skin_detection_module.py

(정면카메라의 skin detection을 활용하기 위한 skin영역 검출 모듈)

```
import cv2
import numpy as np
import imutils



def get_skinmask(frame):  #RGB값을 이용한 피부색의mask 검출
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)      #프레임 이미지를HSV
색공간으로 변경
lowred = np.array([0, 30, 80])                    #skin color의RGB영역 설정
highred = np.array([15, 255, 255])

red_mask = cv2.inRange(hsv, lowred, highred)  # skin color의 범위

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))       #커널 구조화
red_mask = cv2.morphologyEx(red_mask, cv2.MORPH_CLOSE, kernel, 1)  #
return red_mask

def detect_skincontour(mask): #피부영역의contour 추출
contours, hierachy = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE) #피부 영역의contour 검출
contours = sorted(contours, key=lambda x: cv2.contourArea(x), reverse=True)
return contours

def draw_crossline(frame,width,height):      #프레임에 가로선 그리기
cv2.line(frame, (0, height), (width, height), (0, 255, 0), cv2.LINE_4)

def draw_verline(frame, x, height):            #프레임에 세로선 그리기
cv2.line(frame, (x, 0), (x,height),(0,0,0), cv2.LINE_4, 2)
```

piano_recognition_module.py

(피아노의 키보드를 검출하고 화면상에 print하기 위한 모듈)

```
import cv2
import numpy as np

#frame에contours를 그리는 함수
def draw_keys(frame, contours):
for i in range(len(contours)):
cv2.drawContours(frame, [contours[i]], 0, (0,255, 0), 2)

#frame에contours를 그려서 새로운 창으로 띄우는 함수
def print_keys(origianl, contours, name):
frame = origianl.copy()
for i in range(len(contours)):
cv2.drawContours(frame, [contours[i]], 0, (0,0, 255), 2)  #컨투어 프레임에 그리기
cv2.putText(frame, str(i), tuple(contours[i][0][0]), cv2.FONT_HERSHEY_COMPLEX,
0.8, (255, 0, 0), 1) #키보드 번호 텍스트 넣기
cv2.imshow("print " + name, frame)             #이미지show
return frame

#contour를scale하는 함수
```

```
https://github.com/nvs-abhilash/tutorials/blob/master/tutorials/opencv_contour_scale_rotate/Scaling%20and%20Rotating%20contours.ipynb
def scale_contour(cnt, scale):
M = cv2.moments(cnt)
cx = int(M['m10']/M['m00'])
cy = int(M['m01']/M['m00'])

cnt_norm = cnt - [cx, cy]
cnt_scaled = cnt_norm * scale
cnt_scaled = cnt_scaled + [cx, cy]
cnt_scaled = cnt_scaled.astype(np.int32)

return cnt_scaled

#contour를sort하는 함수https://pyimagesearch.com/2015/04/20/sorting-contours-using-python-and-opencv/
def sort_contours(cnts, method="left-to-right"):
        # initialize the reverse flag and sort index
        reverse = False
        i = 0

        # handle if we need to sort in reverse
        if method == "right-to-left" or method == "bottom-to-top":
                reverse = True

        # handle if we are sorting against the y-coordinate rather than
        # the x-coordinate of the bounding box
        if method == "top-to-bottom" or method == "bottom-to-top":
                i = 1
        # construct the list of bounding boxes and sort them from top to
        # bottom
        boundingBoxes = [cv2.boundingRect(c) for c in cnts]
        (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes),
                key=lambda b:b[1][i], reverse=reverse))
