|  |  |
| --- | --- |
| 교육 제목 | **OpenCV Object Detection practice** |
| 교육 일시 | 2021년 12월 24 일 금 요일 |
| 교육 장소 | 영우글로벌러닝 2층 |
| **교육 내용** | |
| 오전 | 1. 객체 단위 분석    1. 객체 단위 분석은 객체를 분할하여 특징을 분석하는 것을 의미합니다.    2. 영상이 입력되었을 때 객체와 배경이 분리될 수 있다고 가정하며 이진화를 통해 객체와 배경을 분리합니다.    3. 각각의 객체의 모양과 크기를 분석해서 내가 원하는 객체가 어디에 이쓰지 확인하고 싶을 때 객체단위 분석이 필요합니다.    4. 객체 위치 및 크기 정보, ROI 추출, 모양 분석 등을 할 수 있습니다.    5. 객체 단위 분석 방법은 레이블링과 외곽선 검출이 있습니다. 2. 객체 정보를 함께 반환하는 레이블링 함수    1. cv2.connectedComponentsWithStats    2. 객체의 크기와 중심위치도 함께 반환합니다.    3. cv2.connectedComponentsWithStats(image, labels=None, stats=None, centroids=None, connectivity=None, ltype=None) -> retval, labels, stats, centroids       * • image: 8비트 1채널 영상       * • labels: 레이블 맵 행렬. 입력 영상과 같은 크기. numpy.ndarray.       * • stats: 각 객체의 바운딩 박스, 픽셀 개수 정보를 담은 행렬. numpy.ndarray. shape=(N, 5), dtype=numpy.int32.       * • centroids: 각 객체의 무게 중심 위치 정보를 담은 행렬 numpy.ndarray. shape=(N, 2), dtype=numpy.float64.       * • ltype: labels 행렬 타입. cv2.CV\_32S 또는 cv2.CV\_16S. 기본값은 cv2.CV\_32S       * 반환값으로 retval, labels, stats, centroids 를 반환합니다.    4. retval : 객체 수 + 1 (배경 포함)    5. labels : 객체에 번호가 지정된 레이블 맵    6. stats : N행 5열, N은 객체 수 + 1이며 각각의 행은 번호가 지정된 객체를 의미, 5열에는 x, y, width, height, area 순으로 정보가 담겨 있습니다. x,y 는 좌측 상단 좌표를 의미하며 area는 면적, 픽셀의 수    7. centroids : N행 2열, 2열에는 x,y 무게 중심 좌표가 입력되어 있습니다. 무게 중심 좌표는 픽셀의 x 좌표를 다 더해서 갯수로 나눈 값입니다. y좌표도 동일합니다. |
| 오후 | * 1. def yolo(frame, size, score\_threshold, nms\_threshold):   2. # YOLO 네트워크 불러오기   3. net = cv2.dnn.readNet(f"yolov3\_{size}.weights", "yolov3.cfg")   4. layer\_names = net.getLayerNames()   5. output\_layers = [layer\_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]   6. # 클래스의 갯수만큼 랜덤 RGB 배열을 생성   7. colors = np.random.uniform(0, 255, size=(len(classes), 3))   8. # 이미지의 높이, 너비, 채널 받아오기   9. height, width, channels = frame.shape   10. # 네트워크에 넣기 위한 전처리   11. blob = cv2.dnn.blobFromImage(frame, 0.00392, (size, size), (0, 0, 0), True, crop=False)   12. # 전처리된 blob 네트워크에 입력   13. net.setInput(blob)   14. # 결과 받아오기   15. outs = net.forward(output\_layers)   16. # 각각의 데이터를 저장할 빈 리스트   17. class\_ids = []   18. confidences = []   19. boxes = []   20. for out in outs:   21. for detection in out:   22. scores = detection[5:]   23. class\_id = np.argmax(scores)   24. confidence = scores[class\_id]   25. if confidence > 0.1:   26. # 탐지된 객체의 너비, 높이 및 중앙 좌표값 찾기   27. center\_x = int(detection[0] \* width)   28. center\_y = int(detection[1] \* height)   29. w = int(detection[2] \* width)   30. h = int(detection[3] \* height)   31. # 객체의 사각형 테두리 중 좌상단 좌표값 찾기   32. x = int(center\_x - w / 2)   33. y = int(center\_y - h / 2)   34. boxes.append([x, y, w, h])   35. confidences.append(float(confidence))   36. class\_ids.append(class\_id)   37. # 후보 박스(x, y, width, height)와 confidence(상자가 물체일 확률) 출력   38. print(f"boxes: {boxes}")   39. print(f"confidences: {confidences}")   40. # Non Maximum Suppression (겹쳐있는 박스 중 confidence 가 가장 높은 박스를 선택)   41. indexes = cv2.dnn.NMSBoxes(boxes, confidences, score\_threshold=score\_threshold, nms\_threshold=nms\_threshold)   43. # 후보 박스 중 선택된 박스의 인덱스 출력   44. print(f"indexes: ", end='')   45. for index in indexes:   46. print(index, end=' ')   47. print("\n\n============================== classes ==============================")   48. for i in range(len(boxes)):   49. if i in indexes:   50. x, y, w, h = boxes[i]   51. class\_name = classes[class\_ids[i]]   52. label = f"{class\_name} {confidences[i]:.2f}"   53. color = colors[class\_ids[i]]   54. # 사각형 테두리 그리기 및 텍스트 쓰기   55. cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)   56. cv2.rectangle(frame, (x - 1, y), (x + len(class\_name) \* 13 + 65, y - 25), color, -1)   57. cv2.putText(frame, label, (x, y - 8), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, (0, 0, 0), 2)   59. # 탐지된 객체의 정보 출력   60. print(f"[{class\_name}({i})] conf: {confidences[i]} / x: {x} / y: {y} / width: {w} / height: {h}")   61. return frame |