Contours

This document lists how to find and draw contours, and some of their properties.

1. Finding Contours

   There is a basic constraint on the type of images on which we can detect contours, namely **Binary Images**, in other words, images whose pixels have only 2 possible values. In our case, we use the threshold function(which in turn works on a grayscale image) to make force all the pixels in our image to either have a value of 0 or 255. Therefore there are 3 steps to finding contours, to consider an example,

   a. Convert to grayscale →*gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)*
      i. img – Source Image
      ii. cv2.COLOR_BGR2GRAY – convert from BGR format to grayscale
      iii. gray – Destination Image
   b. Convert to binary image → *ret,thresh = cv2.threshold(gray,127,255, cv2.THRESH_BINARY)*
      i. gray – Source Image
      ii. 127 – Threshold Value

iii. 255 – Value to set to
iv. cv2.THRESH_BINARY – Type of Thresholding

c. Find Countours → **contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)**
   i. thresh – Source Binary Image
   ii. contours – list of contours that are found
   iii. cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE – Methods of finding contours

2. Drawing Contours

   cv2.drawContours(source, contours , index , colour , thickness)

   This is the command we use to draw contours. Keep in mind that we draw the contours on a BGR image. The different parameters are as follows:

   - source – Source Image
   - contours – List of contours to draw
   - index – Index number of contour to draw, if set as *-1*, then will draw all contours in the list
   - colour – The colour in which to draw the contours
   - thickness - The thickness of the contours drawn

Example of a script to draw contours:

gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

ret,thresh = cv2.threshold(gray,127,255,0)

contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

cv2.drawContours(img,contours,-1,(0,255,0),3)

3. Properties of contours
   The following are ways to find out about the properties of the i^{th} contour in the list *contours*
   a. Area = **cv2.contourArea(contours[i])**
   b. Perimeter = **cv2.arcLength(contours[i],True)**
   c. Centroid → **M = cv2.moments(contours[i])**
   cx = int(M['m10']/M['m00'])
   cy = int(M['m01']/M['m00'])
   centroid = cx,cy