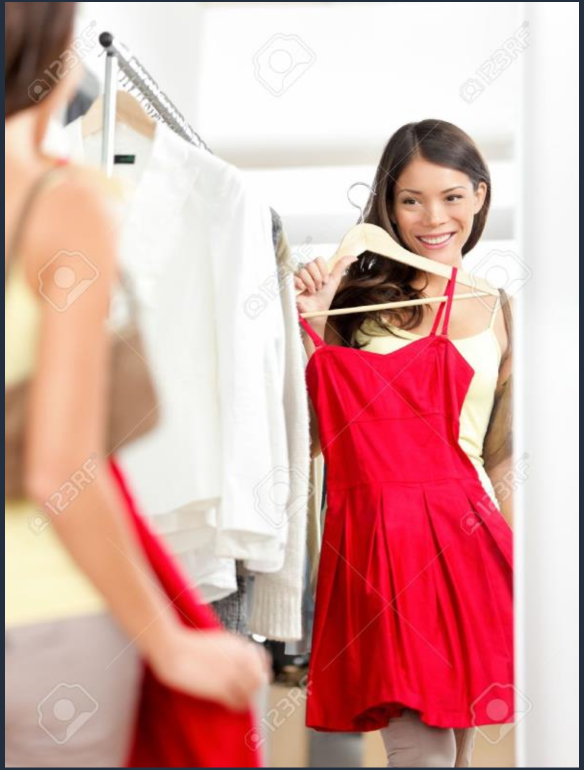


rantOn

Size and Fit in Online Shopping Experience

Indian Institute of Technology, Guwahati

- Tanvi Shahani
- Narmin Kauser
- Reshma Avvaru
- Arsh Kandroo

Major Problems in E-Commerce Apparels

rantOn

Size Estimation

- E-commerce, which is a **rapidly growing sector** now, to be more precise let us take the **clothing area** of e-commerce. One of the most pertinent problems online fashion faces is the customer dissatisfaction due to difference in expected and original size.

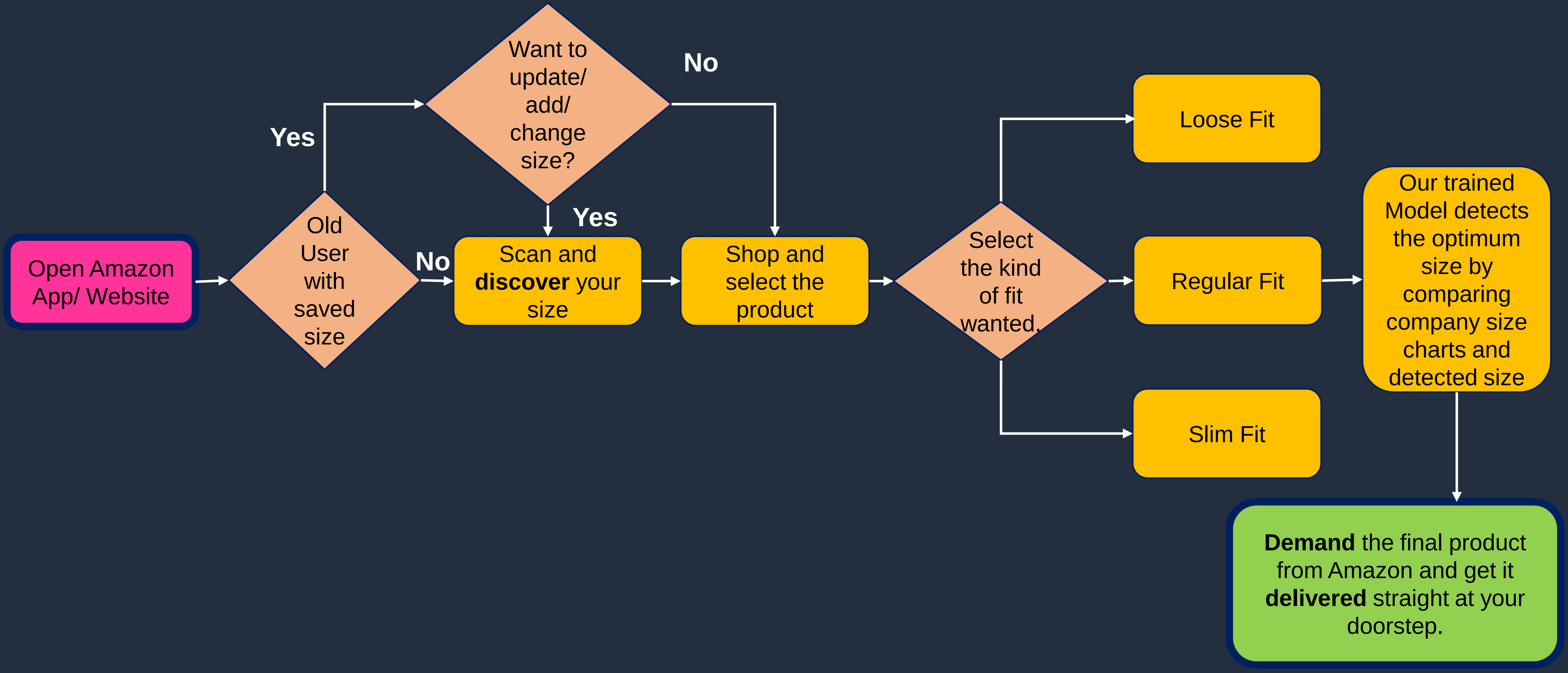
- Nowadays, even though technology has advanced so much, a wide group of people still resist buying clothing online. This happens majorly due to **dissimilarity in size charts** of different companies, **improper size and fit**, **high diversity in the body types** of consumers.

