**차선 검출**

lane1 이미지 먼저 테스트

import numpy as np

import cv2 as cv

import math

img1=cv.imread('lane1.jpeg',cv.IMREAD\_GRAYSCALE)

if img1 is None:

    print('file load failed')

edge1=cv.Canny(img1,50,150)

lines1=cv.HoughLines(edge1,1,math.pi/180,150) #겹친 게 150개 이상인 아이만

dst1=cv.cvtColor(edge1,cv.COLOR\_GRAY2BGR)

if lines1 is not None:

    for i in range(lines1.shape[0]):

        rho = lines1[i][0][0]

        theta = lines1[i][0][1]

        # 각도 계산

        angle = np.degrees(theta) % 180

        print('angle',angle)

        cos\_t = math.cos(theta)

        sin\_t = math.sin(theta)

        x0, y0 = rho \* cos\_t, rho \* sin\_t

        alpha = 1000

        pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

        pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

        cv.line(dst1, pt1, pt2, (0, 0, 255), 2, cv.LINE\_AA)

cv.imshow('dst1',dst1)

cv.waitKey()

cv.destroyAllWindows()

우선은 검출 정확도보다 검출된 차선의 각도를 확인하기 위해서 임계값을 150으로 설정

angle을 출력하도록 코드에 추가

코드 실행하면

스크린샷, 라인, 다채로움, 텍스트이(가) 표시된 사진

자동 생성된 설명

>>

검출된 여러 개의 차선 중 내가 원하는 것은 차선 2개

그 중에서도 내가 원하는 차선 하나가 겹치지 않고 출력되었으므로

출력된 angle 중에서 값이 다른 값들과 차이가 많이 나는 것을 찾음

angle 56.999996185302734

이것으로 예상됨

따라서 angle을 56<=angle<=57로 설정해서 실행해보면

if lines1 is not None:

    for i in range(lines1.shape[0]):

        rho = lines1[i][0][0]

        theta = lines1[i][0][1]

        # 각도 계산

        angle = np.degrees(theta) % 180

        if 56 <= angle <= 57:

            cos\_t = math.cos(theta)

            sin\_t = math.sin(theta)

            x0, y0 = rho \* cos\_t, rho \* sin\_t

            alpha = 1000

            pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

            pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

            cv.line(dst1, pt1, pt2, (0, 0, 255), 2, cv.LINE\_AA)

텍스트, 스크린샷, 라인, 지도이(가) 표시된 사진

자동 생성된 설명

원하는 차선이 검출됨

>>

나머지 차선도 찾기 위해서 출력된 angle 값 중 120.0을 넘는 값에서 찾아보기로 함

if 120.0 <= angle <= 123.0:

            print('angle', angle)  # 출력

위와 같이 코드를 수정한 뒤에 실행시키면

텍스트, 스크린샷, 라인, 지도이(가) 표시된 사진

자동 생성된 설명

angle 122.0

angle 122.99999237060547

angle 120.99999237060547

검출된 3개의 차선 중 첫 번째 차선을 선택한 뒤 최종 코드

if lines1 is not None:

    for i in range(lines1.shape[0]):

        rho = lines1[i][0][0]

        theta = lines1[i][0][1]

        # 각도 계산

        angle = np.degrees(theta) % 180

        if 56 <= angle <= 57 or angle == 122.0:

            cos\_t = math.cos(theta)

            sin\_t = math.sin(theta)

            x0, y0 = rho \* cos\_t, rho \* sin\_t

            alpha = 1000

            pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

            pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

            cv.line(dst1, pt1, pt2, (0, 0, 255), 2, cv.LINE\_AA)

스크린샷, 텍스트, 라인, 도표이(가) 표시된 사진

자동 생성된 설명

>>

lane2 이미지도 이와 같은 과정으로 차선을 검출

import numpy as np

import cv2 as cv

import math

img2=cv.imread('lane2.jpeg',cv.IMREAD\_GRAYSCALE)

if img2 is None:

    print('file load failed')

edge2=cv.Canny(img2,50,150)

lines2=cv.HoughLines(edge2,1,math.pi/180,150)

dst2=cv.cvtColor(edge2,cv.COLOR\_GRAY2BGR)

if lines2 is not None:

    for i in range(lines2.shape[0]):

        rho = lines2[i][0][0]

        theta = lines2[i][0][1]

