```
1
2
3   from time import time
4   import cv2
5   from cv2 import imshow
6   import numpy as np
7   from matplotlib import pyplot as plt
8   import os
9   import tensorflow as tf
10  from object_detection.utils import label_map_util
11  from object_detection.utils import visualization_utils as viz_utils
12  from object_detection.builders import model_builder
13  from object_detection.utils import config_util
14  import time
15  import math
16
17  #Object detection, line(17-109), FOUND HERE:
    https://github.com/nicknochnack/TFODCourse/blob/main/2.%20Training%20and%20Detection.ipynb
18
19
20  #CUSTOM_MODEL_NAME = 'efficientdet_d1_coco17_tpu-32_V11_35000s'
21  #PRETRAINED_MODEL_NAME = 'efficientdet_d1_coco17_tpu-32'
22  #PRETRAINED_MODEL_URL =
    'http://download.tensorflow.org/models/object_detection/tf2/20200711/efficientdet_d1_coco17_tpu-32.tar.gz'
23  #TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
24  #LABEL_MAP_NAME = 'label_map.pbtxt'
25
26  CUSTOM_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_640x640_V4_sveis_40000s'
27  PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_640x640_coco17_tpu-8'
28  PRETRAINED_MODEL_URL =
    'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_640x640_coco17_tpu-8.tar.
    gz'
29  TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
30  LABEL_MAP_NAME = 'label_map.pbtxt'
31
32  paths = {
33      'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
34      'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
35      'APIMODEL_PATH': os.path.join('Tensorflow','models'),
36      'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
37      'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
38      'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
39      'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),
40      'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME),
41      'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'export'),
42      'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
43      'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
44      'PROTOC_PATH':os.path.join('Tensorflow','protoc')
45   }
46
47  files = {
```

```
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}

configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-41')).expect_partial()  #The longer a model trains the more chekpoints it returns, important to chose the last checkpoint bechause its traind the most.

def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections


category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])

cap = cv2.VideoCapture(1)                    #Choses what camera to use, internal web camera is often 0, external camera wil then be 1.

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

display_str_list = []

while cap.isOpened():
    #time.sleep(0.05)
    ret, frame = cap.read()
    image_np = np.array(frame)
    #cv2.imshow('TEST',  image_np)
    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

    label_id_offset = 1
    det_boxes = detections['detection_boxes']
    det_classes = detections['detection_classes']+label_id_offset
    det_scores = detections['detection_scores']
    image_np_with_detections = image_np.copy()

```

```
97
98      viz_utils.visualize_boxes_and_labels_on_image_array(
99                  image_np_with_detections,
100                 det_boxes,
101                 det_classes,
102                 det_scores,
103                 category_index,
104                 use_normalized_coordinates=True,
105                 max_boxes_to_draw=1,            #Only the boundingbox with the highest estimated accuracy wil be
                    showed, since the estimated accuracy tends to drop at longer distances the weld closest to the robot
                    is most likley to be detected.
106                 min_score_thresh=.5,            #The estimated accuracy needs to be at lest 50% to draw a bounding
                    box, this is to secure that it is a weld that is detected
107                 agnostic_mode=False)
108     #cv2.imshow('object detection',  cv2.resize(image_np_with_detections, (800, 650)))
109
110     ##########################################################################################################
111     # Find boundingbox cordinates, line(112-139), FOUND HERE: https://github.com/tensorflow/models/issues/4682
112     boxes = detections['detection_boxes']
113     height, width = image_np_with_detections.shape[:2]
114     box= np.squeeze(boxes)
115     max_boxes_to_draw=box.shape[0]
116     scores=np.squeeze(det_scores)
117     min_score_thresh=0.2
118
119
120     xmax = 0
121     xmin = 0
122     ymin = 0
123     ymax = 0
124
125     if scores [0] > min_score_thresh:           #If multible boxes are found it is bosible to only take the ones with
        highest estimated accuracy, in our case its not nesesary since we only detect one box at the time.