Product Fit

- Another big challenges online fashion faces is the large volume of returns. According to a survey , 23% of all clothing gets returned, and 64% of consumers say **incorrect fit** is the primary reason they return clothing. Size and Fit is a multidimensional problem influenced by various factors like inconsistency in size chart across brands, different body types and Individual fit choices.
- Our aim is to bring out an effective solution to the multidimensional problem of size and fit. The solution must solve the major issues of generating trust among users, creating a solution for uniform sizes across companies and provide a method to show how each specific product looks on every body type .

Product WorkFlow

rantOn


Open Amazon App/ Website
Old User with saved size
Yes
No
Want to update/ add/ change size?
No
Yes
Scan and **discover** your size
Shop and select the product
Select the kind of fit wanted.
Loose Fit
Regular Fit
Slim Fit
Our trained Model detects the optimum size by comparing company size charts and detected size
Demand the final product from Amazon and get it **delivered** straight at your doorstep.

rantOn

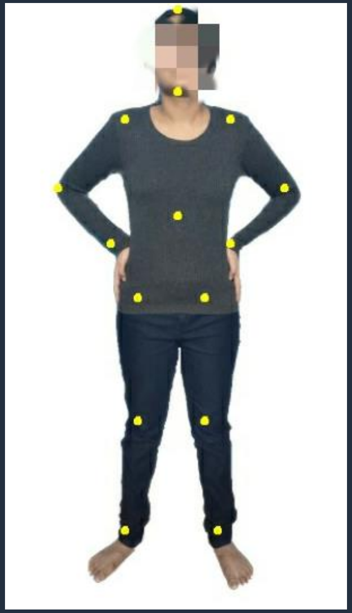
Size Estimation

Step By Step

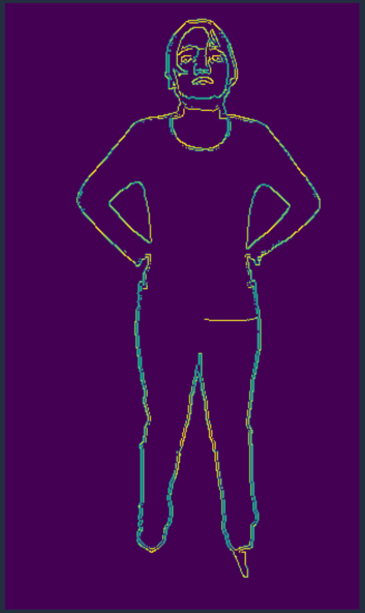
rantOn

01
Seeking Data

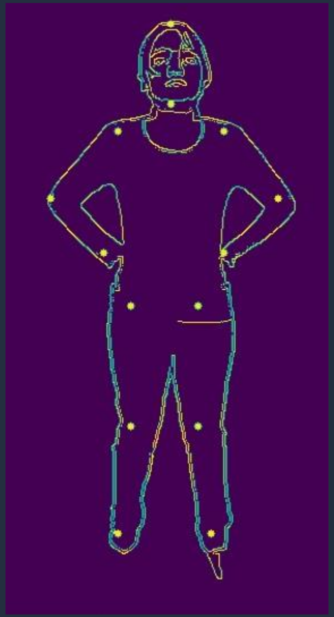
