## findContours

This is a function, we can find a contours of a binary image. In OpenCV, we have:

C++: void findContours(InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset=Point())

**Parameters:**

* Image: an 8-bit single-channel image.
* Contours: detected contours. Each contour is stored as a vector of points.
* Hierarchy: Optional output vector, containing information about the image topology. It has as many elements as the number of contours. For each i-th contour contours[i] , the elements hierarchy[i][0] , hiearchy[i][1] , hiearchy[i][2] , and hiearchy[i][3] are set to 0-based indices in contours of the next and previous contours at the same hierarchical level, the first child contour and the parent contour, respectively. If for the contour i there are no next, previous, parent, or nested contours, the corresponding elements of hierarchy[i] will be negative.
* Mode:

- CV\_RETR\_EXTERNAL retrieves only the extreme outer contours. It sets hierarchy[i][2]=hierarchy[i][3]=-1 for all the contours.

- CV\_RETR\_LIST retrieves all of the contours without establishing any hierarchical relationships.

- CV\_RETR\_CCOMP retrieves all of the contours and organizes them into a two-l evel hierarchy. At the top level, there are external boundaries of the components. At the second level, there are boundaries of the holes. If there is another contour inside a hole of a connected component, it is still put at the top level.

- CV\_RETR\_TREE retrieves all of the contours and reconstructs a full hierarchy of nested contours.

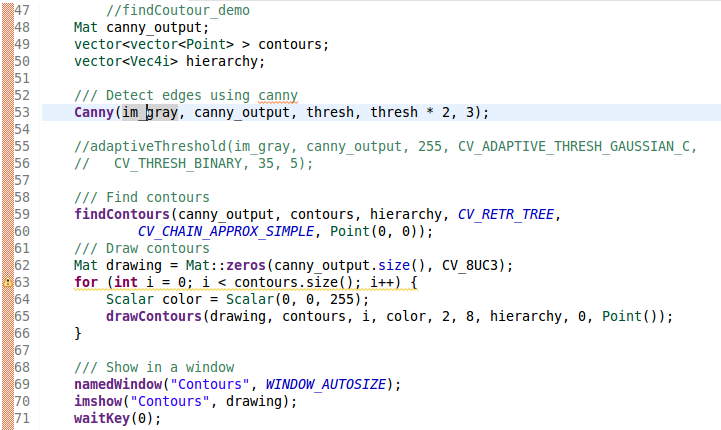
* Method:

- CV\_CHAIN\_APPROX\_NONE stores absolutely all the contour points. That is, any 2 subsequent points (x1,y1) and (x2,y2) of the contour will be either horizontal, vertical or diagonal neighbors, that is, max(abs(x1-x2),abs(y2-y1))==1.

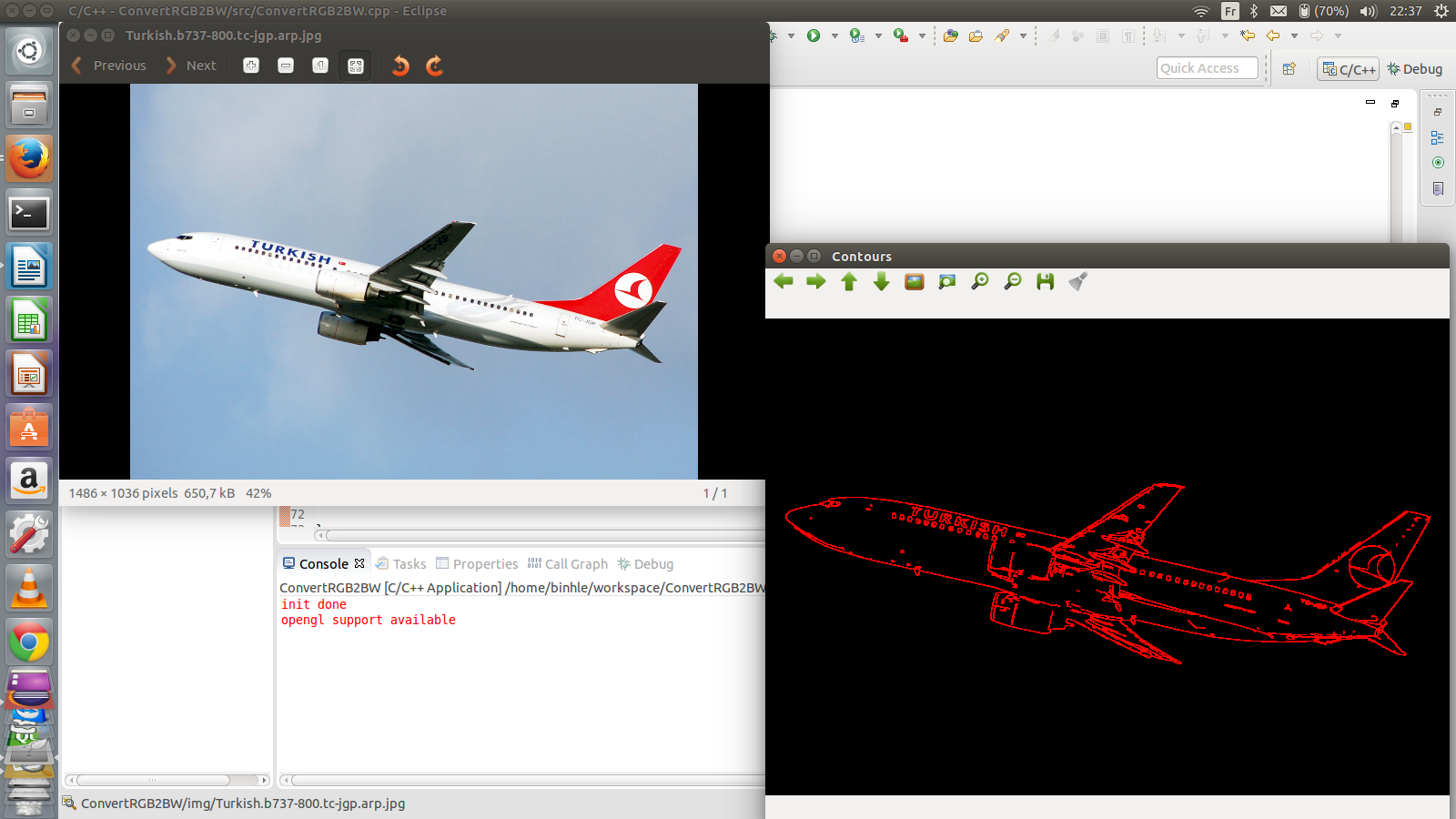
- CV\_CHAIN\_APPROX\_SIMPLE compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points.

