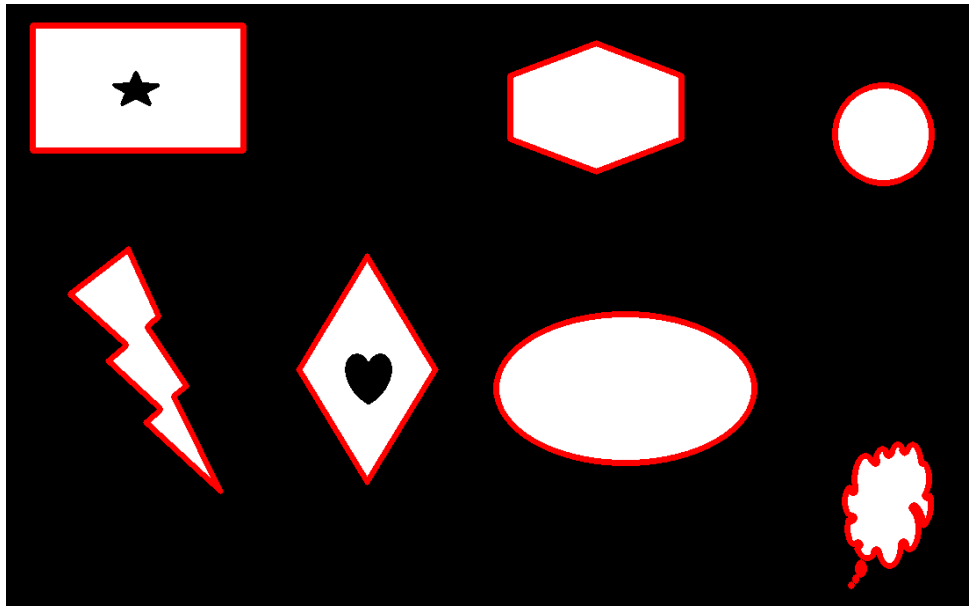
Contours

Outline

- Contour
  - Find contour
  - Draw contour
  - Contour properties
    - Area
    - Perimeter
    - Bounding Box
    - Circle fitting
    - Ellipse fitting

Contours

- Contour is the curve along the boundary of the object
- Useful for shape analysis, object detection and recognition
- Before finding contours, apply threshold or Canny edge detection
- OpenCV finds contours of white object on black background
- (contours, hierarchy) = **cv2.findContours**(image, mode, method[, contours[, hierarchy[, offset]]])



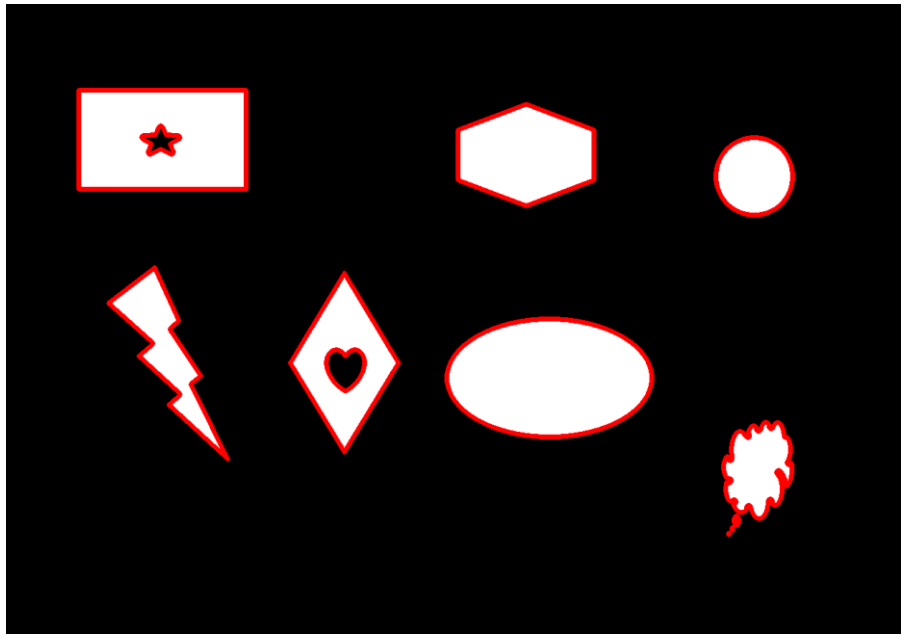
cv2.findContours

(contours, hierarchy) = cv2.findContours(image, mode, method[, contours[, hierarchy[, offset]]])