02
Finding the body points



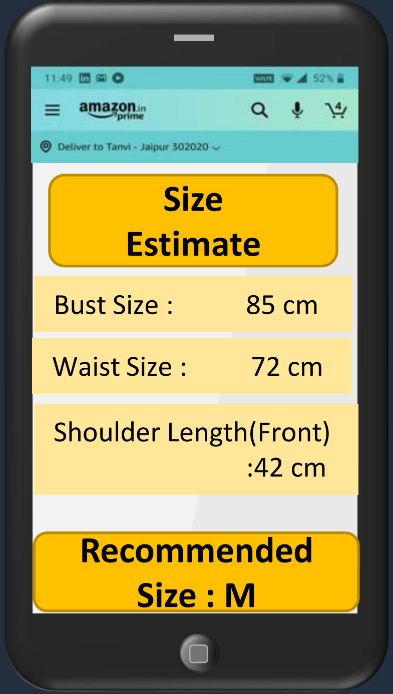
03
Tracing the contour



04
Overlapping points on the contour



05
Final size calculation



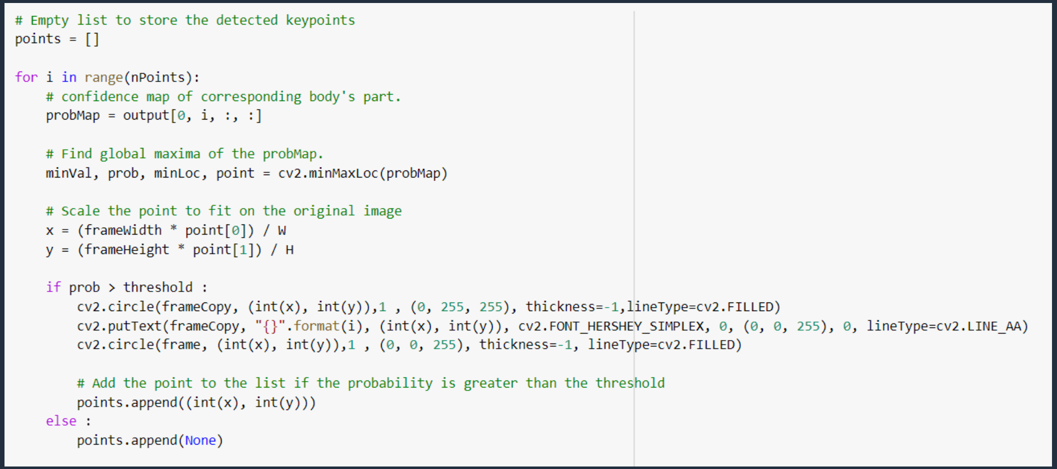
Size Estimate

Bust Size :	85 cm
Waist Size :	72 cm
Shoulder Length(Front)	:42 cm

Recommended Size : M

Code Explanation

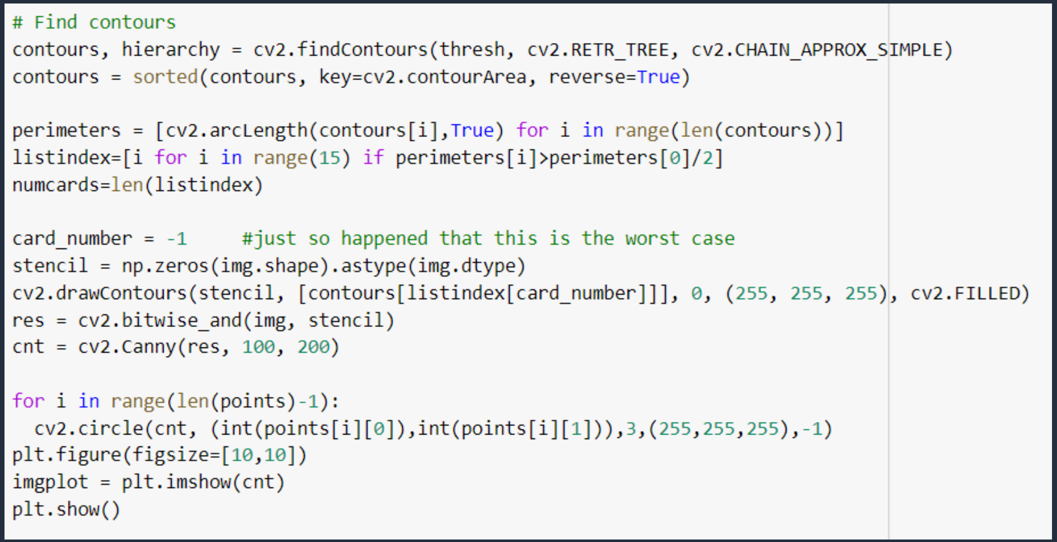
rantOn



```python
# Empty list to store the detected keypoints
points = []

for i in range(nPoints):
    # confidence map of corresponding body's part.
    probMap = output[0, i, :, :]

    # Find global maxima of the probMap.
    minVal, prob, minLoc, point = cv2.minMaxLoc(probMap)

    # Scale the point to fit on the original image
    x = (frameWidth * point[0]) / W
    y = (frameHeight * point[1]) / H

    if prob > threshold :
        cv2.circle(frameCopy, (int(x), int(y)),1 , (0, 255, 255), thickness=-1,lineType=cv2.FILLED)
        cv2.putText(frameCopy, "{}".format(i), (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0, (0, 0, 255), 0, lineType=cv2.LINE_AA)
        cv2.circle(frame, (int(x), int(y)),1 , (0, 0, 255), thickness=-1, lineType=cv2.FILLED)

        # Add the point to the list if the probability is greater than the threshold
        points.append((int(x), int(y)))
    else :
        points.append(None)
```

