Weekly Report 4

Resourses

* [A Stack Overflow question](https://stackoverflow.com/questions/20185895/opencv-object-detection-bounding-box) that shows how to draw a bounding box around an object detected by SVM using SURF features.
* [Another Stack Overflow question](https://stackoverflow.com/questions/59651764/python-opencv-how-to-calculate-angle-of-object-wrt-a-sample) that shows how to calculate the angle of an object with respect to a sample image using OpenCV.
* [An OpenCV forum post](https://forum.opencv.org/t/rotate-the-bounding-box-of-object-detection-and-crop-it/7949) that shows how to rotate the bounding box of an object detection and crop it using OpenCV.
* [A Stack Overflow answer](https://stackoverflow.com/questions/61718596/how-to-rotate-bounding-box-in-open-cv-and-crop-it-python) that shows how to rotate a bounding box in OpenCV and crop it using Python.
* [A Stack Overflow answer](https://stackoverflow.com/questions/64572137/how-to-get-the-rotation-of-and-object-given-its-axis-aligned-bounding-box-coordi) that shows how to get the rotation of an object given its axis-aligned bounding box coordinates using OpenCV.

functions in OpenCV Python that can be used to find the orientation of an object, along with some example code:

* cv2.findContours: This function finds contours in a binary image. Contours are curves that join all the continuous points along the boundary of an object. [They can be used to find the shape and orientation of an object by fitting a minimum area rectangle or an ellipse to them](https://stackoverflow.com/questions/58632469/how-to-find-the-orientation-of-an-object-shape-python-opencv)[1](https://stackoverflow.com/questions/58632469/how-to-find-the-orientation-of-an-object-shape-python-opencv)[2](https://automaticaddison.com/how-to-determine-the-orientation-of-an-object-using-opencv/).

# Read image as grayscale

img = cv2.imread("object.jpg", cv2.IMREAD\_GRAYSCALE)

# Threshold the image

\_, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)

# Find contours

contours, hierarchy = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Draw contours on the original image

img = cv2.cvtColor(img, cv2.COLOR\_GRAY2BGR)

cv2.drawContours(img, contours, -1, (0, 255, 0), 2)

# Fit a minimum area rectangle to the first contour

rect = cv2.minAreaRect(contours[0])

box = cv2.boxPoints(rect)

box = np.int0(box)

# Draw the rectangle on the image

cv2.drawContours(img, [box], 0, (0, 0, 255), 2)

# Get the angle of the rectangle

angle = rect[-1]

print("Angle:", angle)

* cv2.matchShapes: This function compares two shapes or contours and returns a metric showing the similarity. [It can be used to find the orientation of an object by matching it with a reference shape and computing the rotation angle](https://stackoverflow.com/questions/58632469/how-to-find-the-orientation-of-an-object-shape-python-opencv)[3](https://pyimagesearch.com/2021/01/20/opencv-rotate-image/).

# Read the reference image as grayscale

ref = cv2.imread("ref.jpg", cv2.IMREAD\_GRAYSCALE)

# Threshold the reference image

\_, ref\_thresh = cv2.threshold(ref, 127, 255, cv2.THRESH\_BINARY)

# Find the contour of the reference image

ref\_contour, \_ = cv2.findContours(ref\_thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Read the test image as grayscale

test = cv2.imread("test.jpg", cv2.IMREAD\_GRAYSCALE)

# Threshold the test image

\_, test\_thresh = cv2.threshold(test, 127, 255, cv2.THRESH\_BINARY)

# Find the contour of the test image

test\_contour, \_ = cv2.findContours(test\_thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Match the test contour with the reference contour

ret = cv2.matchShapes(ref\_contour[0], test\_contour[0], cv2.CONTOURS\_MATCH\_I2, 0.0)

print("Match:", ret)

# Get the moments of the test contour

M = cv2.moments(test\_contour[0])

# Get the orientation angle of the test contour

angle = 0.5 \* np.arctan2(2 \* M["mu11"], M["mu20"] - M["mu02"])

angle = np.rad2deg(angle)

print("Angle:", angle)

* cv2.phaseCorrelate: This function estimates the rotation angle and the translation between two images using phase correlation. [It can be used to find the orientation of an object by comparing it with a reference image and computing the rotation angle](https://stackoverflow.com/questions/58632469/how-to-find-the-orientation-of-an-object-shape-python-opencv)[4](https://www.pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/).