- **image** - input image (8-bit single-channel). Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as binary . You can use compare, inRange, threshold , adaptiveThreshold, Canny, and others to create a binary image out of a grayscale or color one.
- **contours** - Detected contours. Each contour is stored as a vector of points.
- **hierarchy** - Optional output vector containing information about the image topology.
- **mode** - Contour retrieval mode, (RETR_EXTERNAL, RETR_LIST, RETR_CCOMP, RETR_TREE)
- **method** - Contour approximation method. (CHAIN_APPROX_NONE, CHAIN_APPROX_SIMPLE, CHAIN_APPROX_TC89_L1 etc)
- **offset** - Optional offset by which every contour point is shifted. This is useful if the contours are extracted from the image ROI and then they should be analyzed in the whole image context.

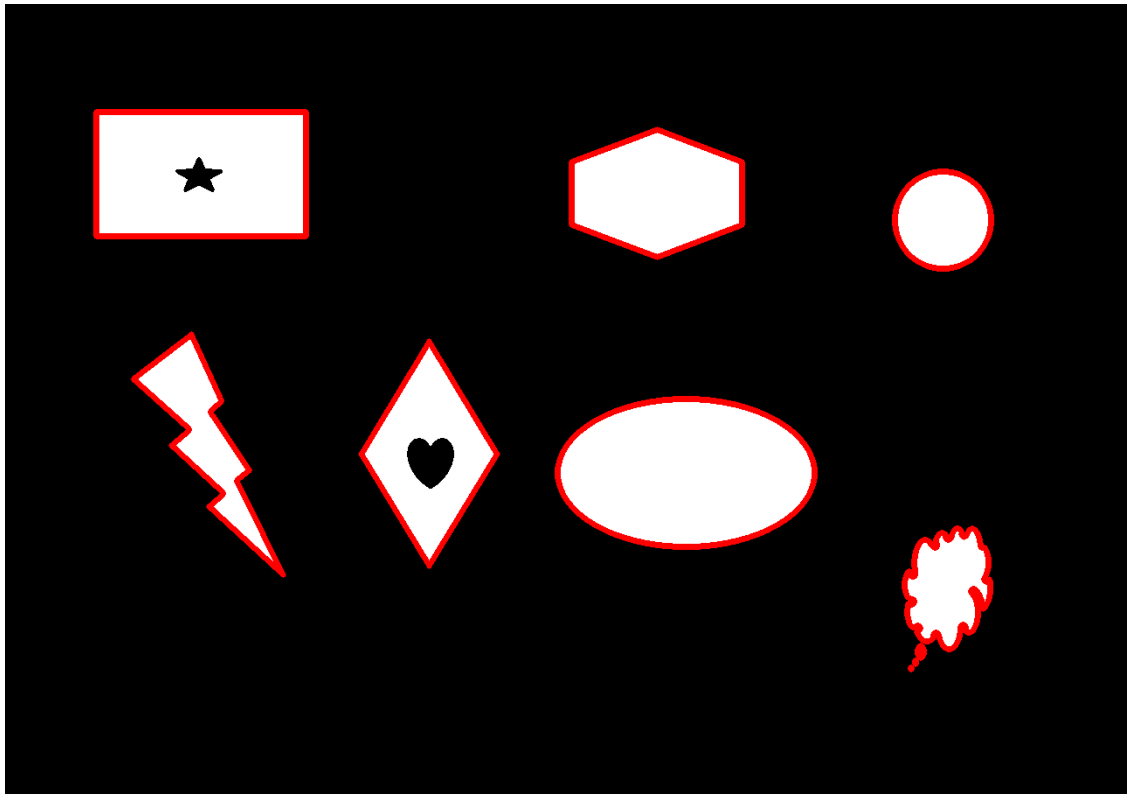
Contours - OpenCV

```
img = cv2.imread(path)
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
thres, imgB = cv2.threshold(imgGray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
# find all contours in the image and draw them
imgCopy = img.copy()
(cnts, _) = cv2.findContours(imgB, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(imgCopy, cnts, -1, (0, 0, 255), 5)
plt.imshow(cv2.cvtColor(imgCopy, cv2.COLOR_BGR2RGB)), plt.title('Contours')
plt.show()
```



Contours - OpenCV

```
# find only external contours in the image and draw them
(cnts, _) = cv2.findContours(imgB, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(imgCopy, cnts, -1, (0, 0, 255), 5)
```