We have the height and photo given by the user.

The photo is sent to the notebook 'Size Estimator' and the body point allocation code written using OpenCV and OpenPose is run through it. This gives us the various points on the body basically of the joints. Coordinates of these body points are stored in a list named 'points' for future use.

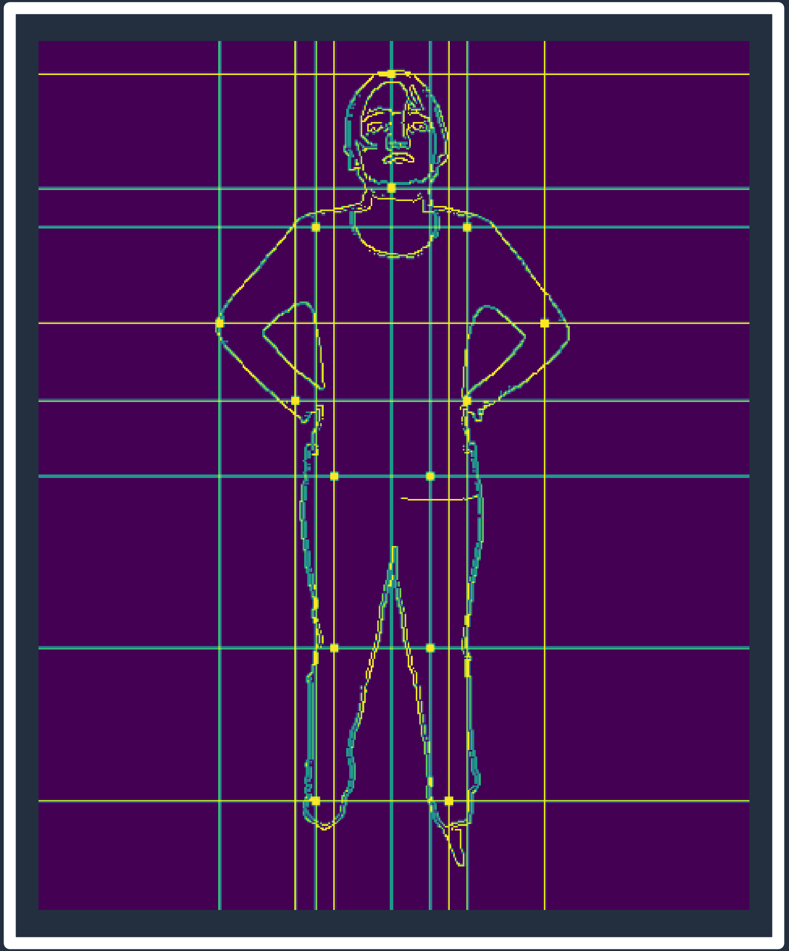


```python
# Find contours
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
contours = sorted(contours, key=cv2.contourArea, reverse=True)

perimeters = [cv2.arcLength(contours[i],True) for i in range(len(contours))]
listindex=[i for i in range(15) if perimeters[i]>perimeters[0]/2]
numcards=len(listindex)

card_number = -1     #just so happened that this is the worst case
stencil = np.zeros(img.shape).astype(img.dtype)
cv2.drawContours(stencil, [contours[listindex[card_number]]], 0, (255, 255, 255), cv2.FILLED)
res = cv2.bitwise_and(img, stencil)
cnt = cv2.Canny(res, 100, 200)

for i in range(len(points)-1):
  cv2.circle(cnt, (int(points[i][0]),int(points[i][1])),3,(255,255,255),-1)
plt.figure(figsize=[10,10])
imgplot = plt.imshow(cnt)
plt.show()
```

Since the points obtained don't represent the actual size of the person, we use a contour based model of OpenCV to get the contours of the actual flesh of the person. We then plot the points onto the contour and using an algorithm the nearest contour-point to every body point is calculated. This gives the precise measurement of the person in pixels. Pixels are then converted into centimeters.