126          ymin = (int(box[0,0]*height))
127          xmin = (int(box[0,1]*width))
128          ymax = (int(box[0,2]*height))
129          xmax = (int(box[0,3]*width))
130          #print (xmin,ymin,xmax,ymax)
131
132     nPixelPipeDimension = xmax-xmin
133     nPixelWeldWidth = ymax-ymin
134
135     BoxDetected = True
136     if(xmax == 0 and xmin == 0 and ymin == 0 and ymax == 0): #If nothing is detectecd, the program notifies it
137         print('Boundingbox not detected')
138         BoxDetected = False
139             #print (xmin,ymin,xmax,ymax)
140
141     ##########################################################################################################
142
143     if(BoxDetected):
```

```
144             Sensitivitet = 10
145             SenseLineSpace = 30 #
146             Linepassage = 10 #In the futere the value needs to be adjusted with the distance from the weld, since att long
                distances 10 piksels wil be a longer compared to a shorter distance to the weld.
147
148
149         nPixelPipeDimension = xmax-xmin
150         nPixelWeldWidth = ymax-ymin
151         NYx = nPixelPipeDimension*0.3
152         NYy = nPixelWeldWidth*5
153
154         HorizontalPipe = False
155
156         if(nPixelPipeDimension<nPixelWeldWidth): #Since the object detection works with both the horisontal and vertical
            the program needs to determ which it is. This program ONLY works for the horizontal pipe
157             nD = nPixelPipeDimension
158             nPixelPipeDimension = nPixelWeldWidth
159             nPixelWeldWidth = nD
160             NYy = nPixelPipeDimension*1
161             NYx = nPixelWeldWidth*3
162             HorizontalPipe = True
163
164         RBxmin = round(xmin-NYx)
165         RBymin = round(ymin-NYy)
166         RBxmax = round(xmax+NYx)
167         RBymax = round(ymax+NYy)
168
169         error = ''
170         OutOfSenter = False
171         h = ''
172         v = ''
173         o = ''
174         u = ''
175
176         if(RBxmax>width):
177             OutOfSenter = True
178             h = 'R' #Go right
179             RBxmax = width
180         if(RBxmin<0):
181             OutOfSenter = True
182             v = 'L' #Go left
183             RBxmin = 0
184         if(RBymax>height):
185             OutOfSenter = True
186             u = 'D' #Go down
187             RBymax = height
188         if(RBymin<0):
189             OutOfSenter = True
190             o = 'U' #Go up
191             RBymin = 0
192
```

```
193         if(OutOfSenter == True):
194             error = 'Error: '+h+v+u+o
195 
196 
197         PipeMarked = cv2.rectangle(image_np,(RBxmin,RBymin),(RBxmax,RBymax),(0,255,0),1)
198         imgCrop = image_np[RBymin:RBymax,RBxmin:RBxmax] #Cuts out a smaller image around the bounding box
199         #imgCrop = cv2.rectangle(imgCrop, (20,20),(10,50),(0, 0, 255), 2) #referansepunkt
200 
201         ##########################################################################################################
202         #Writing text on image, line(203-204), FOUND HERE:
            https://github.com/techwithtim/OpenCV-Tutorials/blob/main/tutorial4.py
203         font = cv2.FONT_HERSHEY_SIMPLEX
204         cv2.putText(imgCrop, error, (25,30), font, 1, (0, 0, 240), 2, cv2.LINE_AA)
205 
206 
207         #imgCrop = cv2.resize(imgCrop,(round(RBxmax-RBxmin)*3,round(RBymax-RBymin)*3))
208 
            ##########################################################################################################

209         #HoughLine linedetection, line(211-253) https://docs.opencv.org/4.0.0/d6/d10/tutorial_py_houghlines.html