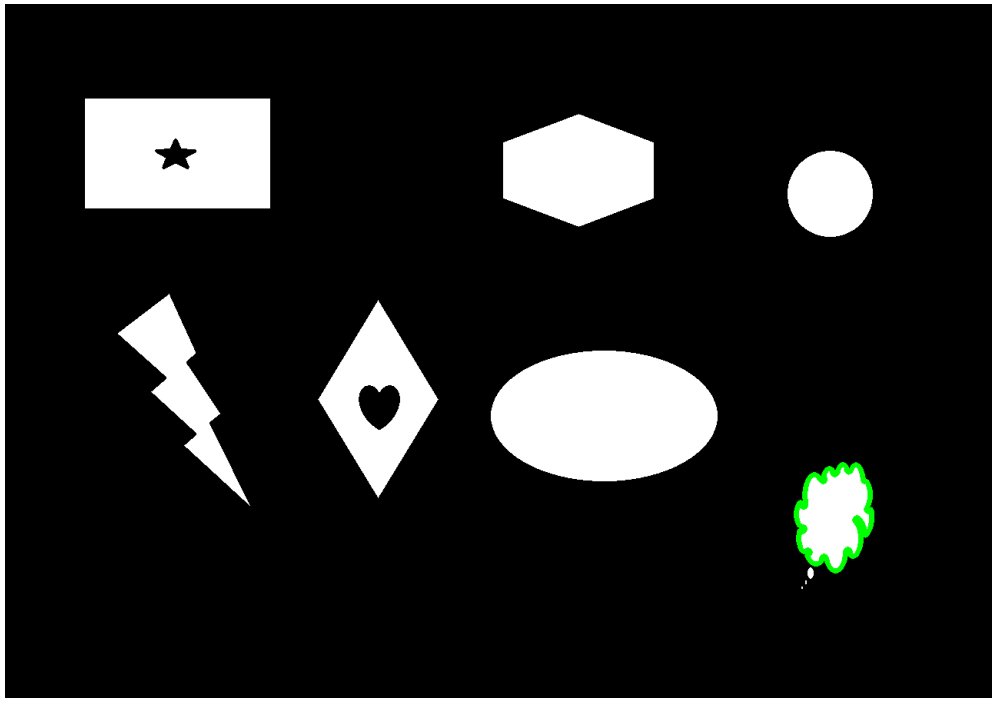
cv2.drawContours

```
image=cv2.drawContours(image, contours, contourIdx, color[,
                       thickness[, lineType[, hierarchy[,
                       maxLevel[, offset]]]]])
```

- **image**: This is the input RGB image on which you want to draw the contour.
- **contours**: Indicates the **contours** obtained from the **findContours()** function.
- **contourIdx**: The pixel coordinates of the contour points are listed in the obtained contours. Using this argument, you can specify the index position from this list, indicating exactly which contour point you want to draw. Providing a negative value will draw all the contour points.
- **color**: This indicates the color of the contour points you want to draw. We are drawing the points in green.
- **thickness**: This is the thickness of contour points.

Contours - OpenCV

```
# draw an individual contour
imgCopy = img.copy()
# draw 4th countour
cv2.drawContours(imgCopy, cnts, 3, (0, 255, 0), 5)
```



