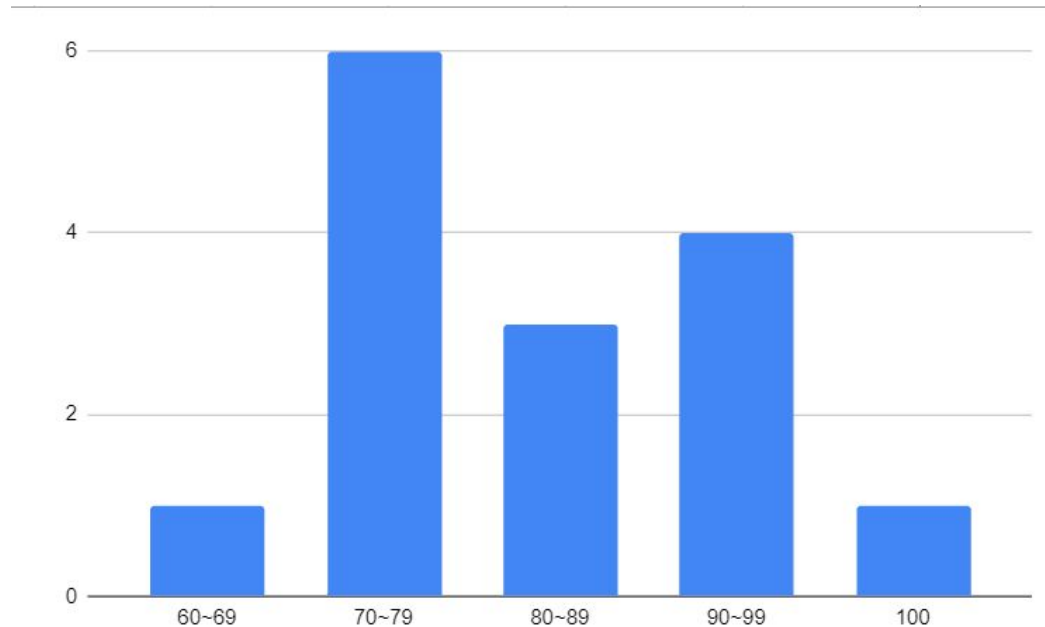


# Lab08

# Midterm score

[https://docs.google.com/spreadsheets/d/1DcLjN0H5C7ECJH3LZxlzj14pxTBRb09GayRh\\_bI0Bb4/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1DcLjN0H5C7ECJH3LZxlzj14pxTBRb09GayRh_bI0Bb4/edit?usp=sharing)



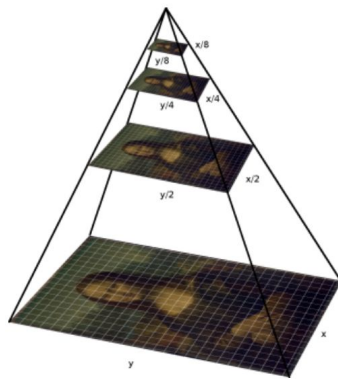
# 目標

- 利用HOG行人檢測及Haar-cascade臉部偵測框出人(25%)與人臉 (25%)
- 利用任一方法算出與其的距離
- demo時為即時影像並用尺量人(25%)與人臉 (25%) 距離準確度
- demo誤差: 人(50cm)、人臉(10cm)

# HOG(Histogram of Oriented Gradient)

*# initialize the HOG descriptor/person detector*

- `hog = cv2.HOGDescriptor()`
  - `hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())`
  - `rects, weights = hog.detectMultiScale(src, #輸入圖`
- winStride**, #在圖上抓取特徵時窗口的移動大小
- scale**, #抓取不同scale (越小就要做越多次)
- `useMeanshiftGrouping = False)`



# Haar-cascade Face Detection

```
face_cascade =  
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

```
rects = face_cascade.detectMultiScale(frame,  
    ScaleFactor, #每次搜尋方塊減少的比例  
    minNeighbors, #每個目標至少檢測到幾次以上, 才可被認定是真數據。  
    minSize: #設定數據搜尋的最小尺寸, 如 minSize=(40,40)  
    )
```

# 畫出長方形

- `image = cv2.rectangle(image, start_point, end_point, color, thickness)`

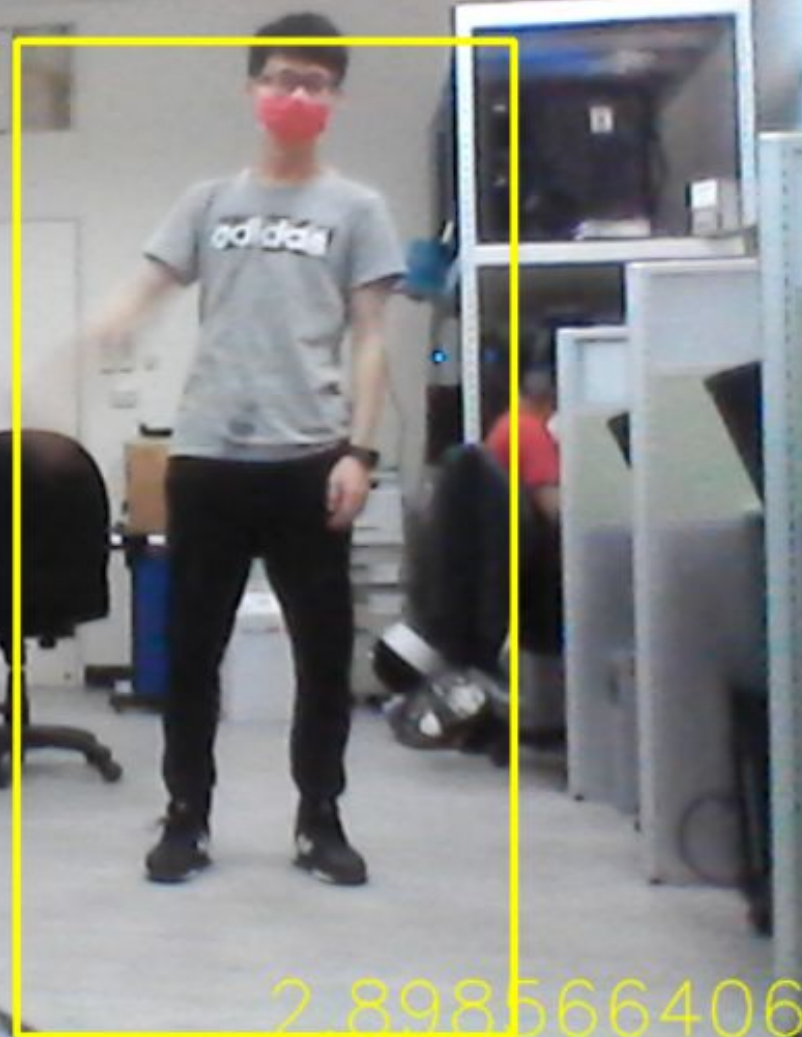
\*start\_point跟end\_point要是整數喔

# 深度預測

- 不限定方法

1. 已知物體大小及相機焦距，用物體在畫面中占的pixel計算  
物件的框會有留白，可以自行判斷要乘多少比例才是物體實際pixel大小
2. 假設人或人臉為平面，已知大小解SolvePnP
  - `cv2.solvePnP(objp, imgPoints, intrinsic, distortion) → retval, rvec, tvec`  
objp的部分要用真實的長度單位，非(0,0), (0,1), (1,0), (1,1)

[54 94645948]



2.898566406