210 
211         heightC, widthC = imgCrop.shape[:2]
212         x1 = int
213         y1 = int
214         x2 = int
215         y2 = int
216 
217         #table = [1,2]*5 ta vekk når ferdig
218         print('H',heightC,'  W',widthC)
219         if(heightC > 0 and widthC > 0):
220             gray = cv2.cvtColor(imgCrop, cv2.COLOR_BGR2GRAY)
221             edges = cv2.Canny(gray,80,100,apertureSize = 3)
222             #imshow('bilder',edges)
223             #imshow('Bilde', edges) #Vise kanter på bildet
224             lines = cv2.HoughLines(edges,1,np.pi/180,100)
225             NoLinesFound = False
226 
227             try:
228                 nlines = len(lines)
229                 print(nlines)
230             except:
231                 print('No lines found')
232                 NoLinesFound = True
233 
234             if(NoLinesFound==False):
235 
236                 listx1 = [0]*nlines
237                 listx2 = [0]*nlines
238                 listy1 = [0]*nlines
239                 listy2 = [0]*nlines
240                 counter = 0
```

```
241
242             for line in lines:
243                     rho,theta = line[0]
244                     a = np.cos(theta)
245                     b = np.sin(theta)
246                     x0 = a*rho
247                     y0 = b*rho
248                     listx1[counter] = int(x0 + 1000*(-b))
249                     listy1[counter] = int(y0 + 1000*(a))
250                     listx2[counter] = int(x0 - 1000*(-b))
251                     listy2[counter] = int(y0 - 1000*(a))
252                     cv2.line(imgCrop,(listx1[counter],listy1[counter]),(listx2[counter],listy2[counter]),(250,2,25),1)
253                     counter = counter + 1
254
            ##########################################################################################################
255             #imshow('bilder',imgCrop)
256             Hpointx1 = []
257             Vpointx1 = []
258             Hpunktx2 = []
259             Vpointx2 = []
260
261             VpointAVGx1 = 0
262             VpointAVGx2 = 0
263             HpointAVGx1 = 0
264             HpunktAVGx2 = 0
265
266             BSpointy1 = []
267             OSpointy1 = []
268             BSpointy2 = []
269             OSpointy2 = []
270
271             BSpointAVGy1 = 0
272             BSpointAVGy2 = 0
273             OSpointAVGy1 = 0
274             OSpointAVGy2 = 0
275
276             BSpointx1 = []
277             OSpointx1 = []
278             BSpointx2 = []
279             OSpointx2 = []
280
281             BSpointAVGx1 = 0
282             BSpointAVGx2 = 0
283             OSpointAVGx1 = 0
284             OSpointAVGx2 = 0
285
286             for i in range(nlines):
287                 xm1 = 0
288                 xm2 = 0
```

```
289
290                 if(HorizontalPipe==True):
291                     xm1 = listy1[i]
292                     xm2 = listy2[i]
293                     ym1 = listx1[i]
294                     ym2 = listx2[i]
295                     #cv2.line(imgCrop,(ym2,xm2),(ym1,xm1),(250,2,25),1)
296                     #cv2.line(imgCrop,(listx1[i],listy1[i]),(listx2[i],listy2[i]),(250,2,25),1)
297                 else:
298                     xm1 = listx1[i]
299                     xm2 = listx2[i]
300                     #cv2.line(imgCrop,(heightC-5,xm1),(5,xm2),(250,2,25),1)
301                     #cv2.line(imgCrop,(listx1[i],listy1[i]),(listx2[i],listy2[i]),(250,2,25),1)
302
303
304                 for j in range(nlines):
305
306                     xt1=0
307                     xt2=0
308                     yt1=0