        # 각도 계산

        angle = np.degrees(theta) % 180

        print('angle',angle)

        cos\_t = math.cos(theta)

        sin\_t = math.sin(theta)

        x0, y0 = rho \* cos\_t, rho \* sin\_t

        alpha = 1000

        pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

        pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

        cv.line(dst2, pt1, pt2, (0, 0, 255), 2, cv.LINE\_AA)

cv.imshow('dst2',dst2)

cv.waitKey()

cv.destroyAllWindows()

필요없는 직선 검출을 최소화하기 위해서 임계값을 190으로 설정, 200으로 하면 원하는 차선 검출도 안 됨

코드 실행하면

스크린샷, 라인, 다채로움, 텍스트이(가) 표시된 사진

자동 생성된 설명

>>

원하는 차선 2개 중 하나는 겹치지 않고 검출 >> 다른 값들과 비슷하지 않은 값 유추

120.0이나 122.0으로 예상

if 120.0 <= angle <= 122.0:

            print('angle', angle)  # 출력

코드에 추가한 뒤 실행

스크린샷, 텍스트, 지도, 라인이(가) 표시된 사진

자동 생성된 설명

angle 120.0

이유를 모르겠는데 angle 120.0만 검출되고 122.0는 안 보임

하지만 120.0가 내가 원하던 차선

>>

나머지 차선은 비슷한 3개의 값 중에서 찾기

51과 54 사이의 값 중 하나인 것 같음

따라서

if 51.0 <= angle <= 54.0:

            print('angle', angle)  # 출력

스크린샷, 지도, 라인, 텍스트이(가) 표시된 사진

자동 생성된 설명

angle 52.999996185302734

angle 53.999996185302734

angle 51.999996185302734

검출된 3개의 차선 중 두 번째 차선 선택하여 코드 실행시키면

if angle == 53.999996185302734:

            cos\_t = math.cos(theta)

            sin\_t = math.sin(theta)

            x0, y0 = rho \* cos\_t, rho \* sin\_t

            alpha = 1000

            pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

            pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

            cv.line(dst2, pt1, pt2, (0, 0, 255), 2, cv.LINE\_AA)

스크린샷, 지도, 텍스트, 라인이(가) 표시된 사진

자동 생성된 설명

>>

lane2 이미지에서 차선 2개 검출하는 최종 코드는

if lines2 is not None:

    for i in range(lines2.shape[0]):

        rho = lines2[i][0][0]

        theta = lines2[i][0][1]

        # 각도 계산

        angle = np.degrees(theta) % 180

        if angle == 53.999996185302734 or angle == 120.0:

            cos\_t = math.cos(theta)

            sin\_t = math.sin(theta)

            x0, y0 = rho \* cos\_t, rho \* sin\_t

            alpha = 1000

            pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

            pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

            cv.line(dst2, pt1, pt2, (0, 0, 255), 2, cv.LINE\_AA)

스크린샷, 라인, 지도, 텍스트이(가) 표시된 사진

자동 생성된 설명

lane1, lane2 이미지에서 동시에 차선 검출하고, 출력됐을 때 원본 이미지 위에 차선 표시되도록 수정한 최종 코드

import numpy as np

import cv2 as cv

import math

img1=cv.imread('lane1.jpeg',cv.IMREAD\_COLOR)

img2=cv.imread('lane2.jpeg',cv.IMREAD\_COLOR)

if img1 is None:

    print('file load failed')

if img2 is None:

    print('file load failed')

gray1=cv.cvtColor(img1,cv.COLOR\_BGR2GRAY)

gray2=cv.cvtColor(img2,cv.COLOR\_BGR2GRAY)

edge1=cv.Canny(gray1,50,150)

lines1=cv.HoughLines(edge1,1,math.pi/180,140) #겹친 게 140개 이상인 아이만

edge2=cv.Canny(gray2,50,150)

lines2=cv.HoughLines(edge2,1,math.pi/180,190)

if lines1 is not None:

    for i in range(lines1.shape[0]):

        rho = lines1[i][0][0]

        theta = lines1[i][0][1]