Code Explanation

rantOn



We drew horizontal and vertical lines onto the contour we got to get the actual extreme points of shoulder, bust, hip and waist. Later these points were used to calculate actual sizes.

Formulas used:

- **Ratio** = Actual height of user / image height
- **Shoulder** = (Right shoulder extreme - Left shoulder extreme) * Ratio
- **Hip** = (Right hip point - Left hip point) * Ratio
- **Innerseam** = Inner ankle point - Centre point of body
- **Outerseam** = Outer ankle point - Hip extreme
- **Thigh** = 2 * pi * thigh_radius
- **Bust** = 2 * pi * sqrt(((bustx)*2) + ((bustz)*2)/2)
- **Waist** = 2 * pi * sqrt(((waistx)*2)+((waistz)*2)/2)

  These formulas used together gives the best size suggestion comparing it with the company’s size chart.

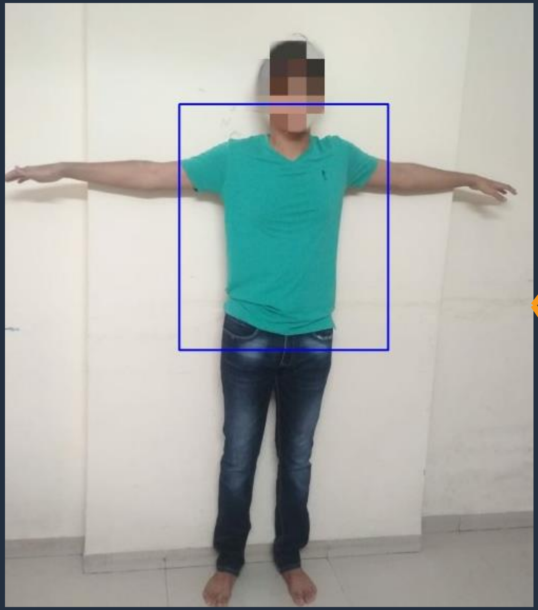
rantOn

Virtual Try-On

Step By Step

rantOn

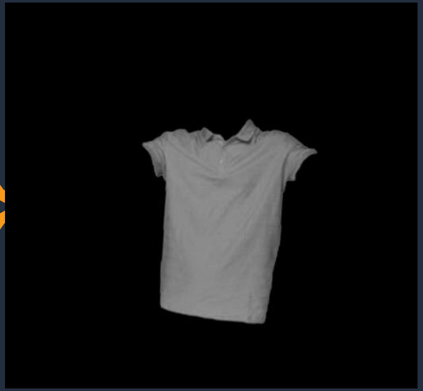
The Customer



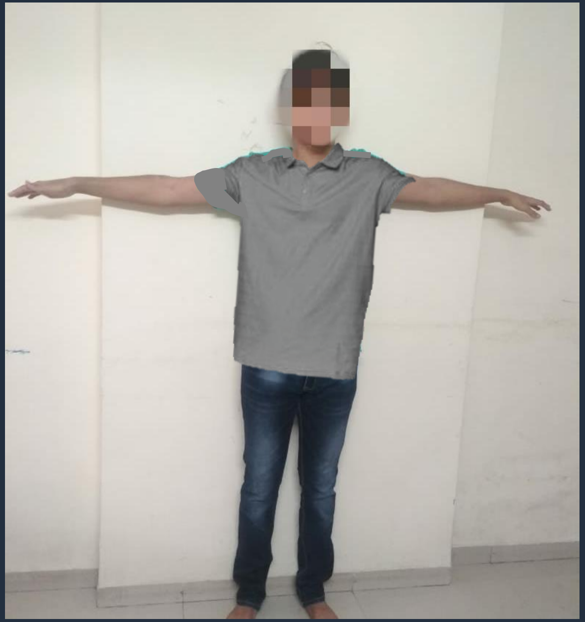
What they want to wear

What they are already wearing

Product Fit on their body



Virtual Try-On



Code Explanation