* Offset: which every contour point is shifted.

**Demo:**



**Result:**



# **Convex Hull**

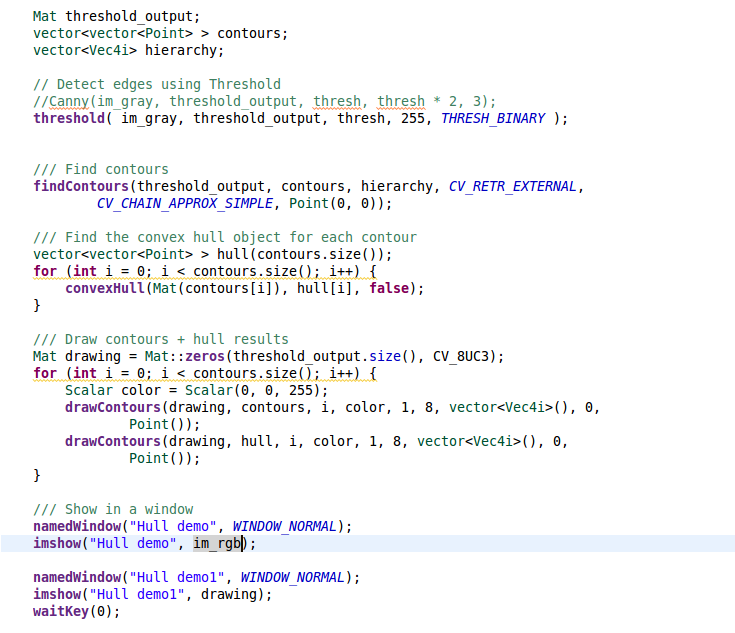
Finds the convex hull of a point set.

C++: void convexHull(InputArray points, OutputArray hull, bool clockwise=false)

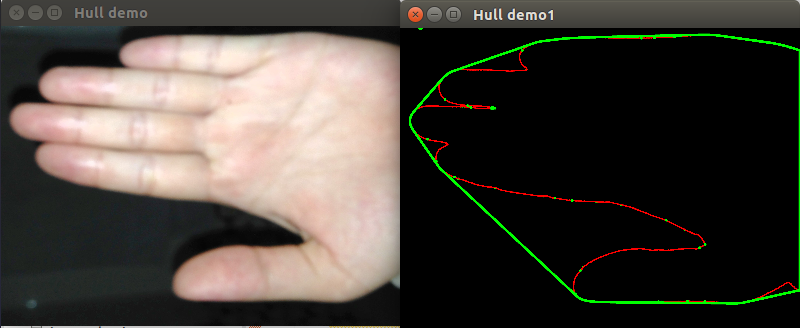
**Parameters:**

* Points: Input 2D point set, stored in std::vector or Mat.
* Hull: Output convex hull
* Clockwise: Orientation flag. If it is true, the output convex hull is oriented clockwise. Otherwise, it is oriented counter-clockwise.

**Demo:**



**Result:**

****

**convexitydefects**

Finds the convexity defects of a contour.

C++: void convexityDefects(InputArray contour, InputArray convexhull, OutputArray convexityDefects)

**Parameters:**

* Contour: Input contour.
* Convexhull: Convex hull obtained using convexHull() that should contain indices of the contour points that make the hull.
* ConvexityDefects: The output vector of convexity defects. In C++ and the new Python/Java interface each convexity defect is represented as 4-element integer vector.