        # 각도 계산

        angle = np.degrees(theta) % 180

        # print('angle',angle) #angle 56.999996185302734, angle 122.0

        if 56 <= angle <= 57 or angle == 122.0:

            # print('angle', angle)  # 출력

            cos\_t = math.cos(theta)

            sin\_t = math.sin(theta)

            x0, y0 = rho \* cos\_t, rho \* sin\_t

            alpha = 1000

            pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

            pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

            cv.line(img1, pt1, pt2, (0, 0, 255), 2, cv.LINE\_AA)

if lines2 is not None:

    for i in range(lines2.shape[0]):

        rho = lines2[i][0][0]

        theta = lines2[i][0][1]

        # 각도 계산

        angle = np.degrees(theta) % 180

        # print('angle',angle) #angle 51.999996185302734, angle 120.0

        if angle == 53.999996185302734 or angle == 120.0:

            print('angle', angle)  # 출력

            cos\_t = math.cos(theta)

            sin\_t = math.sin(theta)

            x0, y0 = rho \* cos\_t, rho \* sin\_t

            alpha = 1000

            pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

            pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

            cv.line(img2, pt1, pt2, (0, 0, 255), 2, cv.LINE\_AA)

desired\_width = 640

desired\_height = 480

resized\_img1 = cv.resize(img1, (desired\_width, desired\_height))

resized\_img2 = cv.resize(img2, (desired\_width, desired\_height))

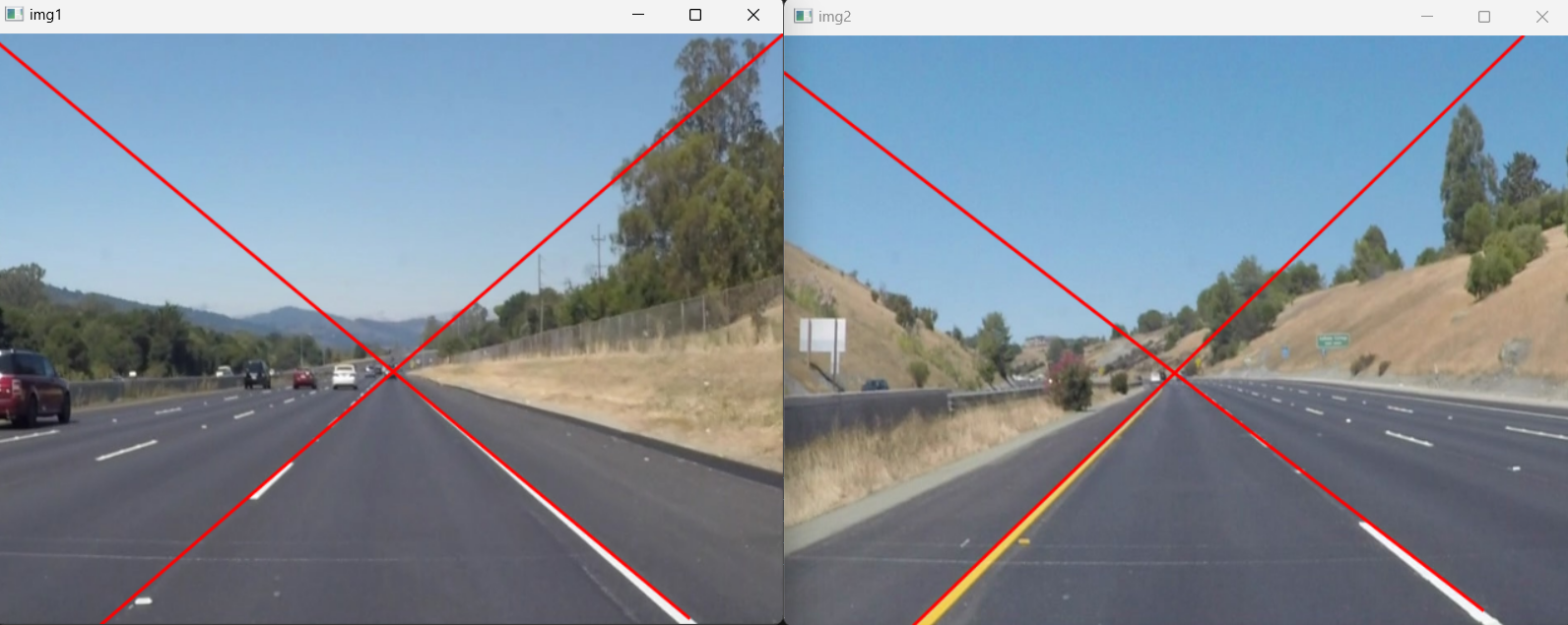
cv.imshow('img1', resized\_img1)

