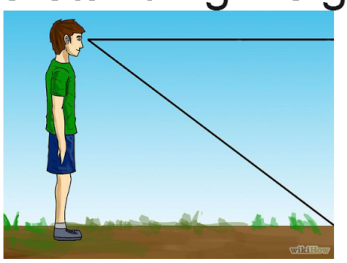
Measurement system

Measurements provide structure and remove the chaos that would result without any congruent method of understanding weight, mass, temperature, etc.



Contents

1 Description

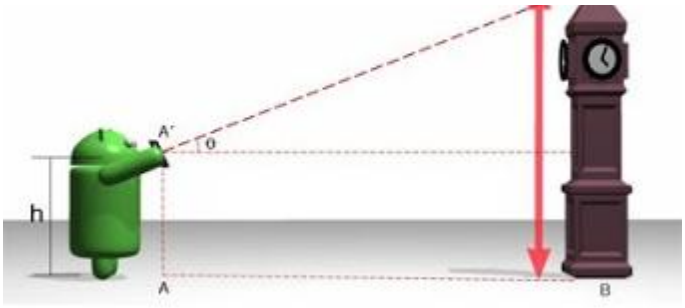
The measurement system receives a image as input, detects the shapes, waits for the user to select an object as reference, and that object as a well-known measure is taken as the base to measure all the others.

2 Objective

My objective with this project is to have an MVP (Minimum Viable Product). I mean, a product as prototype to demonstrate the basic functionality and experiment, in order to gain learning, experience and profits.

3 Applications

You can use an app like this one, to measure things when you do not have a rule, or instead of measuring with the hand, you can use your phone, take a picture, run the app and get the measurement approximately.



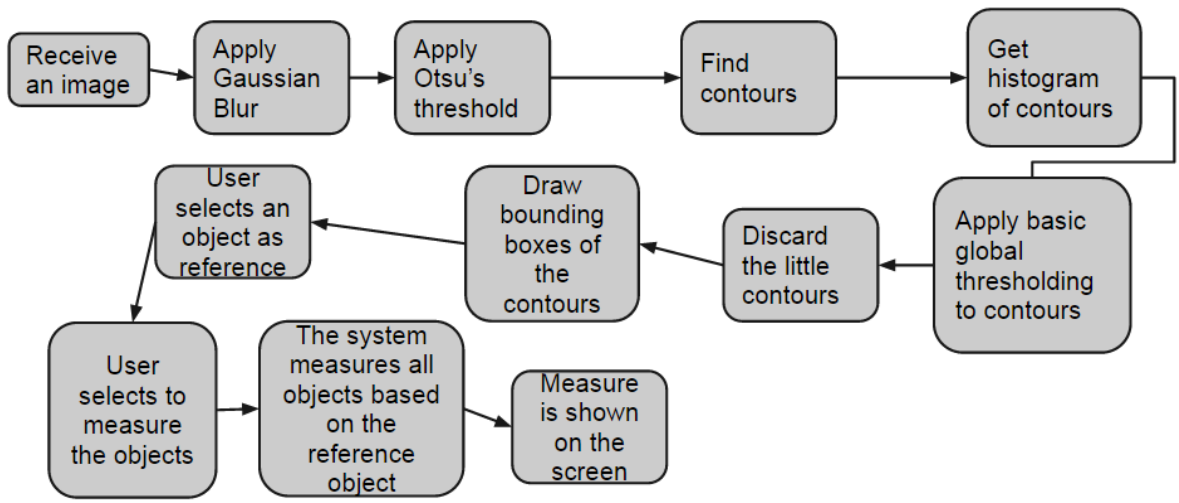
In the industry is also very useful when someone tries to measure something and can not touch physically the object, whether it is too cold or too hot, or just because of the distance between the person and the object.

4 Development

For the construction of this application, the following are the most important parts of it:

1) GUI (Graphical User Interface)
   Created with Pygame
   File picker with tkFileDialog
2) Image processing
   OpenCV

Once being able to build the structure, then the logic or workflow applied is as follows:



