plate

- https://github.com/quantsolutions/anpr

install

darknet

install

bbox (optional)

old

plate detection with neural network

- https://matthewearl.github.io/2016/05/06/cnn-anpr/
- https://github.com/matthewearl/deep-anpr

http://www.pyimagesearch.com/2017/02/13/recognizing-digits-with-opencv-and-python/

```
# load the example image
image = cv2.imread("example.jpg")

# pre-process the image by resizing it, converting it to
# graycale, blurring it, and computing an edge map
image = imutils.resize(image, height=500)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Applying Gaussian blurring with a 5×5 kernel to reduce high-frequency
noise
blurred = cv2.GaussianBlur(gray, (5, 5), 0)

# Computing the edge map via the Canny edge detector.
edged = cv2.Canny(blurred, 50, 200, 255)

# find contours in the edge map, then sort them by their
# size in descending order
cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if imutils.is_cv2() else cnts[1]
cnts = sorted(cnts, key=cv2.contourArea, reverse=True)
displayCnt = None

# loop over the contours
```

```
for c in cnts:
    # approximate the contour
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.02 * peri, True)

    # if the contour has four vertices, then we have found
    # the thermostat display
    if len(approx) == 4:
        displayCnt = approx
        break

# extract the plate, apply a perspective transform to it
# Applying this perspective transform gives us a top-down, birds-eye-view of
plate
warped = four_point_transform(gray, displayCnt.reshape(4, 2))
output = four_point_transform(image, displayCnt.reshape(4, 2))

# threshold the warped image, then apply a series of morphological
# operations to cleanup the thresholded image
thresh = cv2.threshold(warped, 0, 255,
    cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (1, 5))
thresh = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)

# find contours in the thresholded image, then initialize the
# digit contours lists
cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if imutils.is_cv2() else cnts[1]
digitCnts = []

# loop over the digit area candidates
for c in cnts:
    # compute the bounding box of the contour
    (x, y, w, h) = cv2.boundingRect(c)

    # if the contour is sufficiently large, it must be a digit
    if w >= 15 and (h >= 30 and h <= 40):
        digitCnts.append(c)

# TODO display contour
# cv2.rectangle(output, (x, y), (x + w, y + h), (0, 255, 0), 1)


# sort the contours from left-to-right, then initialize the
# actual digits themselves
digitCnts = contours.sort_contours(digitCnts,
    method="left-to-right")[0]
digits = []

#     cv2.putText(output, str(digit), (x - 10, y - 10),
```

```
#     cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 255, 0), 2)
```

From:
https://wiki.csgalileo.org/ - **Galileo Labs**

Permanent link:
https://wiki.csgalileo.org/projects/plate?rev=1652475626

Last update: **2022/05/13 23:00**