cv.imshow('img2', resized\_img2)

cv.waitKey(0)

cv.destroyAllWindows()

최종 출력 결과



**바둑판 돌, 바둑판 선 검출**

바둑판 선 먼저 검출

import numpy as np

import cv2 as cv

import math

img1=cv.imread('go1.jpg',cv.IMREAD\_GRAYSCALE)

img2=cv.imread('go2.jpg',cv.IMREAD\_GRAYSCALE)

if img1 is None or img2 is None:

    print('file load failed')

edge1=cv.Canny(img1,50,150)

edge2=cv.Canny(img2,50,150)

lines=cv.HoughLines(edge1,1,math.pi/180,250) #img1에서 직선 검출해서 img2에 복붙할 것

result\_img1 = cv.cvtColor(img1, cv.COLOR\_GRAY2BGR)

result\_img2 = cv.cvtColor(img2, cv.COLOR\_GRAY2BGR)

if lines is not None:

    for i in range(lines.shape[0]):

        rho=lines[i][0][0]

        theta=lines[i][0][1]

        cos\_t=math.cos(theta)

        sin\_t=math.sin(theta)

        x0,y0=rho\*cos\_t,rho\*sin\_t

        alpha=1000

        pt1=(int(x0-alpha\*sin\_t),int(y0+alpha\*cos\_t))

        pt2=(int(x0+alpha\*sin\_t),int(y0-alpha\*cos\_t))

        cv.line(result\_img1,pt1,pt2,(0,255,0),2,cv.LINE\_AA)

if lines is not None:

    for line in lines:

        rho, theta = line[0]

        cos\_t = math.cos(theta)

        sin\_t = math.sin(theta)

        x0, y0 = rho \* cos\_t, rho \* sin\_t

        alpha = 1000

        pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

        pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

        cv.line(result\_img2, pt1, pt2, (0, 255, 0), 2, cv.LINE\_AA)

cv.imshow('img1',result\_img1)

cv.imshow('img2',result\_img2)

cv.waitKey()

cv.destroyAllWindows()

go2 이미지는 바둑돌(원)이 많이 존재하기 때문에 바둑판 선 검출이 원활하게 되지 않음

따라서 go1 이미지에서 line을 가져와서 그대로 사용 >> 같은 바둑판이기 때문에 가능

BGR을 0,255,0로 설정하여 바둑판 선이 초록색으로 검출되도록 함

코드 실행하면

텍스트, 스크린샷, 라인, 그린이(가) 표시된 사진

자동 생성된 설명

>>

go1 이미지에서 바둑판 돌을 검출하기 위해 circles 코드 추가

blurred1=cv.blur(img1,(3,3))

circles1=cv.HoughCircles(img1,cv.HOUGH\_GRADIENT,1,15,param1=250,param2=70) #블러를 제거하면 더 잘 되려나? >> 아님

#해상도 비율, 최대 반지름, 임계값1(작은 값 입력 >> 더 많은 엣지 검출), 임계값2(작은 값 입력 >> 더 많은 검출)

min\_radius=1

max\_radius=15 # 이 방법 쓰면 검은 돌만 완벽하게 나옴

if circles1 is not None:

    circles1=np.uint16(np.around(circles1)) #circles 배열의 모든 값을 가장 가까운 정수로 반올림(int 대신 쓴 것)

    for i in range(circles1.shape[1]):

        cx,cy,radius=circles1[0][i]

        if min\_radius<=radius<=max\_radius:

            cv.circle(result\_img1,(cx,cy),int(radius),(0,0,255),2,cv.LINE\_AA) #검은 돌 빨간색으로 검출