The user first needs to select the option to find the objects, and the function basically receives the path of the image to filter it , gets the contours, discard the little ones using a histogram of areas applying a basic global thresholding, gets rid of the bounding box of the whole image, draws the bounding boxes and writes the resulting image.

```
def FindShapes(ip):
    imgPath = ip
    img = cv2.imread(imgPath,0)
    imgCopy = cv2.imread(imgPath)
    height, width = img.shape

    ret, th = filter_image(img)

    cv2.imwrite('RESULT_0.png',th)
    contours, hierarchy = cv2.findContours(th,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

    hist_c,contours_array = get_histogram_contours(contours)

    t = basic_global_thresholding(hist_c)
    contourBoxes = discard_contours(hist_c,contours_array,t)
    contourBoxes.pop(0) #delete the box of the background

    draw_bounding_boxes(imgCopy,contourBoxes,15)

    cv2.imwrite('RESULT.png',imgCopy)

    return contourBoxes
```

I filter using a Gaussian blur, and the algorithm of Otsu's thresholding powered by OpenCV. And for the contours I got good results with the cv2.CHAIN_APPROX_SIMPLE which compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points.

```
def filter_image(img):
    #source: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_
    #Process using Otsu's thresholding after Gaussian filtering
    blur = cv2.GaussianBlur(img,(5,5),0)

    cv2.imwrite('blur.png',blur)
    ret,th = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    return ret,th

def get_contours(th):
    #Find contours based on the threshold obtained
    contours, hierarchy = cv2.findContours(th,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    return contours
```

In order to discard the bounding boxes of little objects which are likely to be noise or confusion of being an object I need to get the histogram of contours. Hence, I first adapt the bounding box to the angle of inclination taking into account the minAreaRect because it finds a rotated rectangle of the minimum area enclosing the input 2D point set. Then, I store the area of that new bounding box, and in another array the contours in order to map them when I need it.

```
def get_histogram_contours(contours):
    hist_contours = []
    contours_array = []
    #I use the area of the bounding box in order to discard them by it later on
    for i in range(len(contours)):

        contour = contours[i]
        rect = cv2.minAreaRect(contour)
        box = cv2.cv.BoxPoints(rect)
        box = np.int0(box)
        side = [
            math.hypot(box[1][0] - box[0][0], box[1][1] - box[0][1]),
            math.hypot(box[3][0] - box[0][0], box[3][1] - box[0][1])
        ]
        area = side[0]*side[1]
        hist_contours.append(float("{0:.2f}".format(area)))
        contours_array.append(box)

    #print "hist_contours: ",hist_contours
    return hist_contours,contours_array
```

Once I have my histogram of contour areas I apply a basic global thresholding, and at the end I get T, where T is my calculated threshold, and I finally return T-T*95, because after a lot of testing I found that this was a good value for the type of images that I used, but I will need to test out more and try out other methods to analyze the results, because this one is not optimal.

```
def basic_global_thresholding(histogram):
    #Basic global thresholding
    t_new = max(histogram)/2.0 #the threshold should be close to the center
    t_old = 0.0
    mean = [0,0]
    T = 0
    while abs(t_old-t_new) > 0.001: #the loop ends until the difference of the old threshold and the new one is minimum

        mean = get_averages(histogram,t_new) #get the average of each side based on the current threshold
        t_old = t_new
        t_new = 0.5*(mean[0]+mean[1]) # get a new threshold by obtaining the average of the means

        T = t_new/2.5

    return T-T*.95 #After testing this kind of threshold gave me good results
```

After getting my threshold value, I only take into account those contours which bounding box area is greater than the threshold.

```
def discard_contours(histogram,contours,T):
    contours_filtered = []
    #Only those greater than the Threshold will be taken into account
    for i in range(len(contours)):
        if histogram[i] > T:
            contours_filtered.append(contours[i])
            #print "average selected: ",histogram[i]
    return contours_filtered
```

In order to get the object as a reference, whenever I detect a click I validate if that click coordinate is inside of one bounding box, and if so, that bounding box is taken as the reference object, borders turn red and the others turn blue. To detect the coordinates I need to scale them, because the original image is of a different size when it is displayed on the screen of my GUI.

```
def getReferenceObject(coord,contourBoxes,scaleW,scaleH):
    print "coord: ",coord
    coord = coord[0]*scaleW,coord[1]*scaleH
    resultImg = cv2.imread("RESULT.png")
    found = False
    index = -1
    counter = 0
    for box in contourBoxes:
        xmin,xmax,ymin,ymax = getLimits(box)
        #print "BOX: ",box
        if xmin < coord[0] and ymin < coord[1] and  coord[0] < xmax and coord[1] < ymax :
            #print "coord in box - "
            found = True
            index = counter
            cv2.drawContours(resultImg,[box],-1,(0,0,255),25)

        else:
            #print "coord NOT IN box -"
            cv2.drawContours(resultImg,[box],-1,(255,0,0),25)

        counter += 1
        print "validation: ",(xmin,ymin)," - ",(xmax,ymax), "coord: ",coord

    cv2.imwrite("RESULT.png",resultImg)
```

