APPENDIX

IMPORT THE NECESSARY MODULE

```
import numpy as np
import argparse
import imutils
import glob, os
import cv2
```

CONSTRUCT THE ARGUMENT PARSER AND PARSE THE ARGUMENTS

```
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--visualize",
help="Flag indicating whether or not to visualize each iteration")
args = vars(ap.parse_args())
```

READ THE TEMPLATE IMAGE FROM FILE

```
img_folder_path='C://Users/meldanophia/OneDrive/img/beargguy/temp'
dirListing = os.listdir(img_folder_path)
cv_img=[]
imagePath=[]
```

PROCESSING IMAGE INPUT FROM FILE

```
For imagePath in
   glob.glob("C://Users/meldanophia/OneDrive/img/beargguy/images/*
   .*"):
   i=0
   j=0
   startXx = 0
   StartYy = 0
   startXxx = 0
   StartYyy = 0
   maxValneh = 0.0
   startA = [1]*len(dirListing)
   startB = [1]*len(dirListing)
   startC = [1]*len(dirListing)
   startD = [1]*len(dirListing)
   nama = [1]*len(dirListing)
   maxValboi = [1]*len(dirListing)
```

READING IMAGE INPUT

```
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

LOOPING TEMPLATE BY TEMPLATE TO MATCH WITH INPUT IMAGE

```
for cv_img in
   glob.glob("C://Users/meldanophia/OneDrive/img/beargguy/temp/*.*
   "):
   found = None
   template = cv2.imread(cv_img)

   template = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)
   template = cv2.Canny(template, 50, 200)
   (tH, tW) = template.shape[:2]
```

LOOPING 20 TIMES IN LINE WITH RESIZING THE IMAGE

```
for scale in np.linspace(0.2, 1.0, 20)[::-1]:
      resized = imutils.resize(gray, width = int(gray.shape[1] *
   scale))
      r = gray.shape[1] / float(resized.shape[1])
      if resized.shape[0] < tH or resized.shape[1] < tW:
            break
```

PROCESS INPUT IMAGE WITH CANNY EDGE DETECTION AND PERFORM IT WITH TEMPLATE MATCHING METHOD IN CONJUNCTION WITH MINMAXLOC TO FIND MAXVAL

```
edged = cv2.Canny(resized, 50, 200)
result = cv2.matchTemplate(edged, template, cv2.TM_CCOEFF)
(_, maxVal, _, maxLoc) = cv2.minMaxLoc(result)
```

CHECK TO SEE IF THE ITERATION SHOULD BE VISUALIZED

```
if args.get("visualize", False):
   clone = np.dstack([edged, edged, edged])
   cv2.rectangle(clone, (maxLoc[0], maxLoc[1]),
   (maxLoc[0] + tW, maxLoc[1] + tH), (0, 0, 255), 2)
   cv2.imshow("Visualize", clone)
   cv2.waitKey(0)
```

INDICATE WHETHER MAXVAL IS FOUND OR NOT

```
if found is None or maxVal > found[0]:
   found = (maxVal, maxLoc, r)

   (maxVal, maxLoc, r) = found
   (startA [i], startC [i]) = (int(maxLoc[0] * r), int(maxLoc[1] *
   r))
   (startD[i],   StartB[i])   =   (int((maxLoc[0]   +   tW)   *   r),
   int((maxLoc[1] + tH) * r))
```

PROCESS TO MATCH MORE THAN ONE TEMPLATE IN ONE INPUT IMAGE

```
if startA [i] == startXx and startC [i] == StartYy or maxVal <
   4000000.0:
   continue
   maxValboi[i] = maxVal
   nama[i] = cv_img[50:-4]
   startXx = startA [i]
   StartYy = startC [i]
   i+=1

twin = [1.0]*len(maxValboi)
while j<len(maxValboi):
   if maxValboi[j] < 4000000.0:
       j+=1
       continue
   x = np.where(abs(np.array(startA) - startA [j]) <= 13)
   y = np.where(abs(np.array(startC) - startC [j]) <= 13)

   loc = j
   tempo = maxValboi[j]
   if len(x[0]) > 1 and len(y[0]) > 1:
       k = 0
       tempo = 0
       loc = 0
       while k<len(x[0]):
             if tempo < maxValboi[x[0][k]]:
                   loc = x[0][k]
                   tempo = maxValboi[loc]
                   k+=1
       z = np.where(np.array(twin) == tempo)
       if len(z[0]) != 0:
             j+=1
             continue
       twin[j] = tempo
```

DRAWING RECTANGLE AROUND THE MATCHED REGION WITH TEXT THAT SHOW WHAT THE SIGN IS IT

```
cv2.rectangle(image,  (startA[loc],  startC[loc]),  (startD[loc],
   StartB[loc]), (0, 0, 255), 2)
cv2.putText(image, nama[loc], (startA [loc], StartB [loc]+13),
cv2.FONT_HERSHEY_SIMPLEX, 0.36, (255, 255, 0), 1, cv2.LINE_AA)

maxValneh = maxValboi[j]
j+=1
```

SHOW THE OUTPUT RESULT

```
cv2.imshow("Image", image)
cv2.waitKey(0)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

PLAGIARISM CHECK.ORG

1.11% PLAGIARISM APPROXIMATELY

Report #10916004

IntroductionBackgroundThere are 5 groups of traffic signs that spreading in Indonesia s road, such as warning signs, instruction signs, prohibition signs, command signs and temporary signs. For example, bend left and right are two example of a warning sign, no stop and no parking are two example of a prohibition signs and etc. Conventional signs consist of leaf signs and pole posts. Generally, leaf signs are made of aluminum plates of other materials that meet the standard where signs are posted. While the sign pole is made of metal bars. Effective signs must meet the following standards, such as, meet the needs, attract attention and get respect for road users, give a message that is easy to understand, provide enough time for road users to respond. With the diversity of traffic signs in Indonesia it does not decreasing traffic offense in Indonesia. For example, in Jakarta, traffic infringement actually increased around 24,13% from 2017 entering 2018(Muhammad Ikbal, 2019, p. 3-4) [1]. The majority of these type of infringement such as infringing traffic signs, against directions, not wearing helmets, running red lights and so on. Some traffic signs are also made not according to standards. One common traffic offense is infringing traffic signs. In fact, too many traffic signs are unknown for many people. Therefore, the detection of traffic signs is needed to reduce violations that can cause traffic