20190969 박고은

|  |  |
| --- | --- |
| 6-1 자동차 번호판 숫자 인식 | |
| Plate\_candidate.py | import numpy as np, cv2 def color\_candidate\_img(image, center):  h, w = image.shape[:2]  fill = np.zeros((h + 2, w + 2), np.uint8) # 채움 영역  dif1, dif2 = (25, 25, 25), (25, 25, 25) # 채움 색상 범위  flags = 4 + 0xff00 + cv2.FLOODFILL\_FIXED\_RANGE # 채움 방향 및 방법  flags += cv2.FLOODFILL\_MASK\_ONLY   # 후보 영역을 유사 컬러로 채우기  pts = np.random.randint( -15, 15, (20,2) ) # 20개 좌표 생성  pts = pts + center  for x, y in pts: # 랜덤 좌표 평행 이동  if 0 <= x < w and 0 <= y < h:  \_, \_, fill, \_ = cv2.floodFill(image, fill, (x,y), 255, dif1, dif2, flags)   # 이진화 및 외곽영역 추출후 회전 사각형 검출  return cv2.threshold(fill, 120, 255, cv2.THRESH\_BINARY)[1]    def rotate\_plate(image, rect):  center, (w, h), angle = rect # 중심점, 크기, 회전 각도  if w < h : # 세로가 긴 영역이면  w, h = h, w # 가로와 세로 맞바꿈  angle += 90 # 회전 각도 조정   size = image.shape[1::-1] # 행태와 크기는 역순  rot\_mat = cv2.getRotationMatrix2D(center, angle, 1) # 회전 행렬 계산  rot\_img= cv2.warpAffine(image, rot\_mat, size, cv2.INTER\_CUBIC) # 회전 변환   crop\_img = cv2.getRectSubPix(rot\_img, (w, h), center) # 후보영역 가져오기  crop\_img = cv2.cvtColor(crop\_img, cv2.COLOR\_BGR2GRAY)  return cv2.resize(crop\_img, (144, 28)) |
| Plate\_classify.py | import numpy as np, cv2 from Common.knn import find\_number, place\_middle  def kNN\_train(train\_fname, K, nclass, nsample):  size = (40, 40) # 숫자 영상 크기  train\_img = cv2.imread(train\_fname, cv2.IMREAD\_GRAYSCALE) # 학습 영상 적재  h, w = train\_img.shape[:2]  dy = h % size[1]// 2  dx = w % size[0]// 2  train\_img = train\_img[dy:h-dy-1, dx:w-dx-1] # 학습 영상 여백 제거  cv2.threshold(train\_img, 32, 255, cv2.THRESH\_BINARY, train\_img)   cells = [np.hsplit(row, nsample) for row in np.vsplit(train\_img, nclass)]  nums = [find\_number(c) for c in np.reshape(cells, (-1, 40,40))]  trainData = np.array([place\_middle(n, size) for n in nums])  labels = np.array([i for i in range(nclass) for j in range(nsample)], np.float32)   knn = cv2.ml.KNearest\_create()  knn.train(trainData, cv2.ml.ROW\_SAMPLE, labels) # k-NN 학습 수행  return knn  # 번호판 영상 전처리 def preprocessing\_plate(plate\_img):  plate\_img = cv2.resize(plate\_img, (180, 35)) # 번호판 영상 크기 정규화  flag = cv2.THRESH\_BINARY | cv2.THRESH\_OTSU # 이진화 방법  cv2.threshold(plate\_img, 32, 255, flag, plate\_img) # 이진화   h, w = plate\_img.shape[:2]  dx, dy = (6, 3)  ret\_img= plate\_img[dy:h-dy, dx:w-dx] # 여백 제거  return ret\_img  # 숫자 및 문자 객체 검색 def find\_objects(sub\_mat):  results = cv2.findContours(sub\_mat, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  contours = results[0] if int(cv2.\_\_version\_\_[0]) >= 4 else results[1]   rois = [cv2.boundingRect(contour) for contour in contours]  rois = [(x, y, w, h, w\*h) for x,y,w,h in rois if w / h < 2.5]   text\_rois = [(x, y, x+w, y+h) for x, y, w, h, a in rois if 45 < x < 80 and a > 60]  num\_rois = [(x, y, w, h) for x, y, w, h, a in rois if not(45 < x < 80) and a > 150]   if text\_rois: # 분리된 문자 영역 누적  # pts= cv2.sort(np.array(text\_rois), cv2.SORT\_EVERY\_COLUMN) # 열단위 오름차순  pts= np.sort(text\_rois, axis=0) # y 방향 정렬  x0, y0 = pts[ 0, 0:2] # 시작좌표 중 최소인 x, y 좌표  x1, y1 = pts[-1, 2:] # 종료좌표 중 최대인 x, y 좌표  w, h = x1-x0, y1-y0 # 너비, 높이 계산  num\_rois.append((x0, y0, w, h)) # 문자 영역 구성 및 저장   return num\_rois  # 검출 객체 영상의 숫자 및 문자 인식 def classify\_numbers(cells, nknn, tknn, K1, K2, object\_rois):  if len(cells) != 7:  print("검출된 숫자(문자)가 7개가 아닙니다.")  return   texts = "가나다라마거너더러머버서어저고노도로모보" \  "소오조구누두루무부수우주아바사자바하허호"   numbers = [find\_number(cell) for cell