309                     yt2=0
310
311                     if(HorizontalPipe == True):
312                         xt1 = listy1[j]
313                         xt2 = listy2[j]
314                         yt1 = listx1[j]
315                         yt2 = listx2[j]
316                     else:
317                         xt1 = listx1[j]
318                         xt2 = listx2[j]
319
320                     def hightSenterOfHyp(xm1,xm2,ym1,ym2): #Calculating where the line passes at the senter of the
                    image
321                          x5 = xm2-xm1
322                          y4 = abs(ym1)+ym2
323                          ang = math.atan(x5/y4)
324                          x3 = (abs(ym1)+widthC/2)*math.tan(ang)
325                          x4 = round(x3 + xm1)
326
327                          return x4
328
329                     def pointcheck(h):
330                          wc2 = heightC/2
331                          ovs = wc2 - nPixelPipeDimension/2
332                          uns = wc2 + nPixelPipeDimension/2
333                          x1 = ovs-Linepassage
334                          x2 = ovs+Linepassage
335                          y1 = uns+Linepassage
336                          y2 = uns-Linepassage
337
338                          cv2.circle(imgCrop,(round(widthC/2),round(x1)),5,(30,249,100),-1)
```

```
                    cv2.circle(imgCrop,(round(widthC/2),round(x2)),5,(30,249,100),-1)
                    cv2.circle(imgCrop,(round(widthC/2),round(y1)),5,(30,249,100),-1)
                    cv2.circle(imgCrop,(round(widthC/2),round(y2)),5,(30,249,100),-1)

                    i = False

                    if(x1<h<x2 or y2<h<y1):
                        i = True

                    return i



            if(HorizontalPipe == True):

                if((nPixelPipeDimension-Sensitivitet<abs(xm1-xt1)<nPixelPipeDimension+Sensitivitet)and(
                nPixelPipeDimension-Sensitivitet<abs(xm2-xt2)<nPixelPipeDimension+Sensitivitet)): #checks if
                the lines are close to each other, this is from a erlier stage and is not of many use.

                    h = hightSenterOfHyp(xm1,xm2,ym1,ym2)
                    #print('h: ',h)
                    #qcv2.circle(imgCrop,(round(widthC/2),round(heightC/2)),5,(30,249,100),-1)

                    #cv2.circle(imgCrop,(round(widthC/2),round(heightC/2)+round(nPixelRDim/2)),5,(30,249,100),-
                    1)

                    #cv2.circle(imgCrop,(round(widthC/2),round(heightC/2)-round(nPixelRDim/2)),5,(30,249,100),-
                    1)
                    #cv2.circle(imgCrop,(round(widthC/2),h),5,(0,0,200),-1)

                    if(h>heightC/2 and xm2>0 and xm2>0 and pointcheck(h)):
                        #cv2.circle(imgCrop,(round(widthC/2),h),5,(0,0,200),-1)
                        BSpointx1.insert(0,xm1)
                        BSpointx2.insert(0,xm2)
                        BSpointy1.insert(0,ym1)
                        BSpointy2.insert(0,ym2)
                        #cv2.line(imgCrop,(ym2,xm2),(ym1,xm1),(250,2,25),1)
                        #print('BS: xm1: ',xm1, 'xm2 :',xm2,' y',j,' ', widthC/2)


                    elif(h<heightC/2 and xm2>0 and xm2>0 and pointcheck(h)):
                        #cv2.circle(imgCrop,(round(widthC/2),h),5,(0,0,200),-1)
                        OSpointx1.insert(0,xm1)
                        OSpointx2.insert(0,xm2)
                        OSpointy1.insert(0,ym1)
                        OSpointy2.insert(0,ym2)
                        #cv2.line(imgCrop,(ym2,xm2),(ym1,xm1),(250,2,25),1)
                        #print('OS: xm1: ',xm1, 'xm2 :',xm2,' y',j,' ', widthC/2)