Contour properties

- Area
- Perimeter
- Bounding Box
- Circle fitting
- Ellipse fitting
- And more

Contour – area and perimeter

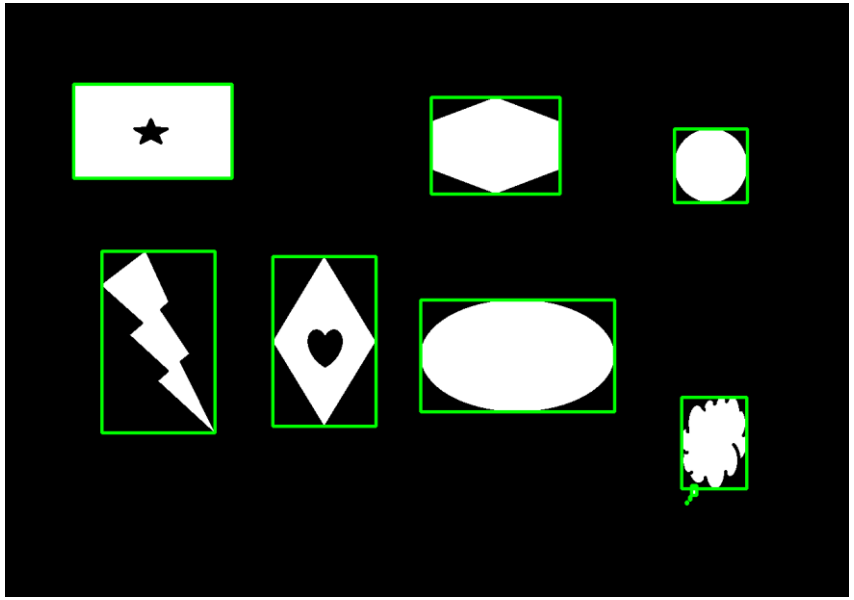
- Area of a contour - the number of pixels inside the contour.
- Perimeter /arc length - the length of the contour

```
(cnts, _) = cv2.findContours(imgB, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for i in range(len(cnts)):
    area = cv2.contourArea(cnts[i])
    perimeter = cv2.arcLength(cnts[i], True)
    print("Countour #", i, ": perimeter =", perimeter, " area =", area)
```

Contour – bounding rectangle

- Bounding box - an **upright** rectangle that bounds a contour.

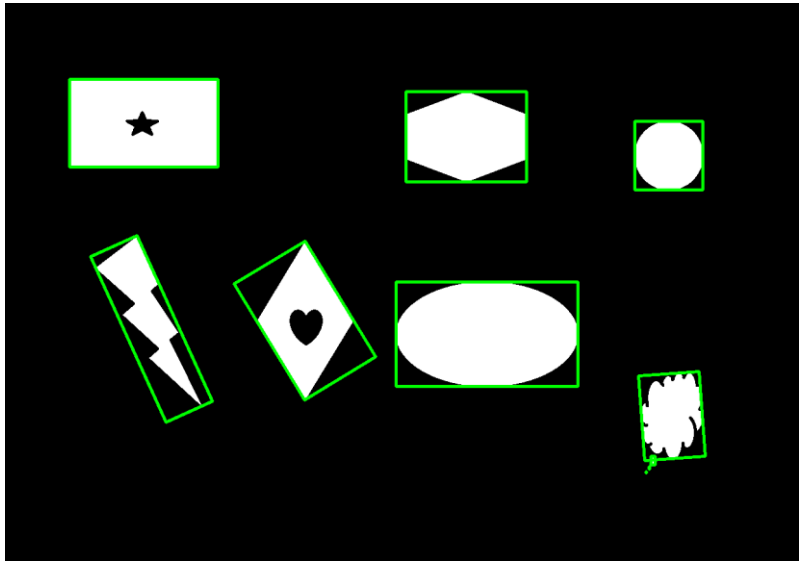
```
(cnts, _) = cv2.findContours(imgB, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
imgCopy = img.copy()
for i in range(len(cnts)):
    (x, y, w, h) = cv2.boundingRect(cnts[i])
    cv2.rectangle(imgCopy, (x, y), (x + w, y + h), (0, 255, 0), 4)
cv2.imwrite(os.path.join(folder, "BoundingBox.png"), imgCopy)
```