in cells]  datas = [place\_middle(num, (40,40)) for num in numbers]  datas = np.reshape(datas, (len(datas), -1))   idx = np.argsort(object\_rois, axis=0).T[0]  text = datas[idx[2]].reshape(1,-1)   \_, resp1, \_, \_ = nknn.findNearest(datas, K1) # 숫자 k-NN 분류 수행  \_, [[resp2]], \_, \_ = tknn.findNearest(text, K2) # 문자 k-NN 분류 수행   resp1 = resp1.flatten().astype('int')  results = resp1[idx].astype(str)  results[2] = texts[int(resp2)]   print("정렬 인덱스:", idx)  print("숫자 분류 결과:", resp1)  print("문자 분류 결과:", int(resp2))  print("분류 결과: ", ' '.join(results)) |
| Plate\_preprocess.py | import numpy as np, cv2  def find\_value\_position(img, direct):  project = cv2.reduce(img, direct, cv2.REDUCE\_AVG).ravel()  p0, p1 = -1, -1 # 초기값  len = project.shape[0] # 전체 길이  for i in range(len):  if p0 < 0 and project[i] < 250: p0 = i  if p1 < 0 and project[len-i-1] < 250 : p1 = len-i-1  return p0, p1  def find\_number(part):  x0, x1 = find\_value\_position(part, 0) # 수직 투영  y0, y1 = find\_value\_position(part, 1) # 수평 투영  return part[y0:y1, x0:x1]  # 숫자 객체 셀 중심 배치 def place\_middle(number, new\_size):  h, w = number.shape[:2]  big = max(h, w)  square = np.full((big, big), 255, np.float32) # 실수 자료형   dx, dy = np.subtract(big, (w,h))//2  square[dy:dy + h, dx:dx + w] = number  return cv2.resize(square, new\_size).flatten() # 크기변경 및 벡터변환 후 반환  def preprocessing(car\_no):  image = cv2.imread("images/car/%02d.jpg" % car\_no, cv2.IMREAD\_COLOR)  if image is None: return None, None   kernel = np.ones((5, 13), np.uint8) # 닫힘 연산 마스크  gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) # 명암도 영상 변환  gray = cv2.blur(gray, (5, 5)) # 블러링  gray = cv2.Sobel(gray, cv2.CV\_8U, 1, 0, 3) # 소벨 에지 검출   th\_img = cv2.threshold(gray, 120, 255, cv2.THRESH\_BINARY)[1] # 이진화 수행  morph = cv2.morphologyEx(th\_img, cv2.MORPH\_CLOSE, kernel, iterations=3)   # cv2.imshow("th\_img", th\_img); cv2.imshow("morph", morph)  return image, morph   def verify\_aspect\_size(size):  w, h = size  if h == 0 or w == 0: return False   aspect = h / w if h > w else w / h # 종횡비 계산   chk1 = 3000 < (h \* w) < 12000 # 번호판 넓이 조건  chk2 = 2.0 < aspect < 6.5 # 번호판 종횡비 조건  return (chk1 and chk2)   def find\_candidates(image):  results = cv2.findContours(image, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  contours = results[0] if int(cv2.\_\_version\_\_[0]) >= 4 else results[1]   rects = [cv2.minAreaRect(c) for c in contours] # 외곽 최소 영역  candidates = [(tuple(map(int, center)), tuple(map(int, size)), angle)  for center, size, angle in rects if verify\_aspect\_size(size)]   return candidates |
| 08.classify\_numbers.py | from plate\_preprocess import \* from plate\_candidate import \* from plate\_classify import \*  car\_no = int(input("자동차 영상 번호 (0~20): ")) image, morph = preprocessing(car\_no) # 전처리 candidates = find\_candidates(morph) # 후보 영역 검색  fills = [color\_candidate\_img(image, size) for size, \_, \_ in candidates] new\_candi = [find\_candidates(fill) for fill in fills] new\_candi = [cand[0] for cand in new\_candi if cand] candidate\_imgs = [rotate\_plate(image, cand) for cand in new\_candi]  svm = cv2.ml.SVM\_load("SVMTrain.xml") # 학습된 데이터 적재 rows = np.reshape(candidate\_imgs, (len(candidate\_imgs), -1)) # 1행 데이터들로 변환 \_, results = svm.predict(rows.astype('float32')) # 분류 수행 result = np.where(results.flatten() == 1)[0] # 1인 값의 위치 찾기  plate\_no = result[0] if len(result)>0 else -1  K1, K2 = 10, 10 nknn = kNN\_train("images/train\_numbers.png", K1, 10, 20) # 숫자 학습 tknn = kNN\_train("images/train\_texts.png", K2, 