                #imshow('Crop',imgCrop)

```

```
384
                    ##########################################################################################
                    #
385                 #This code is for vertical pip and is not operational at the moment
386                 else:
387                     if((nPixelPipeDimension-Sensitivitet<abs(xm1-xt1)<nPixelPipeDimension+Sensitivitet)and(
                        nPixelPipeDimension-Sensitivitet<abs(xm2-xt2)<nPixelPipeDimension+Sensitivitet)): #Sjekker att
                        punkta ikkje er for nerme/vekke frå kvarandre, kunn x vediene blir sjekka, fungerer kunn
                        horisontalt
388
389
390                         if(xm1>widthC/2 and xm2>0 and xm2>0):
391                             if(widthC/2 + nPixelPipeDimension/2 -Sensitivitet<xm1<widthC/2 + nPixelPipeDimension/2
                            +Sensitivitet):
392                                 Hpointx1.insert(0,xm1)
393                                 Hpunktx2.insert(0,xm2)
394                                 #print('H: xm1: ',xm1, 'xm2 :',xm2,' y',j,' ', widthC/2)
395
396                         elif(xm1<widthC/2 and xm1>0 and xm2>0):
397                             if(widthC/2 - nPixelPipeDimension/2 -Sensitivitet<xm1<widthC/2 - nPixelPipeDimension/2
                            +Sensitivitet):
398                                 Vpointx1.insert(0,xm1)
399                                 Vpointx2.insert(0,xm2)
400                                 #print('V: xm1: ',xm1, 'xm2 :',xm2,' y',j,' ',widthC/2)
401                     #imshow('Crop',imgCrop)
402
403
404
405             def AVGpunkt(ListPoints):
406                 pointAVG = 0
407                 for i in ListPoints:
408                     pointAVG = pointAVG+i
409
410                 try:
411                     pointAVG = round(pointAVG/len(ListPoints))
412                 except:
413                     pointAVG = 0
414
415                 return(pointAVG)
416
417             if(HorizontalPipe == True):
418
419                 OSpointAVGx1 = AVGpunkt(OSpointx1)
420                 OSpointAVGx2 = AVGpunkt(OSpointx2)
421                 OSpointAVGy1 = AVGpunkt(OSpointy1)
422                 OSpointAVGy2 = AVGpunkt(OSpointy2)
423
424                 BSpointAVGx1 = AVGpunkt(BSpointx1)
425                 BSpointAVGx2 = AVGpunkt(BSpointx2)
426                 BSpointAVGy1 = AVGpunkt(BSpointy1)
427                 BSpointAVGy2 = AVGpunkt(BSpointy2)
```

```
428
429                 print('Start')
430
431                 print('OSx1 ',OSpointAVGx1)
432                 print('OSy1 ',OSpointAVGy1)
433                 print('OSx2 ',OSpointAVGx2)
434                 print('OSy2 ',OSpointAVGy2)
435
436                 print('BSx1 ',BSpointAVGx1)
437                 print('BSy1 ',BSpointAVGy1)
438                 print('BSx2 ',BSpointAVGx2)
439                 print('BSy2 ',BSpointAVGy2)
440
441                 print('Slutt')
442
443                 #cv2.line(imgCrop,(OSpunkty2[i],OSpunktx2[i]),(OSpunkty1[i],OSpunktx1[i]),(250,2,25),1)
444                 imgCrop = cv2.line(imgCrop,(OSpointAVGy2,OSpointAVGx2),(OSpointAVGy1,OSpointAVGx1),(250,2,25),1)
445                 imgCrop = cv2.line(imgCrop,(BSpointAVGy2,BSpointAVGx2),(BSpointAVGy1,BSpointAVGx1),(250,2,25),1)
446                 imshow('Crop',imgCrop) #Shows cropped image with line detection
447
448
449                 #cv2.circle(imgCrop,(xs,ys),5,(0,0,200),-1)
450
451             else:
452                 VpointAVGx1 = AVGpunkt(Vpointx1)
453                 VpointAVGx2 = AVGpunkt(Vpointx2)