# Read the reference image as grayscale

ref = cv2.imread("ref.jpg", cv2.IMREAD\_GRAYSCALE)

# Read the test image as grayscale

test = cv2.imread("test.jpg", cv2.IMREAD\_GRAYSCALE)

# Apply a Gaussian blur to both images

ref = cv2.GaussianBlur(ref, (3, 3), 0)

test = cv2.GaussianBlur(test, (3, 3), 0)

# Convert both images to float32

ref = ref.astype(np.float32)

test = test.astype(np.float32)

# Compute the phase correlation between the images

response, (dx, dy), angle = cv2.phaseCorrelate(ref, test)

print("Response:", response)

print("Translation:", dx, dy)

print("Angle:", angle)

The imutils.rotate\_bound function is a convenient way to rotate an image by any angle in Python using OpenCV. However, by default, it uses the center of the image as the rotation point. If you want to use a vertex of a rectangle as the reference point, you need to modify the function slightly.

Here are the steps to do that:

* First, you need to get the coordinates of the vertex you want to use as the rotation point. You can use cv2.boundingRect to get the top-left and bottom-right coordinates of the rectangle, and then choose one of them as the vertex.
* Next, you need to create a rotation matrix using cv2.getRotationMatrix2D. You can specify the vertex as the center, and the angle you want to rotate by in degrees.
* Then, you need to adjust the rotation matrix to account for the translation of the vertex. You can use np.dot to multiply the rotation matrix by a translation matrix that shifts the vertex to the origin, and then shifts it back to its original position after rotation.
* Finally, you need to apply the rotation matrix to the image using cv2.warpAffine. You can also specify the output size and the border mode.

Code scheme

# Import cv2, numpy, and imutils

import cv2

import numpy as np

import imutils

# Define a custom function to rotate an image around a vertex of a rectangle

def rotate\_bound\_vertex(image, angle, vertex):

# Grab the dimensions of the image and then determine the center

(h, w) = image.shape[:2]

# Grab the rotation matrix (applying the negative of the angle to rotate clockwise), then grab the sine and cosine

M = cv2.getRotationMatrix2D(vertex, -angle, 1.0)

cos = np.abs(M[0, 0])

sin = np.abs(M[0, 1])

# Compute the new bounding dimensions of the image

nW = int((h \* sin) + (w \* cos))

nH = int((h \* cos) + (w \* sin))

# Adjust the rotation matrix to take into account translation

T = np.array([[1, 0, -vertex[0]], [0, 1, -vertex[1]], [0, 0, 1]])

M = np.dot(T, M)

T = np.array([[1, 0, vertex[0]], [0, 1, vertex[1]], [0, 0, 1]])

M = np.dot(M, T)

M = M[:2]

# Perform the actual rotation and return the image

return cv2.warpAffine(image, M, (nW, nH))

# Read the image

img = cv2.imread("image.jpg")

# Get the coordinates of the rectangle

x, y, w, h = cv2.boundingRect(cnt) # cnt is the contour of the rectangle

x1, y1 = x, y # top-left vertex

x2, y2 = x + w, y + h # bottom-right vertex

x3, y3 = x, y + h # bottom-left vertex

x4, y4 = x + w, y # top-right vertex

# Choose the vertex to rotate around and the angle

vertex = (x1, y1) # top-left vertex

angle = 30 # degrees

# Rotate the image around the vertex

rotated = rotate\_bound\_vertex(img, angle, vertex)

# Display the rotated image

cv2.imshow("Rotated", rotated)

cv2.waitKey(0)

cv2.destroyAllWindows()