        # return the list of sorted contours and bounding boxes
        return (cnts, boundingBoxes)


#흰색 건반들을 찾는 함수
def finding_keys_white(frame):
key_num=7
"""""필요한 파일들 세팅"""""
#original = cv2.imread('source/piano_frame_up.jpg')
original = frame
blank = np.zeros(original.shape[:2], dtype='uint8')
blur = cv2.GaussianBlur(original, (3,3), 0)
cv2.imshow('blur', blur)
#blur = original
gray = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)
canny = cv2.Canny(gray, 120, 175)
_, thresh = cv2.threshold(gray, 130, 255, cv2.THRESH_BINARY) #binarying the image

"""""피아노(사각형) 찾아서 마스킹하기"""""
cnts, _ = cv2.findContours(canny.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]
#c = cnts[1]
c = max(cnts, key=cv2.contourArea)
cnt_scaled = scale_contour(c, 0.98)
piano_mask = cv2.drawContours(blank, [cnt_scaled], 0, 255, -1)
#original = cv2.drawContours(original, [cnt_scaled], 0, (0,0,255), 3)
#cv2.imshow('Mask', piano_mask)
#cv2.imshow('original', original)

"""""흰색 키 마스킹"""""
```

```
w_masked = cv2.bitwise_and(thresh, thresh, mask=piano_mask)
#cv2.imshow('Wmasked', w_masked)

w_cnts, _ = cv2.findContours(w_masked, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
w_cnts = sorted(w_cnts, key = cv2.contourArea, reverse = True)[:key_num]


print(len(w_cnts))
white_mask = np.zeros(original.shape[:2], dtype='uint8')

for i in range(len(w_cnts)):
white_mask = cv2.drawContours(white_mask, [w_cnts[i]], 0, 255, -1)
#cv2.imshow("White", white_mask)

"""흰색 키 왼쪽부터sort하기"""
white_key, _ = cv2.findContours(white_mask, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
print(len(white_key))
sorted_white_keys, _ = sort_contours(white_key, method="right-to-left")#[:key_num]

return sorted_white_keys


def finding_keys_black(frame):
key_num=5
"""필요한 파일들 세팅"""
#original = cv2.imread('source/piano_frame_up.jpg')
original = frame
blank = np.zeros(original.shape[:2], dtype='uint8')
blur = cv2.GaussianBlur(original, (3,3), 0)
cv2.imshow('blur', blur)
#blur = original
gray = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)
canny = cv2.Canny(gray, 120, 175)
gray = 255 - gray
_, thresh = cv2.threshold(gray, 130, 255, cv2.THRESH_BINARY) #binarying the image

"""피아노(사각형) 찾아서 마스킹하기"""
cnts, _ = cv2.findContours(canny.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]
#c = cnts[1]
c = max(cnts, key=cv2.contourArea)
cnt_scaled = scale_contour(c, 0.94)
piano_mask = cv2.drawContours(blank, [cnt_scaled], 0, 255, -1)
#original = cv2.drawContours(original, [cnt_scaled], 0, (0,0,255), 3)
#cv2.imshow('Mask', piano_mask)
#cv2.imshow('original', original)

"""흰색 키 마스킹"""
b_masked = cv2.bitwise_and(thresh, thresh, mask=piano_mask)
#cv2.imshow('Wmasked', w_masked)

b_cnts, _ = cv2.findContours(b_masked, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
b_cnts = sorted(b_cnts, key = cv2.contourArea, reverse = True)[:key_num]


print(len(b_cnts))
black_mask = np.zeros(original.shape[:2], dtype='uint8')

for i in range(len(b_cnts)):
black_mask = cv2.drawContours(black_mask, [b_cnts[i]], 0, 255, -1)
#cv2.imshow("White", white_mask)