To measure just one line, I need 2 points, the distance in pixels of those points, in this case I know for sure the size of my reference object which is a square of 4cm, and I just get the proportion of my line, in this case I prepared it for inches.

```
def MeasureLine(contourBoxes,objectIndex,point1,point2,scaleW,scaleH,unit):
    resultImg = cv2.imread("RESULT.png")

    box = contourBoxes[objectIndex]
    boxDist = math.hypot(box[1][0] - box[0][0], box[1][1] - box[0][1])

    point1 = int(point1[0]*scaleW),int(point1[1]*scaleH)
    point2 = int(point2[0]*scaleW),int(point2[1]*scaleH)

    dist = math.hypot(abs(point1[0]-point2[0]),abs(point1[1]-point2[1]))
    baseMeasure = 4.0#cm ... I already know the measure of my object as reference
    inches = 0.393700787
    distR = (dist*baseMeasure/boxDist)
    label = str(float("{0:.1f}".format(distR*inches)))

    label = label+" in"

    cv2.line(resultImg, (point1), (point2), (0,255,0),10)
    cv2.putText(resultImg,label,point1, cv2.FONT_HERSHEY_SIMPLEX, 2, (0,0,250),5)

    cv2.imwrite("RESULT.png",resultImg)
```

Moreover, for getting the measure of all the bounding boxes the algorithm is the same, based on the measure in pixels of the reference object I got the sizes of the bounding boxes, then get the proportion, and depending of the command I validate if they will be on cm, in, or mm.

```
for i in range(len(contourBoxes)):
    if i != objectIndex or 1 == 1:

        boxDist = []
        boxToMeasure = contourBoxes[i]
        boxDist.append(math.hypot(boxToMeasure[1][0] - boxToMeasure[0][0], boxToMeasure[1][1] - boxToMeasure[0][1]))
        boxDist.append(math.hypot(boxToMeasure[3][0] - boxToMeasure[0][0], boxToMeasure[3][1] - boxToMeasure[0][1]))



        width = boxDist[0]*baseMeasure/dist[0]
        height = boxDist[1]*baseMeasure/dist[1]

        label1 = str(float("{0:.1f}".format(height)))
        label2 = str(float("{0:.1f}".format(width)))

        if unit == "cm":
            label = label1 + " X " + label2+" cm"
        else:
            if unit == "in":
                inches = 0.393700787
                label1 = str(float("{0:.1f}".format(height*inches)))
                label2 = str(float("{0:.1f}".format(width*inches)))

            if unit == "mm":
                mm = 10
                label1 = str(float("{0:.1f}".format(height*mm)))
                label2 = str(float("{0:.1f}".format(width*mm)))

            label = label1 + " X "+label2+" "+unit

        labelSize = len(label)*15
        xmin,xmax,ymin,ymax = getLimits(boxToMeasure)
        cv2.putText(resultImg,label, (xmin + int(abs(xmax-xmin)/2)-labelSize,ymin+int(abs(ymax-ymin)/2)), cv2.FONT_HERSHEY_SIMPLEX, 2, (0,13\
55),5)
```

Finally, the GUI.py just contains the visual part to interact with the user and consumes the MeasurementSystem.py

It scales the image before displaying it and displays the visual part using pygame.