454                 if(VpointAVGx1 != 0): print('VpunktAVGx1: ',VpointAVGx1,'  VpunktAVGx2: ',VpointAVGx2)
455                 else: print('ingen punkt til venstre')
456
457
458                 HpointAVGx1 = AVGpunkt(Hpointx1)
459                 HpunktAVGx2 = AVGpunkt(Hpunktx2)
460                 if(HpointAVGx1 != 0): print('HpunktAVGx1: ',HpointAVGx1,'  HpunktAVGx2: ',HpunktAVGx2)
461                 else: print('ingen punkt til Høgre')
462
463                 if not(VpointAVGx1==0 or HpointAVGx1==0 or VpointAVGx1==0 or VpointAVGx2==0):
464                     #cv2.line(imgCrop,(VpunktAVGx1,heightC-5),(VpunktAVGx2,5),(250,2,25),1)
465                     #cv2.line(imgCrop,(HpunktAVGx1,heightC-5),(HpunktAVGx2,5),(250,2,25),1)
466                     print('VERKTIKAL')
467                     cv2.circle(imgCrop,(VpointAVGx1,heightC-5),3,(100,0,200),-1)
468                     cv2.circle(imgCrop,(VpointAVGx2,5),3,(100,0,200),-1)
469
470                     cv2.circle(imgCrop,(HpointAVGx1,heightC-5),3,(100,0,200),-1)
471                     cv2.circle(imgCrop,(HpunktAVGx2,5),3,(100,0,200),-1)
472
473                     #imgCrop = cv2.resize(imgCrop,(round(RBxmax-RBxmin)*3,round(RBymax-RBymin)*3))
474                     #edges = cv2.resize(edges,(round(RBxmax-RBxmin)*3,round(RBymax-RBymin)*3))
475                     if(VpointAVGx2-VpointAVGx1 != 0 and HpunktAVGx2 - HpointAVGx1 != 0):
476                         v1 = math.atan(heightC/(VpointAVGx2-VpointAVGx1))*180/math.pi
477                         print('VinkelV: ',v1)
478                         v2 = np.arctan(heightC/(HpunktAVGx2-HpointAVGx1))*180/math.pi
```

```
                        print('VinkelH: ',v2)

                        AVGvinkel = round((v1+v2)/2,2)
                        print('AVGvinkel: ', AVGvinkel )

                        if(AVGvinkel<89):
                            VinkelDisplay = "Vinkel:"+str(AVGvinkel)+"deg Roter "+str(round(90-AVGvinkel,2)) +" deg
                            til venstre"
                        elif(AVGvinkel>91):
                            VinkelDisplay = "Vinkel:"+str(AVGvinkel)+"deg Roter "+str(round(AVGvinkel-90,2)) +" deg
                            til Høgre"
                        elif(89<AVGvinkel<91):
                            VinkelDisplay = 'Tilnermet 90 deg'
                        #print(VinkelDisplay)

                        ##########################################################################################
                        #Text display, line(494-500
                        )https://www.tutorialspoint.com/display-text-on-an-opencv-window-by-using-the-function-puttext#
                        :~:text=Display%20text%20on%20an%20OpenCV%20window%20by%20using%20the%20function%20putText(),-O
                        penCVPythonServer&text=In%20this%20program%2C%20we%20will,%2C%20color%2C%20thickness%2C%20etc.
                        coordinates = (10,40)
                        font = cv2.FONT_HERSHEY_SIMPLEX
                        fontScale = 0.5
                        color = (0,255,255)
                        thickness = 1
                        #imgCrop=cv2.putText(imgCrop, VinkelDisplay, coordinates, font, fontScale, color, thickness,
                        cv2.LINE_AA)
                        #cv2.imshow("Text", image)
                        ##########################################################################################
            #imshow('Crop',imgCrop)


        if cv2.waitKey(10) & 0xFF == ord('q'):
            cap.release()
            cv2.destroyAllWindows()
            break
```