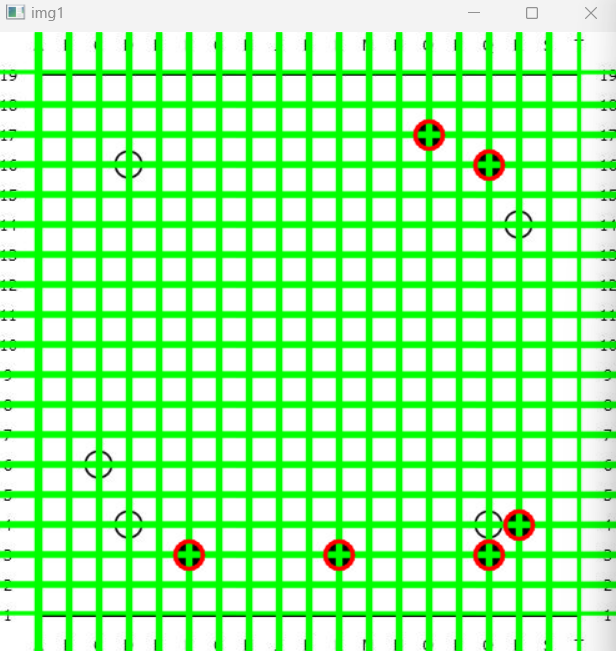
#20,150,70 >> 검은 돌 4개 일치

#10,150,70 >> 검은 돌 6개 일치

#15,250,70 >> 검은 돌 6개 일치, 지금까지 중에서는 제일 깔끔 >> 임계값 설정 과정

min, max 반지름을 각각 1과 15로 설정하여 반지름이 1보다 크고 15보다는 작은 원만 검출하도록 설정

코드 실행하면



검은 돌만 검출되었음

>>

흰 돌이 검출되지 않는 이유?

이미지의 바둑판이 흰색이라서 흰 돌과 바둑판이 잘 구분되지 않는 것으로 예상됨

위에서 설정한 임계값 대신 적응적 임계값을 사용하여 흰 돌을 검출하는 방법 사용해 봄

# 흰색 돌 검출을 위한 적응적 임계값

thresh1 = cv.adaptiveThreshold(blurred1, 255, cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv.THRESH\_BINARY\_INV, 11, 2)

circles\_1 = cv.HoughCircles(thresh1, cv.HOUGH\_GRADIENT, 1, 15, param1=100, param2=20, minRadius=1, maxRadius=15)

# 흰 돌 검출

if circles\_1 is not None:

    circles = np.uint16(np.around(circles\_1)) #circles 배열의 모든 값을 가장 가까운 정수로 반올림(int 대신 쓴 것)

    for i in circles[0, :]:

        cv.circle(result\_img1,(i[0],i[1]),int(i[2]),(255,0,0),2,cv.LINE\_AA) #흰 돌은 파란색으로 검출

cv.adaptiveThreshold를 사용하여 적응적 임계값 방식 적용

cv.ADATIVE\_THRESH\_GAUSSIAN\_C로 임계값 계산 시 주변 픽셀의 가우시안 가중치 합을 사용

cv.THRESH\_BINARY\_INV 밝은 객체를 검출할 때 사용되는 임계 방법 사용

검은 돌 검출과 마찬가지로 min\_radius를 1로, max\_radius는 15로 설정

코드 실행하면

패턴, 다채로움, 스크린샷, 직사각형이(가) 표시된 사진

자동 생성된 설명 (>> 흰 돌 검출하기까지 정말 오랜 과정이 걸렸습니다..)

