컴퓨터비전 Assignment1

컴퓨터학부 20192403 박상철

과제 1

구현 방법

먼저 이미지를 (600, 600) 사이즈로 조정하여 양방향 필터를 통해 잡음을 제거하였습니다. 해당 기능을 구현하기 위해 저는 크게 3단계로 나누었습니다. 가장먼저 Canny함수를 통해 전저리를 하고 HouphLine함수를 통해 직선을 찾아내었습니다. 그러나 하나의 선분이 두개 혹은 그 이상의 직선으로 검출되는 문제가 발생해 인접 선분을 찾아 픽셀값이 30이내라면 하나의 선분으로 병합하였습니다.(처음 이미지의 크기를 설정하여 서로 다른 선분은 30픽셀 이내에 있을 수 없다고 판정). 마지막 단계로는 sin과 cos을 통해 선분이 가로와 세로의 직선으로 나누어 count를 진행하고 탐색된 선의 수로 체커판을 판정하였습니다.

핵심 코드

# 이미지 전처리

edge = cv.Canny(src, 100, 200)

# Hough 변환 파라미터 조정

lines = cv.HoughLines(edge, 1, math.pi / 180, 130)

# 가까운 두 선의 거리를 계산

distance = np.sqrt((x0 - x0\_m) \*\* 2 + (y0 - y0\_m) \*\* 2)

# 만약 두 선이 30 픽셀 이내에 있다면, 이 두 선을 병합

if distance < 30:

merge = True

break

a = np.cos(theta); x0 = a \* rho

b = np.sin(theta); y0 = b \* rho

pt1 = (int(x0 + 1000 \* (-b)), int(y0 + 1000 \* (a)))

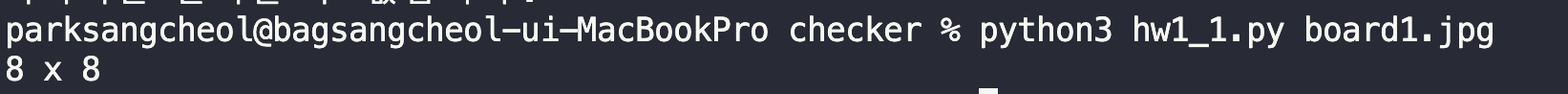
pt2 = (int(x0 - 1000 \* (-b)), int(y0 - 1000 \* (a)))

결과

스크린샷, 패턴, 원, 사각형이(가) 표시된 사진

자동 생성된 설명 스크린샷, 패턴, 라인, 사각형이(가) 표시된 사진

자동 생성된 설명



과제 2

구현 방법

onMouse를 통해 네점을 받아오도록 한후 네점의 좌표를 합과 차 연산을 통해 좌우상하 좌표로 나누었습니다 (시계방향으로 주어진다는 조건 때문에 해당 과정이 필요없으니 2,3을 한번에 구현하려다보니 해당 코드가 생성됨) 이후 투시변환 함수를 호출하여 해결했습니다.

핵심 로직

topLeft = pts[np.argmin(sm)] # x+y가 가장 작은 값이 좌상단 좌표

topRight = pts[np.argmin(diff)] # x-y가 가장 작은 것이 우상단 좌표

bottomRight = pts[np.argmax(sm)] # x+y가 가장 큰 값이 우하단 좌표

bottomLeft = pts[np.argmax(diff)] # x-y가 가장 큰 값이 좌하단 좌표

# 변환 후 4개 좌표

pts2 = np.float32([[0, 0], [width - 1, 0],[width - 1, height - 1], [0, height - 1]])

# 변환 행렬 계산

mtrx = cv.getPerspectiveTransform(pts1, pts2)

# 투시 변환 적용

result = cv.warpPerspective(*param*, mtrx, (width, height))

결과

체스, 사각형, 보드게임, 실내 게임 및 스포츠이(가) 표시된 사진

자동 생성된 설명

과제 3

구현 방법

해당 코드의 일부 부분을 다른 과제에 적용하기 위해 해당 과제는 이미지 인식 및 각 기능을 함수별로 세세히 분리하였습니다. 먼저 불러온 이미지를 화소처리하여 선명하게하여 Canny로 모든 꼭짓점을 검출하였습니다. 검출한 모든 꼭짓점에서 합과 차 연산을 통해 가장 좌우측 상하단의 좌표 4개를 가져와 투시변환을 진행하였습니다.

핵심 로직

def detect\_edges(*image*, *canny\_low*, *canny\_high*):

return cv.Canny(*image*, *canny\_low*, *canny\_high*)

def classify\_four\_points(*points*):

top\_left = min(*points*, *key*=lambda *point*: *point*[0] + *point*[1])

top\_right = max(*points*, *key*=lambda *point*: *point*[0] - *point*[1])

bottom\_left = min(*points*, *key*=lambda *point*: *point*[0] - *point*[1])

bottom\_right = max(*points*, *key*=lambda *point*: *point*[0] + *point*[1])

return top\_left, top\_right, bottom\_right, bottom\_left

def hough\_transform(*image*, *hough\_threshold*, *hough\_min\_len*, *hough\_min\_gap*):

lines = cv.HoughLinesP(*image*, 1, math.pi / 180, *hough\_threshold*, *minLineLength*=*hough\_min\_len*, *maxLineGap*=*hough\_min\_gap*)

return lines

def warp\_perspective(*image*, *src\_pts*, *width*, *height*):

pers\_mat = cv.getPerspectiveTransform(np.array(*src\_pts*).astype(np.float32), np.array([[0, 0], [*width*-1, 0], [*width*-1, *height*-1], [0, *height*-1]]).astype(np.float32))

result = cv.warpPerspective(*image*, pers\_mat, (*width*, *height*))

return result

결과

사각형, 체스, 보드게임, 직사각형이(가) 표시된 사진

자동 생성된 설명

과제 4

구현 방법

해당 과정에서 많이 누워있는 케이스에서 원 검출이 명확하지 않아 3번 과제의 코드를 이용하여 투시변환을 먼저 시행해 주었습니다. 원 검출과정에서 잡음을 인식하는 현상이 잦아 양방향 필터와 중앙값 필터를 모두 사용하여 잡음을 최대한 제거하여 HoughCircle 함수로 원들을 찾아 내었습니다, 이후 찾은 원의 중심점의 좌표를 평균으로 내어 평균보다 작은 값을 가지면 W 크면 B로 분류하였습니다.

핵심 로직

circles = cv.HoughCircles(median\_filtered, cv.HOUGH\_GRADIENT, 1, 50, *param1*=140, *param2*=12, *minRadius*= 21, *maxRadius*= 30)

dst = cv.cvtColor(src, cv.COLOR\_GRAY2BGR)

if circles is not None:

circles = np.uint16(np.around(circles)) # 소수점 좌표를 정수로 변환

for i in circles[0, :]:

cx, cy, radius = i

cv.circle(dst, (cx, cy), radius, (0, 0, 255), 2, cv.LINE\_AA)

median\_bright += median\_filtered[cy,cx]

median\_bright /= circles.shape[1]

결과

체커, 스크린샷, 실내 게임 및 스포츠, 게임이(가) 표시된 사진

자동 생성된 설명