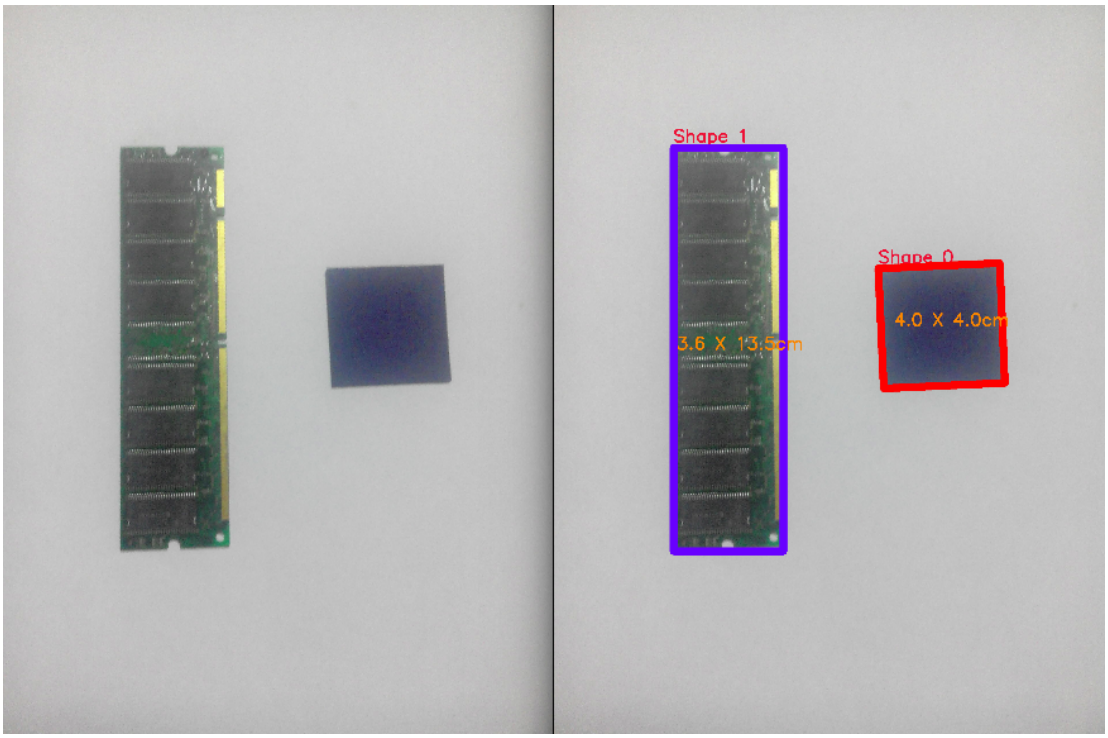
```
import sys,pygame
from MeasurementSystem import *
from tkFileDialog import *

def loadFile():
    fname = ""
    try:
        fname = askopenfilename(filetypes=(("All files", "*.*"),
                                           ("PNG", "*.png")))

    except ValueError:
        print "Please select an image. Try again..."

    return fname

pygame.init()
imgPath = loadFile()
image = pygame.image.load(imgPath)

imagerect = image.get_rect()
realWidth, realHeight = imagerect.size

scaleWidth, scaleHeight = 400,600
image = pygame.transform.scale(image,(scaleWidth,scaleHeight))

imagerect = image.get_rect()
width, height = imagerect.size

panel = 90
topPanel = 40

size = ((width+panel),(height+topPanel))
screen = pygame.display.set_mode(size)

pygame.display.set_caption("Measurement System")
font = pygame.font.Font(None, 20)
```