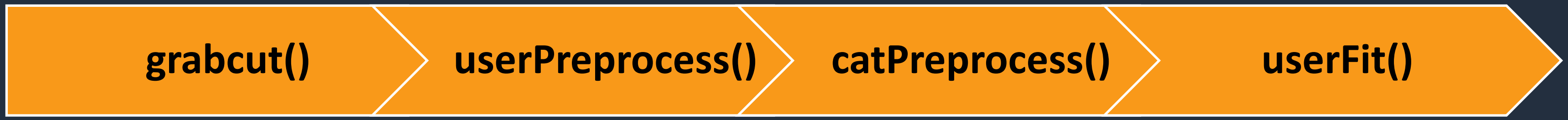
rantOn

- We are devising a solution to real-life trying on problem with the help of OpenCV Library.
- As we start to run the file 'RantOn_Final.py' , we give 3 arguments to our system : <Path_of_our_File> <Path_of_the_image_of_the_customer> <Path_of_the_image_of_the_apparel_we_want_to_try_on>
- We have three supporting .py file :

  - Apparel.py has functions that deals with the preprocessing of the new apparel we want to try on.

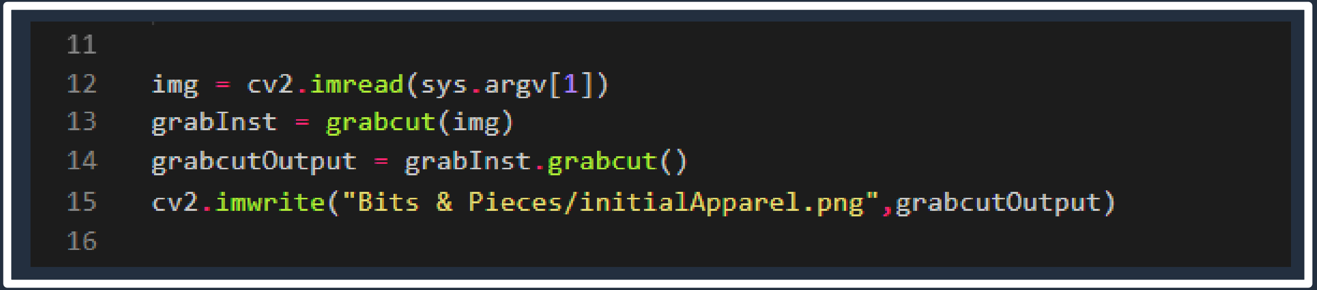
  - Customer.py has functions that deals with the preprocessing of the image of the customer.

  - Join.py has functions which fits and puts the new apparel onto the customer to give a very close real-life apparel trying experience.
- Given are the classes of function which are defined in the above mentioned files:



Code Explanation

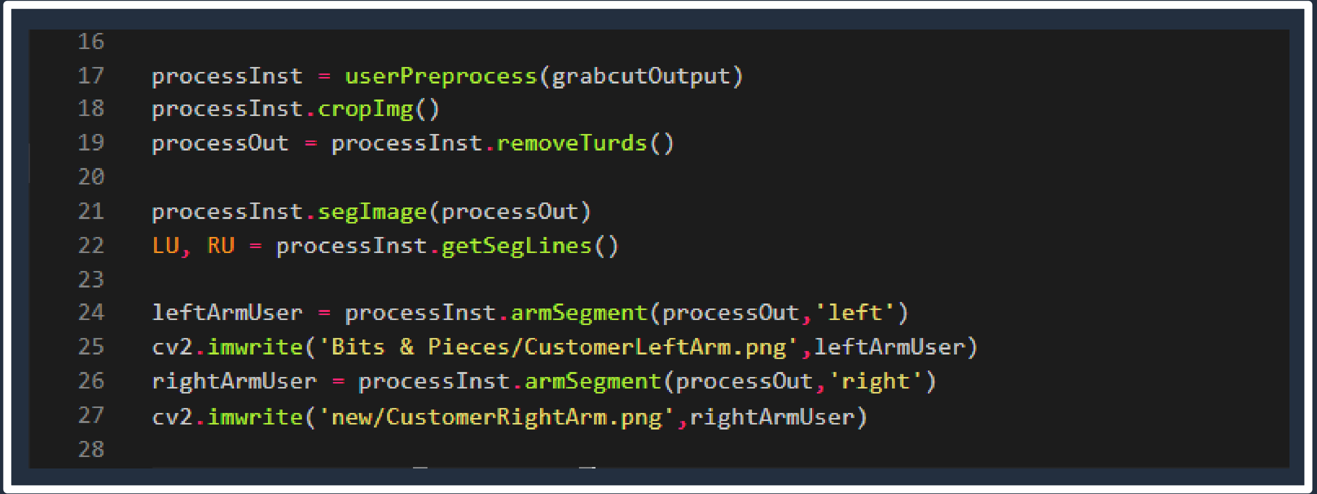
rantOn



```python
img = cv2.imread(sys.argv[1])
grabInst = grabcut(img)
grabcutOutput = grabInst.grabcut()
cv2.imwrite("Bits & Pieces/initialApparel.png",grabcutOutput)
```