Bounding box with the minimum area

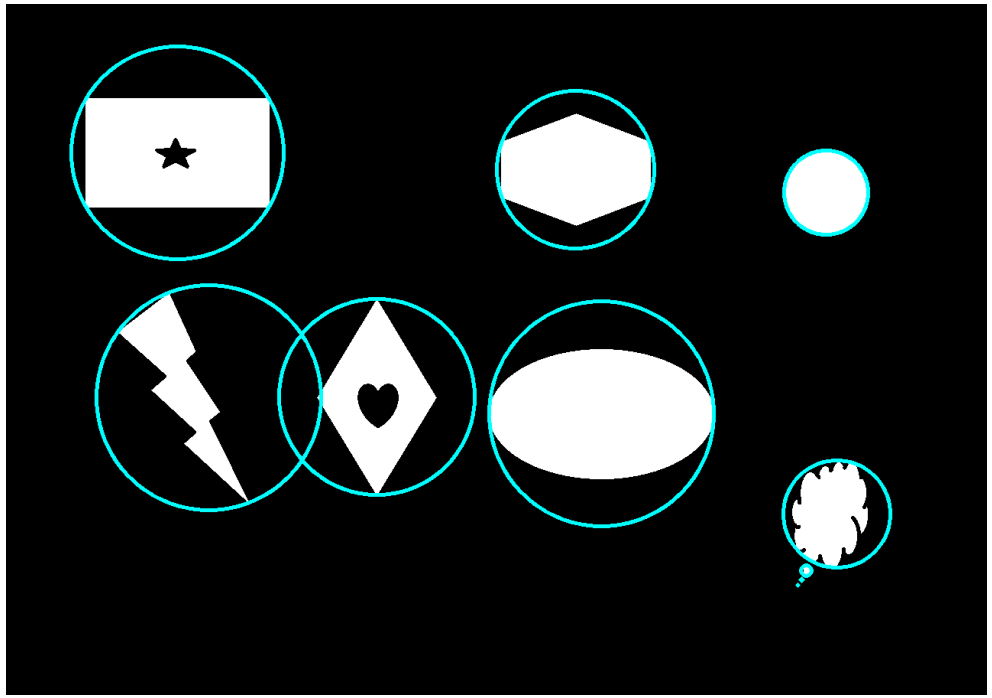
- A rotated rectangle - the bounding box with the minimum area

```
(cnts, _) = cv2.findContours(imgB, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
imgCopy = img.copy()
for i in range(len(cnts)):
    box = cv2.minAreaRect(cnts[i])
    boxPts = np.int0(cv2.boxPoints(box))
    cv2.drawContours(imgCopy, [boxPts],
                     -1, (0, 255, 0), 4)
```



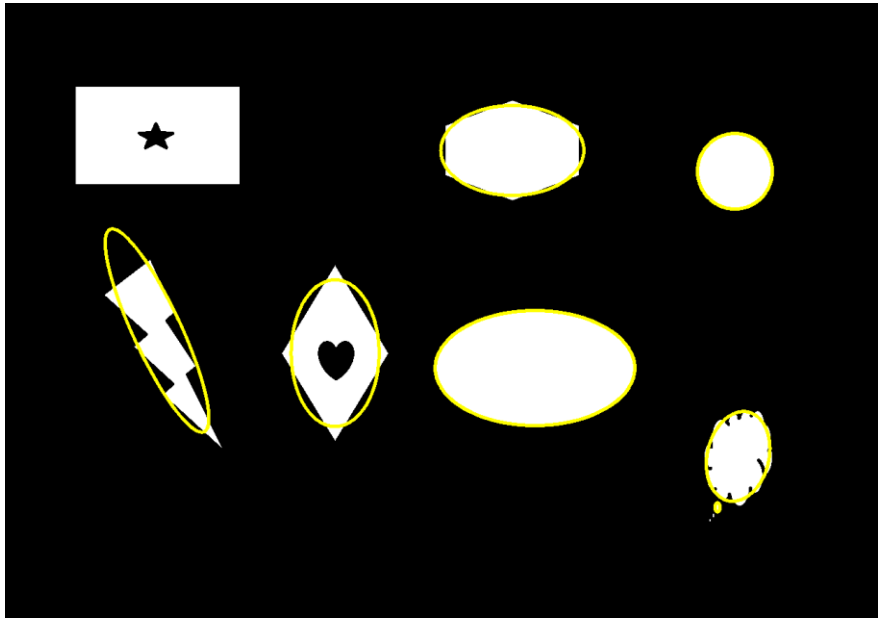
Fit a circle to a contour

```
(cnts, _) = cv2.findContours(imgB, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
imgCopy = img.copy()
for i in range(len(cnts)):
    ((x, y), radius) = cv2.minEnclosingCircle(cnts[i])
    cv2.circle(imgCopy, (int(x), int(y)), int(radius), (255, 255, 0), 4)
cv2.imwrite(os.path.join(folder, "BoundingCircle.png"), imgCopy)
```



Fit an ellipse to a contour

```
(cnts, _) = cv2.findContours(imgB, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
imgCopy = img.copy()
for i in range(len(cnts)):
    if (len(cnts[i]) >= 5):  # must have at least five points
        ellipse = cv2.fitEllipse(cnts[i])
        cv2.ellipse(imgCopy, ellipse, (0, 255, 255), 4)
cv2.imwrite(os.path.join(folder, "FittedEllipse.png"), imgCopy)
```



Summary

- Contour
  - Find contour
  - Draw contour
  - Contour properties
    - Area
    - Perimeter
    - Bounding Box
    - Circle fitting
    - Ellipse fitting

Practice

Lab_Countours.py