>>

같은 방법으로 go2 이미지의 바둑판 돌도 검출

blurred2=cv.blur(img2,(3,3))

circles2=cv.HoughCircles(img2,cv.HOUGH\_GRADIENT,1,1,param1=250,param2=70)

if circles2 is not None:

    circles2=np.uint16(np.around(circles2)) #circles 배열의 모든 값을 가장 가까운 정수로 반올림(int 대신 쓴 것)

    for i in range(circles2.shape[1]):

        cx,cy,radius=circles2[0][i]

        if min\_radius<=radius<=max\_radius:

            cv.circle(result\_img2,(cx,cy),int(radius),(0,0,255),2,cv.LINE\_AA)

# 흰색 돌 검출을 위한 적응적 임계값

thresh2 = cv.adaptiveThreshold(blurred2, 255, cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv.THRESH\_BINARY\_INV, 11, 2)

circles\_2 = cv.HoughCircles(thresh2, cv.HOUGH\_GRADIENT, 1, 15, param1=120, param2=20, minRadius=1, maxRadius=15)

# 흰 돌 검출

if circles\_2 is not None:

    circles = np.uint16(np.around(circles\_2))

    for i in circles[0, :]:

        cv.circle(result\_img2,(i[0],i[1]),int(i[2]),(255,0,0),2,cv.LINE\_AA) #흰 돌은 파란색으로 검출

코드 실행하면

패턴, 다채로움, 라인이(가) 표시된 사진

자동 생성된 설명

흰 돌과 검은 돌에서 동시에 같은 원이 검출되었음

>>

흰 돌 검출 코드에서 minRadius 값을 높이면?

circles\_2 = cv.HoughCircles(thresh2, cv.HOUGH\_GRADIENT, 1, 20, param1=120, param2=20, minRadius=8, maxRadius=15)

패턴, 다채로움, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명 검출된 원 표시의 반지름만 커지고 중복 현상은 똑같음

>>

흰 돌을 검출하고 검은 돌을 검출하는 순서로 바꾸면?

# 흰 돌 검출

if circles\_2 is not None:

    circles = np.uint16(np.around(circles\_2))

    for i in circles[0, :]:

        cv.circle(result\_img2,(i[0],i[1]),int(i[2]),(255,0,0),2,cv.LINE\_AA)

# 검은 돌 검출

if circles2 is not None:

    circles2=np.uint16(np.around(circles2))

    for i in range(circles2.shape[1]):

        cx,cy,radius=circles2[0][i]

        if min\_radius<=radius<=max\_radius:

            cv.circle(result\_img2,(cx,cy),int(radius),(0,0,255),2,cv.LINE\_AA)

패턴, 다채로움, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명 원이 중복 검출되는 현상 해결

최종 코드는

import numpy as np

import cv2 as cv

import math

img1=cv.imread('go1.jpg',cv.IMREAD\_GRAYSCALE)

img2=cv.imread('go2.jpg',cv.IMREAD\_GRAYSCALE)

if img1 is None or img2 is None:

    print('file load failed')

edge1=cv.Canny(img1,50,150)

edge2=cv.Canny(img2,50,150)

lines=cv.HoughLines(edge1,1,math.pi/180,250) #img1에서 직선 검출해서 img2에 복붙할 것

result\_img1 = cv.cvtColor(img1, cv.COLOR\_GRAY2BGR)

result\_img2 = cv.cvtColor(img2, cv.COLOR\_GRAY2BGR)

if lines is not None:

    for i in range(lines.shape[0]):

        rho=lines[i][0][0]

        theta=lines[i][0][1]

        cos\_t=math.cos(theta)

        sin\_t=math.sin(theta)

        x0,y0=rho\*cos\_t,rho\*sin\_t

        alpha=1000

        pt1=(int(x0-alpha\*sin\_t),int(y0+alpha\*cos\_t))

        pt2=(int(x0+alpha\*sin\_t),int(y0-alpha\*cos\_t))

        cv.line(result\_img1,pt1,pt2,(0,255,0),2,cv.LINE\_AA)

if lines is not None:

    for line in lines:

        rho, theta = line[0]

        cos\_t = math.cos(theta)

        sin\_t = math.sin(theta)

        x0, y0 = rho \* cos\_t, rho \* sin\_t

        alpha = 1000

        pt1 = (int(x0 - alpha \* sin\_t), int(y0 + alpha \* cos\_t))

        pt2 = (int(x0 + alpha \* sin\_t), int(y0 - alpha \* cos\_t))

        cv.line(result\_img2, pt1, pt2, (0, 255, 0), 2, cv.LINE\_AA)

blurred1=cv.blur(img1,(3,3))

circles1=cv.HoughCircles(img1,cv.HOUGH\_GRADIENT,1,15,param1=250,param2=70) #블러를 제거하면 더 잘 되려나? >> 아님

