Shape detection

real-time basic geometric shapes

Md.Maniruzzaman & J. Cole & Zafarullah
Digital Image Processing

Contact Information:
University of Tartu
Robotic and Computer Engineering
Nooruse 1, 50411 Tartu, Estonia

Phone: 506 9803
Email: shb@ut.ee

Abstract
WRITE A REAL ABSTRACT HERE

Introduction

Vision is one of our most important senses through which we can perceive and identify things of different shapes and texture and then respond accordingly. When this comes to machine vision, this is highly based on computer-based image processing and shape detection becomes an important task as its at the base of a variety of applications with different purposes.

Main Objectives

As our project for the course, we want to build a program that will be able to recognize the most basic regular object shapes, such as triangle, rectangle, square, pentagon and circle. To do this, we will use Python programming language v. 3 with the resource of the library OpenCV that is structured to specifically handle the processing of images and videos. The intention is to make the use of the built-in camera of the device where the program will run and through this detect the shapes of objects in real time. The program able to detect the same shapes even at different distances and multiple-shapes at the same time. The main insight, is to detect the edges of the target and by this classify the correct shape.

Materials and Methods

1. Resize: First we resize the image with 300 frame and find out the ratio of the changed image.
2. Gray-scale: later we convert the image into gray scale
3. Blurring: We blur the image using a Gaussian smoothing in order to reduce the details and the image noise
4. Thresholding: The next process is thresholding in which a certain threshold value of gray level is chosen to separate the object from its background. We use a binary threshold setting the frame in black and white according to a threshold of 120.
5. Find Contours: Later we find out contours in the thresholded image and initialize the shape detector process. We use RETR TREE as a parameter to be able to detect even the inner shapes in the contour hierarchy. Furthermore we use CHAIN APPROX SIMPLE which is a function that permits us to recognize a shape by only the end points of horizontal, vertical and diagonal segments.
6. Calculate Moments: We find the center of each polygon in order to detect the final shape.
7. Find Edge: We calculate the area or the perimeter of a polygon and we find the approximated edges through the approxPolyDP function which is an implementation of the Douglas-Peucker algorithm.
8. Draw shape: We draw the shape of the detected image in different color.

The process is visible through a flowchart:

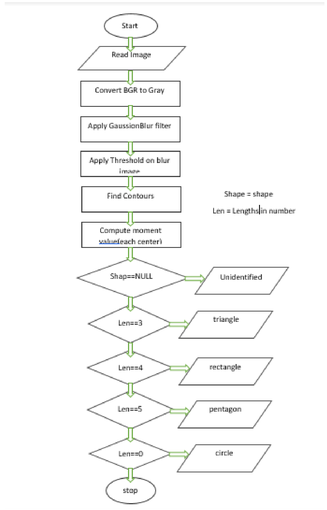

Figure 1: Flowchart

Mathematical Section

Centroid of a shape: Centroid of a shape is the arithmetic mean(i.e. the average) of all the points in the shape. Suppose a image consist of n distinct points $x_1.....x_n$ then the centroid will be

$$c = \frac{1}{n}\sum_{i=0}^{n} x_i$$

Figure 2: Equation for center point

Centroid is simply the weighted average of all the pixel of a shape. **Image Moment**: Image Moment is a particular weighted average of image pixel intensities using this we can find the specific property of an image just like centroid, area, radius etc.We have applied the following formula to find the center in our application.

$$C_x = \frac{M_{10}}{M_{00}} * ratio \;\&\; C_y = \frac{M_{10}}{M_{00}} * ratio$$

Figure 3: Applied Equation

C_x is the x coordinate and C_y is the y coordinate of the centroid and M denotes the Moment and
$ratio = \frac{shape}{resizedshape}$

Testing and Result:

We have tested the application on different laptops with Linux and Windows operating systems. The application has also been tested with fixed, moving and static backgrounds. For the statistic background, we drew the desired shapes filling them with black color on a white paper. The result is visible on Figure 2. The application on the static background seems to be reliable and accurate ad shapes are detected correctly. The only difficulties are found, sometimes, in distinguishing the square from the rectangle polygon. Furthermore, it seems that repetitively an external square/rectangle is detected as a frame of the pentagon, circle and triangle shapes.

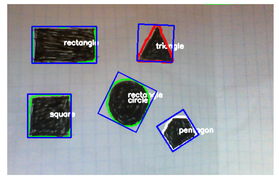

Figure 4: Shape detection applied on a static background

For the fixed background test, we used a black t-shirt weaved with a pattern of independent white symbols of an undefined shape. The result is visible on Figure 3. We can notice that the program recognized almost all the shapes present on the image. However, what is interesting is that although the shapes were in principle all the same, due to the foldings of the material and the lighter or darker light posing on them, the shapes have been detected by the application as different overall. The most recognized shapes have been the green ones, namely the color that has been assigned to undefined polygons. There are then some detected pentagons in white mainly in the lower part of the picture and blue squares and rectangles in the upper part. To complete, also few red triangles have been detected.

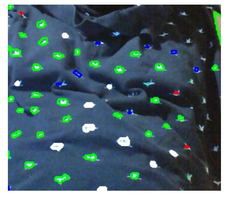

Figure 5: Shape detection applied on a fixed background

For the moving background test we moved a pen randomly in front of camera. An example of the result is reported on Figure 4. As we can notice, this last test environment is where our program drops more in performance as the application will find it complicated to stack the object from the background and draw its contours.

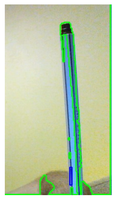

Figure 6: Shape detection applied on a moving background

Conclusions

The main aim of the project was to detect and differentiate basic geometric shapes (circle, square, triangle, pentagon and rectangle) in a real-time by merely employing computer vision technique. We tested our application into three different environments and we found it significantly trustworthy in situations where shapes are filled in solid color on a static background.

Forthcoming Research

For future research, it would be interesting to approach the task using a different methodology. For example, the image can be converted into HSL and use the lightness component to produce a good color separation of the image and its background. A different thresholding could also be applied, taking advantage of the Otsu's binarization. Furthermore, a crucial step that we didn't manage to implement in our project it would be apply a filling to thresholded image, in this way discontinuities and irregularities within the image of the shapes would be smoothed resulting probably in a better performance of the application.

Acknowledgements

This project is successfully completed from advice, assistance and support of our respected project Advisor Mr. G.Anbarjafari(Shahab) professor of Robotic and Computer Engineering, University of Tartu. We desire to express our warm sense of gratefulness for his vigorous supervision, uniform advise, encouraging behavior and the constructive, fruitful and insightful exchanges of ideas that assist us throughout this study.