grabcut()

Takes the required portion of the customer's image, and gathers the cutout of the already worn up clothing.



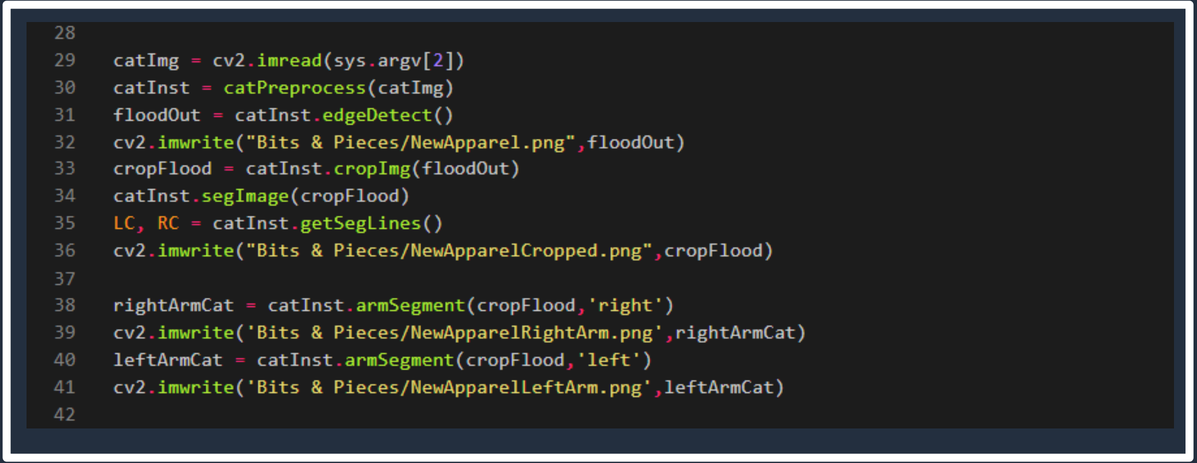
```python
processInst = userPreprocess(grabcutOutput)
processInst.cropImg()
processOut = processInst.removeTurds()

processInst.segImage(processOut)
LU, RU = processInst.getSegLines()

leftArmUser = processInst.armSegment(processOut,'left')
cv2.imwrite('Bits & Pieces/CustomerLeftArm.png',leftArmUser)
rightArmUser = processInst.armSegment(processOut,'right')
cv2.imwrite('new/CustomerRightArm.png',rightArmUser)
```

userPreprocess()

Breaks up the cutout gathered from the last step into various sections (at the joints) to get a better analysis and size of each & every part at minute stages as well.