#해상도 비율, 최대 반지름, 임계값1(작은 값 입력 >> 더 많은 엣지 검출), 임계값2(작은 값 입력 >> 더 많은 검출)

#20,150,70 >> 검은 돌 4개 일치

#10,150,70 >> 검은 돌 6개 일치

#1,250,70 >> 검은 돌 6개 일치, 지금까지 중에서는 제일 깔끔

min\_radius=1

max\_radius=15 # 이 방법 쓰면 검은 돌만 완벽하게 나옴

if circles1 is not None:

    circles1=np.uint16(np.around(circles1)) #circles 배열의 모든 값을 가장 가까운 정수로 반올림(int 대신 쓴 것)

    for i in range(circles1.shape[1]):

        cx,cy,radius=circles1[0][i]

        if min\_radius<=radius<=max\_radius:

            cv.circle(result\_img1,(cx,cy),int(radius),(0,0,255),2,cv.LINE\_AA) #검은 돌 파란색으로 검출

# 흰색 돌 검출을 위한 적응적 임계값

thresh1 = cv.adaptiveThreshold(blurred1, 255, cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv.THRESH\_BINARY\_INV, 11, 2)

# 원 검출

circles\_1 = cv.HoughCircles(thresh1, cv.HOUGH\_GRADIENT, 1, 15, param1=100, param2=20, minRadius=1, maxRadius=15)

# 흰 돌 검출

if circles\_1 is not None:

    circles = np.uint16(np.around(circles\_1)) #circles 배열의 모든 값을 가장 가까운 정수로 반올림(int 대신 쓴 것)

    for i in circles[0, :]:

        cv.circle(result\_img1,(i[0],i[1]),int(i[2]),(255,0,0),2,cv.LINE\_AA) #흰 돌 파란색으로 검출

blurred2=cv.blur(img2,(3,3))

circles2=cv.HoughCircles(img2,cv.HOUGH\_GRADIENT,1,15,param1=250,param2=70)

# 흰색 돌 검출을 위한 적응적 임계값

thresh2 = cv.adaptiveThreshold(blurred2, 255, cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv.THRESH\_BINARY\_INV, 11, 2)

circles\_2 = cv.HoughCircles(thresh2, cv.HOUGH\_GRADIENT, 1, 20, param1=120, param2=20, minRadius=4, maxRadius=15)

#이미 검출된 검은 돌도 같이 표시됨 >> 해결 필요

#minRadius를 1에서 8로 올리니까 됨

# 흰 돌 검출

if circles\_2 is not None:

    circles = np.uint16(np.around(circles\_2))

    for i in circles[0, :]:

        cv.circle(result\_img2,(i[0],i[1]),int(i[2]),(255,0,0),2,cv.LINE\_AA) #흰 돌 파란색으로 검출

# 검은 돌 검출

if circles2 is not None:

    circles2=np.uint16(np.around(circles2)) #circles 배열의 모든 값을 가장 가까운 정수로 반올림(int 대신 쓴 것)

    for i in range(circles2.shape[1]):

        cx,cy,radius=circles2[0][i]

        if min\_radius<=radius<=max\_radius:

            cv.circle(result\_img2,(cx,cy),int(radius),(0,0,255),2,cv.LINE\_AA)

cv.imshow('img1',result\_img1)

cv.imshow('img2',result\_img2)

cv.waitKey()

cv.destroyAllWindows()

최종 출력 결과는

패턴, 다채로움, 라인이(가) 표시된 사진

자동 생성된 설명