40, 20) # 문자 학습  if plate\_no >= 0:  plate\_img = preprocessing\_plate(candidate\_imgs[plate\_no]) # 번호판 영상 전처리  cells\_roi = find\_objects(cv2.bitwise\_not(plate\_img))  cells = [plate\_img[y:y+h, x:x+w] for x,y,w,h in cells\_roi]   classify\_numbers(cells, nknn, tknn, K1, K2, cells\_roi) # 숫자 객체 분류   pts = np.int32(cv2.boxPoints(new\_candi[plate\_no]))  cv2.polylines(image, [pts], True, (0, 255, 0), 2)   color\_plate = cv2.cvtColor(plate\_img, cv2.COLOR\_GRAY2BGR) # 컬러 번호판 영상  for x,y, w, h in cells\_roi:  cv2.rectangle(color\_plate, (x,y), (x+w,y+h), (0, 0, 255), 1) # 번호판에 사각형 그리기   h,w = color\_plate.shape[:2]  image[0:h, 0:w] = color\_plate else:  print("번호판 미검출")  cv2.imshow("image", image) cv2.waitKey(0) |
| 결과 |  |
|  |

|  |  |
| --- | --- |
| **6-2 YOLO를 이용한 물체 인식** | |
| import cv2 import numpy as np import pafy from PIL import ImageFont, ImageDraw, Image  classes = [] with open("./dnn/coco\_k.names", "rt", encoding="UTF8") as f:  classes = [line.strip() for line in f.readlines()]  colors = np.random.uniform(0, 255, size=(len(classes), 3)) # 실습 model = cv2.dnn.readNet("./dnn/yolov3.weights", "./dnn/yolov3.cfg") layer\_names = model.getLayerNames() output\_layers = [layer\_names[i[0] -1] for i in model.getUnconnectedOutLayers()]  CONF\_THR = 0.5   font = ImageFont.truetype("fonts/gulim.ttc", 20)  url = 'https://www.youtube.com/watch?v=Q1iyBo-sewE&t=431s' video = pafy.new(url) best = video.getbest(preftype='mp4') cap = cv2.VideoCapture(best.url) if not cap.isOpened(): raise Exception("동영상파일 개방 안됨")  while True:  ret, frame = cap.read()  if not ret: break   # 프레임을 모델에 넣기위해 전처리  blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)  # 모델에 넣어주기  model.setInput(blob)  #결과 받아오기  output = model.forward(output\_layers)   #주어진 이미지의 크기는 실행 속도에 영향을 미치므로 크기를 줄여서 진행  h, w = frame.shape[0:2]  img = cv2.resize(frame, dsize = (int(frame.shape[1] / 2), int(frame.shape[0]/2)))  ih = int(h/2)  iw = int(w/2)   class\_ids= []  confidences =[]  boxes = []   for out in output:  for detection in out: #각각의 물체에 대해 처리  scores = detection[5:]  class\_id = np.argmax(scores)  conf = scores[class\_id] #확률값   if conf > CONF\_THR: #확률값이 일정값을 넘는 경우  center\_x=int(detection[0] \* iw)  center\_y = int(detection[1]\*ih)  w = int(detection[2]\*iw)  h = int(detection[3]\*ih)  x = int(center\_x-w/2)  y=int(center\_y-h/2)   #각각의 박스에 대한 정보 저장  boxes.append([x,y,w,h])   confidences.append(float(conf))  class\_ids.append(class\_id)    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)   for i in range(len(boxes)):   if i in indexes: #인덱스에 남아있는 박스들을 표시해준다  x, y, w, h = boxes[i]  label = str(classes[class\_ids[i]])  color = colors[i]  conf = confidences[i]  # 네모로 물체를 표시  cv2.rectangle(img, (x,y), (x+w, y+h), color, 2)  #cv2.putText(img, '{}: {: 2f}'.format(label, conf), (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 1, color, 2)  # 물체가 무엇인지 표시  img\_pil = Image.fromarray(img)  draw = ImageDraw.Draw(img\_pil)  draw.text((x, y - 30), '{}: {:.2f}'.format(label, conf), font=font, fill=(0, 255, 0))  img = np.array(img\_pil)    cv2.imshow('frame', img)   #cv2.imshow('frame', frame)  key = cv2.waitKey(3)  if key == 27: break  cv2.destroyAllWindows() | |
| 결과(밑줄은 처음 등장한 물건) | |
| https://www.youtube.com/watch?v=Cc3q-kidzrI&t=148s | |
|  |  |
| 도서, 의자, 키보드(건반), 마우스(쥐), 모니터 | 사람, 휴대폰 |
| https://www.youtube.com/watch?v=Q1iyBo-sewE&t=431s | https://www.youtube.com/watch?v=zayjaCvkWQc&t=55s |
|  |  |
| 노트북, 화분, 사람, 키보드(건반), 마우스(쥐), 모니터, 의자 | 사람, 식탁, 병, 의자, 싱크대 |