```

```
"""흰색 키 왼쪽부터sort하기"""
black_key, _ = cv2.findContours(black_mask, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
print(len(black_key))
sorted_black_keys, _ = sort_contours(black_key, method="right-to-left")#[:key_num]

return sorted_black_keys

def finding_keys(frame):
keys = finding_keys_black(frame) + finding_keys_white(frame)
sorted, _ = sort_contours(keys, method="right-to-left")
return sorted
```

hand_recognition_and_running_2Keypoints_BlackKeys.py
(키보드 연주를 위한 모듈)

```
import cv2
import numpy as np
import pyautogui
import mediapipe as mp
import piano_recognition_module as piano
import skin_detection_module as skin
import imutils
import pygame.mixer

#연결된 음원파일을 음계와 매칭
pygame.init()
pygame.mixer.set_num_channels(50)
c4=pygame.mixer.Sound('25.mp3')
c4s=pygame.mixer.Sound('26.mp3')
d4=pygame.mixer.Sound('27.mp3')
d4s=pygame.mixer.Sound('28.mp3')
e4=pygame.mixer.Sound('29.mp3')
f4=pygame.mixer.Sound('30.mp3')
f4s=pygame.mixer.Sound('31.mp3')
g4=pygame.mixer.Sound('32.mp3')
g4s=pygame.mixer.Sound('33.mp3')
a5=pygame.mixer.Sound('34.mp3')
a5s=pygame.mixer.Sound('35.mp3')
b5=pygame.mixer.Sound('36.mp3')


#메인코드 시작
cap = cv2.VideoCapture(0)   # 위에서 아래로 키보드를 찍을 카메라
cap2 = cv2.VideoCapture(1)  # 정면에서 찍을 카메라

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)      # 화면의 크기를 설정
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

#손인식 모듈 사용
mpHands = mp.solutions.hands
my_hands = mpHands.Hands(static_image_mode=False,
max_num_hands=2,
min_detection_confidence=0.5,
```

```
min_tracking_confidence=0.5)
mpDraw = mp.solutions.drawing_utils

# 키보드 영역 산출하는 부분s키를 통해 제대로 인식될때까지 누르고 제대로
인식되면q키버튼 누르기
while True:
ret, frame = cap.read()        #카메라 화면을frame에 저장
ret2, frame2 = cap2.read()

#정면카메라 사이징
frame2 = imutils.resize(frame2, height=700, width=900)

cv2.imshow("original", frame) #1번카메라 화면 출력


skin.draw_crossline(frame2, 900, 400)
front_key = [140, 200, 240, 250, 300, 320, 342, 400, 410, 442, 490, 536, 565, 580,
630, 650, 680, 730] #전면 카메라에서 건반을 나누는 임의의 값(수동으로 맞춤)
for i in front_key:
skin.draw_verline(frame2, i, 900)

cv2.imshow("side", frame2) #2번카메라 화면 출력

if cv2.waitKey(1) == ord('s'):        #s키를 눌러 영역 검출
white_keys = piano.finding_keys_white(frame)
black_keys = piano.finding_keys_black(frame)
piano.print_keys(frame, white_keys, 'white')
piano.print_keys(frame, black_keys, 'black')
keys = piano.finding_keys(frame)
piano.print_keys(frame, keys, 'keys')



cv2.imshow("front", frame2) #검출한 화면 출력

if cv2.waitKey(1) == ord('q'): #비디오 처리용으로 변경. 'q' 키로 끝내기
print("Next")
cv2.destroyAllWindows()
break


finger = 5     #사용할 손가락 개수
w_key_num = 7 #사용할 흰건반 개수
b_key_num = 5 #사용할 흑건반 개수
key_num = w_key_num + b_key_num

# keyboard별로fingertip에 따른 부울값(영역안에 들어왔으면1, 아니면0)
# keyboard_id[키보드넘버][손끝id넘버]= 부울값(엄지부터 소지까지0,1,2,3,4 인덱스
부여)
keyboard_id= np.zeros((key_num, finger), int)

#정면카메라를통해 눌렸는지 안눌렸는지 판별하기 위한 변수
pressed= 0
#직전 좌표 저장값
pre_coordinate = 0

#이전에 키보드를 누른id를 저장하는 변수(초기값-1로 설정)
#키보드 영역안에 손끝keypoint가 들어갔을때 딱 한 번만 연주되도록 하기 위한 장치