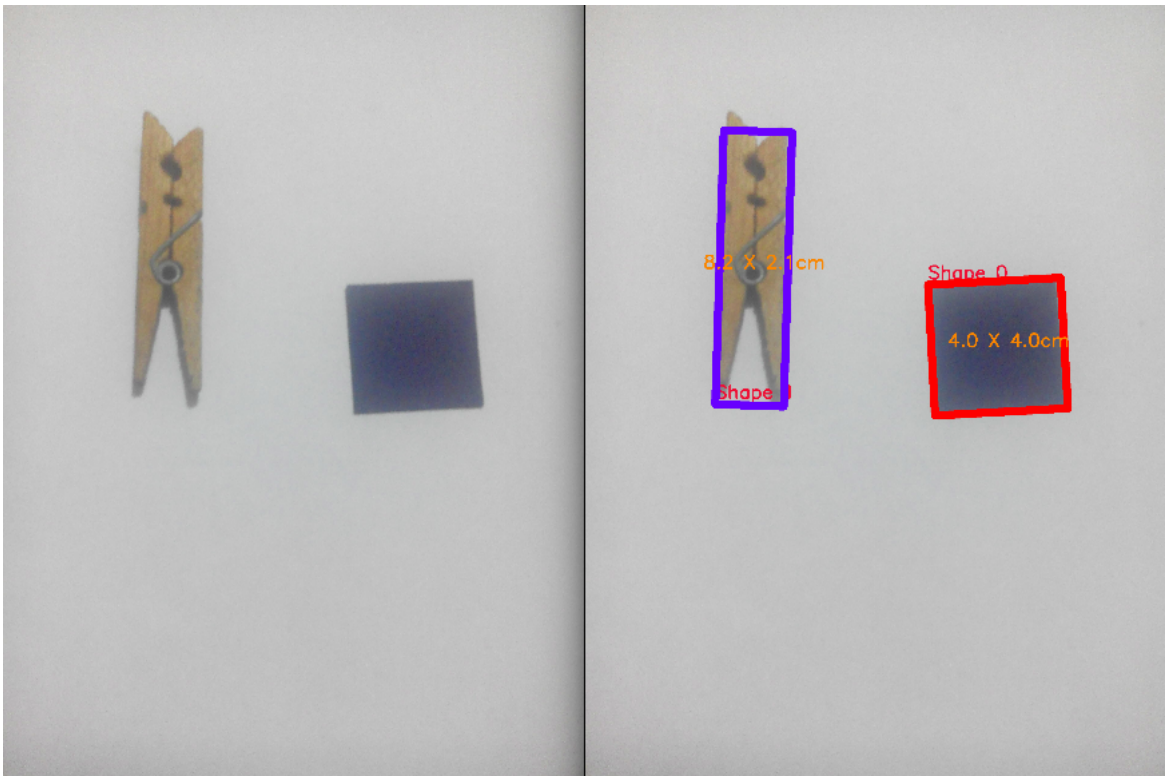
I define all the buttons and use variables as flags to determine the flow and actions taken. Also, it uses a cycle which is aware of any click so it can react executing the functions depending on the coordinates.

```
pygame.display.set_caption("Measurement System")
font = pygame.font.Font(None, 20)

#Options to display
btnFindObjects = font.render('Find Objects',1,(0,0,250))
btnMeasureCm = font.render('Measure (cm)',1,(0,255,0))
btnMeasureIn = font.render('Measure (in)',1,(0,255,0))
btnMeasureMm = font.render('Measure (mm)',1,(0,255,0))
btnMeasureLine = font.render('Measure (line)',1,(0,255,0))

found = False
isReadyToMeasure = False
message = font.render('',1,(255,0,0))
isObject = False
objectIndex = int()
isWaiting2P = False

scaleW, scaleH = int(),int()

point1, point2 = (0,0)
counter = 0

global contourBoxes

while 1:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

        if event.type == pygame.MOUSEBUTTONDOWN:
            cordx, cordy  = pygame.mouse.get_pos()
            if 0 < cordx < panel and topPanel < cordy < 25+topPanel: #Find Shapes
                print "btnFindObjects"
                print "Select an object as reference"
                message = font.render('Select an object as reference',1,(255,0,0))
                found = True

                contourBoxes = FindShapes(imgPath)
```

5 Tests

In order to test the behaviour of my application I used a variety of objects:





My conclusion with this testing is that shadows do affect in the measurement, and the angle of the camera affects a little bit. It will work better if it is placed at 0 °, or parallel to the surface, because my program does not retrieve the angle of perspective. There is little error of between 0.5 cm and 1 cm.

6 Demo video:

https://www.youtube.com/watch?v=GBAVYRhBBdU&feature=youtu.be

7 Future work

1. Identify perspective of the image and calculate the distance depending on the angle of the camera.
2. Implement convex hull recognition.
3. Add a preprocessing method to get rid of the shadows without affecting the border recognition.
4. Indicate any object as reference with its size
5. Enhance the user experience with a better design
6. When it draws, it opens the image, draw and then write and reopen the image every time. I will need to open the image and then put a canvas on it to draw anything without writing and opening the image.
7. Add button to “save the image as…”
8. When I select an object I have to click inside the invisible bounding box formed by xmin,xmax - ymin,ymax. This creates a bug when an object is at 45°for example.
9. Export it to app mobile (Android, Windows Phone, IOS)
10. Adapt it for Google glasses and Microsoft Hololens

8 References

- Intel® RealSense™. (2015). Intel RealSense Snapshot - Photo Measurement . Mayo 03, 2015, de Youtube Sitio web: https://www.youtube.com/watch?v=zCVoAggGvVY
- Stefano Benamati. (2012). OpenCV] Object measurement. Mayo 03, 2015 de Youtube Sitio web: https://www.youtube.com/watch?v=bcswZLwhTUI
- Abid Rahman K . (2012). Contours - 1 : Getting Started. Mayo 03, 2015, de blogspot Sitio web: http://opencvpython.blogspot.mx/2012/06/hi-this-article-is-tutorial-which-try.html
- Alexander Mordvintsev & Abid K. (2014). Contour Features. Mayo 03, 2015, de readthedocs Sitio web: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc/py_contours/py_contour_features/py_contour_features.html
- Anónimo. (2015). Perspective transform with OpenCV. Mayo 03, 2015, de tutorialsplay Sitio web: https://tutorialsplay.com/opencv/2015/01/10/perspective-transform-with-opencv