Code Explanation

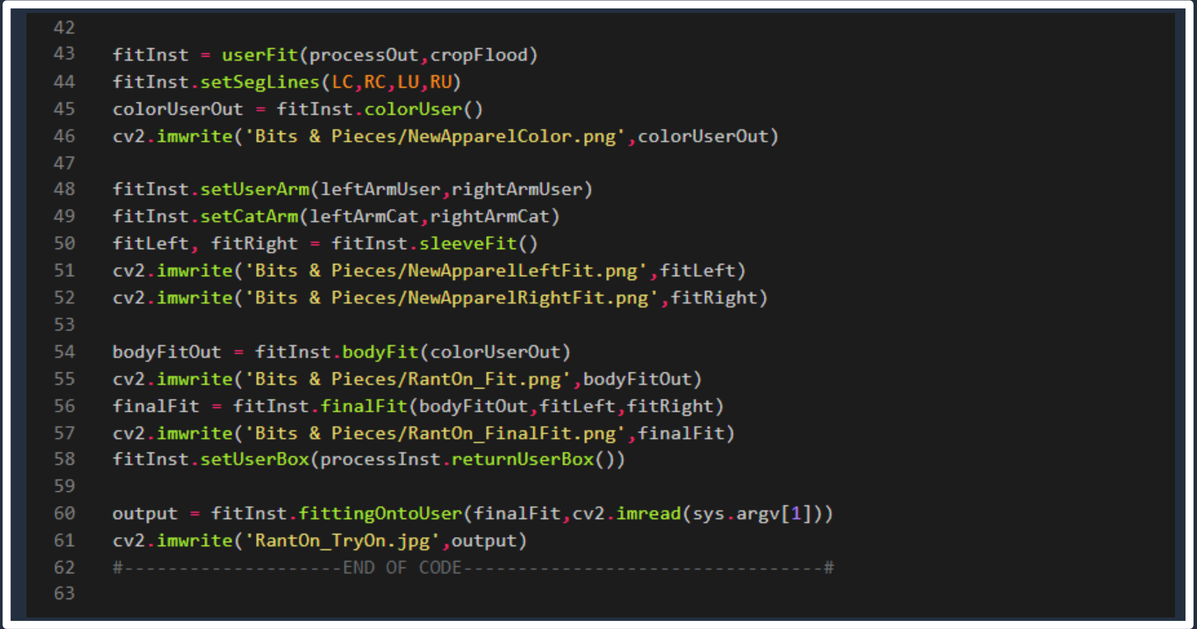
rantOn



```
catImg = cv2.imread(sys.argv[2])
catInst = catPreprocess(catImg)
floodOut = catInst.edgeDetect()
cv2.imwrite("Bits & Pieces/NewApparel.png",floodOut)
cropFlood = catInst.cropImg(floodOut)
catInst.segImage(cropFlood)
LC, RC = catInst.getSegLines()
cv2.imwrite("Bits & Pieces/NewApparelCropped.png",cropFlood)

rightArmCat = catInst.armSegment(cropFlood,'right')
cv2.imwrite('Bits & Pieces/NewApparelRightArm.png',rightArmCat)
leftArmCat = catInst.armSegment(cropFlood,'left')
cv2.imwrite('Bits & Pieces/NewApparelLeftArm.png',leftArmCat)
```

catPreprocess()

Takes up the new apparel's flattened image, breaks it and resizes it with the cutout sections of the original clothing.



```

fitInst = userFit(processOut,cropFlood)
fitInst.setSegLines(LC,RC,LU,RU)
colorUserOut = fitInst.colorUser()
cv2.imwrite('Bits & Pieces/NewApparelColor.png',colorUserOut)

fitInst.setUserArm(leftArmUser,rightArmUser)
fitInst.setCatArm(leftArmCat,rightArmCat)
fitLeft, fitRight = fitInst.sleeveFit()
cv2.imwrite('Bits & Pieces/NewApparelLeftFit.png',fitLeft)
cv2.imwrite('Bits & Pieces/NewApparelRightFit.png',fitRight)

bodyFitOut = fitInst.bodyFit(colorUserOut)
cv2.imwrite('Bits & Pieces/RantOn_Fit.png',bodyFitOut)
finalFit = fitInst.finalFit(bodyFitOut,fitLeft,fitRight)
cv2.imwrite('Bits & Pieces/RantOn_FinalFit.png',finalFit)
fitInst.setUserBox(processInst.returnUserBox())

output = fitInst.fittingOntoUser(finalFit,cv2.imread(sys.argv[1]))
cv2.imwrite('RantOn_TryOn.jpg',output)
#--------------------END OF CODE--------------------------------#
```

userFit()

Takes up all the modified sections of the new apparel and places them at their appropriate positions on the image of the customer to get a near to real-life visual of the apparel on the customer

Problems Faced In Implementation

rantOn

Problem	Problem Description	Solution
Image Posture	Improper posture may result in incorrect results to find out size.	Directing customers clearly with a borderline of the best posture.
Difference in measurement scales	While taking a photograph, the measuring scale is in pixels whereas while size estimation, it is in centimetres/inches.	Asking for height, that is the most commonly known feature to figure out ratios.
Loosely fit clothes	People while estimating size may have improper/loose clothes on which can be deceptive in size estimation	Clear Instructions for wearing fitted clothes for the best size detection.
Product Image Flattening	Since a lot of times, product images are put in a 3-D manner and while virtually overlapping the product to customer, image flattening is required.	Passing through suitable function and applying certain geometrical operations to flatten the 3-D images of the apparel.

* Due to time restrictions, we managed to pull off the models and implementation but the deployment to app can certainly be done using Flask.

rantOn

Thank You!