idc = [-1,-1,-1, -1,-1,-1, -1,-1,-1, -1,-1,-1] #


#검출한 영역을 통해 연주하기 위한 화면
while True :
success, frame = cap.read()
cam2, frame2 = cap2.read()
```

```
frame2 = imutils.resize(frame2, height=700, width=900)

#########위에서 검출한 키보드 영역 그리기##########
for white_key in white_keys:
cv2.drawContours(frame, [white_key], 0, (0, 255, 0), 2)
for black_key in black_keys:
cv2.drawContours(frame, [black_key], 0, (0, 0, 255), 2)




height_limit = 400 #건반이 눌리는 임의의 높이

# 손인식 모듈

hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
results = my_hands.process(hsv)
#스킨디텍션 모듈
mask = skin.get_skinmask(frame2)  #스킨영역 마스크 검출
contours = skin.detect_skincontour(mask) # 검출한 마스크영역 컨투어 구하기
skin.draw_crossline(frame2, 900, height_limit) #가로선 그리기
for i in front_key:        #키 영역에 따른 세로선 그리기
skin.draw_verline(frame2, i, 900)

#정면카메라 인식을 통한 과정(손가락의 높이를 이용한 누르기 판별)
for cnt in contours:
cv2.drawContours(frame2, [cnt], 0, (255, 0, 0), 3) #스킨 디텍션 영역의 컨투어 그리기

#가장 낮은점 구하기 위한 절차( 키보드를 눌렀는지 판별하기 위해 구하기)
contours_min = np.argmin(contours[0], axis=0)
contours_max = np.argmax(contours[0], axis=0)
y_max = contours[0][contours_max[0][1]][0][1]  #가장 낮은위치의y좌표
x_max = contours[0][contours_max[0][1]][0][0]  #가장 낮은 위치의x좌표
cv2.circle(frame2,(x_max, y_max), 10, (255,0,0), cv2.FILLED)
print("y-Max =", y_max)
print("x-Max =", x_max)

if y_max > height_limit : #쓰레시홀드를 넘어갔을 때 연주된 것으로 판별
pressed = 1
break
elif y_max < height_limit : #쓰레시홀드를 넘지 않았을때 이전좌표와 누른 상태0으로
초기화
pre_coordinate = 0
pressed = 0
idc = [-1,-1,-1, -1,-1,-1, -1,-1,-1, -1,-1,-1]
break

break

# 위에서 아래로 찍는 카메라를 통해서 키보드영역안에 키포인트 들어왔는지 판별
if results.multi_hand_landmarks:
for handLms in results.multi_hand_landmarks:
for id, lm in enumerate(handLms.landmark):
h,w,c = frame.shape
cx, cy = int(lm.x*w), int(lm.y*h) ##손끝점의 좌표

if id == 4 or id == 8 or id == 12 or id == 16 or id == 20:  ## 손끝5개지점에
파란 동그라미그리기
cv2.circle(frame,(cx,cy), 10, (255,0,0), cv2.FILLED)

##원의 중심좌표cx,cy 이용해서 키보드 영역 안에 있을 시1값 반환하는 함수(영역
밖일시-1 반환)
for i in range(len(keys)):
result = cv2.pointPolygonTest(keys[i], (cx, cy), False)
```

```
#id값에 따라result값 저장(엄지부터 소지까지)
if id == 4:       #엄지
keyboard_id[i][0]= result
elif id == 8:     #검지
keyboard_id[i][1]= result
elif id == 12:    #중지
keyboard_id[i][2]= result
elif id == 16:    #약지
keyboard_id[i][3]= result
elif id == 20:    #소지
keyboard_id[i][4]= result


#조건 만족시 음원파일 재생
for j in range(5): #5개의 키포인트에 대해for문을통해 검사

if sum(keyboard_id[0])  > -3 or sum(keyboard_id[1]) > -3 or sum(keyboard_id[2])
> -3 or sum(keyboard_id[3]) > -3 or sum(keyboard_id[4]) > -3 or
sum(keyboard_id[5]) > -3 or sum(keyboard_id[6]) > -3 or sum(keyboard_id[7]) >
-3 or sum(keyboard_id[8]) > -3 or sum(keyboard_id[9]) > -3 or
sum(keyboard_id[10]) > -3 or sum(keyboard_id[11]) > -3 :
# 키포인트 두개이상 영역안에있을시 아무것도 실행하지 않기 위한 장치(키포인트 하나만
영역안에 있을시sum값이-3, 2개이상일시-3보다 큼)
break

# 도
if keyboard_id[0][j] == 1  and (idc[0] == -1 or idc[0] != j) and pressed == 1
and (front_key[17] > x_max and front_key[15] < x_max):
idc[0] = j
c4.play()
break
#키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[0] == j and keyboard_id[0][j] == -1 :
idc[0] = -1
break
# 도샵
elif keyboard_id[1][j] == 1  and (idc[1] == -1 or idc[1] != j) and pressed == 1
and (front_key[16] > x_max and front_key[14] < x_max):
idc[1] = j
c4s.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[1] == j and keyboard_id[1][j] == -1 :
idc[1] = -1
break
# 레
elif keyboard_id[2][j] == 1  and (idc[2] == -1 or idc[2] != j )and pressed == 1
and (front_key[15] > x_max and front_key[12] < x_max):
idc[2] = j
d4.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[2] == j and keyboard_id[2][j] == -1 :
idc[2] = -1
break
# 레샵
elif keyboard_id[3][j] == 1  and (idc[3] == -1 or idc[3] != j) and pressed == 1
and (front_key[13] > x_max and front_key[11] < x_max):
idc[3] = j
d4s.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[3] == j and keyboard_id[3][j] == -1 :
idc[4] = -1
break
```

```
# 미
elif keyboard_id[4][j] == 1  and (idc[4] == -1 or idc[4] != j) and pressed == 1
and (front_key[12] > x_max and front_key[10] < x_max):
idc[4] = j
e4.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[4] == j and keyboard_id[4][j] == -1 :
idc[4] = -1
break
# 파
elif keyboard_id[5][j] == 1  and (idc[5] == -1 or idc[5] != j) and pressed == 1
and (front_key[10] > x_max and front_key[8] < x_max):
idc[5] = j
f4.play()
break
elif idc[5] == j and keyboard_id[5][j] == -1 :
idc[5] = -1
break
#파샵
elif keyboard_id[6][j] == 1  and (idc[6] == -1 or idc[6] != j) and pressed == 1
and (front_key[9] > x_max and front_key[7] < x_max):
idc[6] = j
f4s.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[6] == j and keyboard_id[6][j] == -1 :
idc[6] = -1
break
# 솔
elif keyboard_id[7][j] == 1  and (idc[7] == -1 or idc[7] != j) and pressed == 1
and (front_key[8] > x_max and front_key[5] < x_max):
idc[7] = j
g4.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[7] == j and keyboard_id[7][j] == -1 :
idc[7] = -1
break
# 솔샵
elif keyboard_id[8][j] == 1  and (idc[8] == -1 or idc[8] != j) and pressed == 1
and (front_key[6] > x_max and front_key[4] < x_max):
idc[8] = j
g4s.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[8] == j and keyboard_id[8][j] == -1 :
idc[8] = -1
break
# 라
elif keyboard_id[9][j] == 1  and (idc[9] == -1 or idc[9] != j) and pressed == 1
and (front_key[5] > x_max and front_key[2] < x_max):
idc[9] = j
a5.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[9] == j and keyboard_id[9][j] == -1 :
idc[9] = -1
break
# 라샵
elif keyboard_id[10][j] == 1  and (idc[10] == -1 or idc[10] != j) and pressed ==
1 and (front_key[3] > x_max and front_key[1] < x_max):
idc[10] = j
a5s.play()
break
```

```
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[10] == j and keyboard_id[10][j] == -1 :
idc[10] = -1
break
# 시
elif keyboard_id[11][j] == 1  and (idc[11] == -1 or idc[11] != j) and pressed ==
1 and (front_key[2] > x_max and front_key[1] < x_max):
idc[11] = j
b5.play()
break
# 키포인트가 건반영역밖에 나갔을시 초기화(중복연주 방지를 위한 장치)
elif idc[11] == j and keyboard_id[11][j] == -1 :
idc[11] = -1
break

cv2.imshow("side", frame2)
cv2.imshow("up", frame)
cv2.waitKey(1)
```