연구 주제: 눈동자 인식을 이용한 화상 마우스/키보드

연구 목표: 시선-마우스 좌표의 matching 정확도 개선 및 계속하여 눈동자 인식률 향상 시도

연구 요약:

0. 최근 연구까지의 진척도 :

실시간 웹캠으로부터 안면의 시선 좌표를 인식하여 마우스 포인터를 움직이는 프로그램의 대략적인 “뼈대”까지 제작

1. 매 frame 좌표 대응점보다 벡터지향적으로 변경

2. 마우스 움직임이 튐은 방지하기 위해 마우스 민감도(sensitivity) , 움직임 임계값(threshold_dis) 파라미터를 새로이 추가

5 월 25 일

국민대학교 전자공학부 전자종합설계 I

연구원: 김정민(20162934) , 신미나(20201463)

연구 내용

개선이 필요한 부분

- 지난 연구와 마찬가지로, (1)시선의 인식률 향상과, (2)마우스 좌표와의 적절한 maching 문제에 대한 개선방안이 필요했다.

1. 시선 인식률 향샹 ?
- [1] 해상도 높은 웹캠을 마련하기
- [2] 영상처리 기법 바꾸기

2. 시선좌표 - 마우스좌표의 matching 정확도 향상 ?
- [1] 사용자가 마우스 민감도(sensitivity)파라미터를 새로이 추가하기 :
  장점: 마우스장치에 있는 민감도처럼 사용자가 시선 좌표에 따른 [마우스포인터의 이동 비율 = 민감도]를 조절하여 정확도 올릴 수 있음
  단점: 실제로 사용자의 안면과 카메라 사이의 거리가 변화할 때마다 최적의 민감도도 변하게 됨. 사용자가 자주 파라미터를 조절하는 것은 실용적이지 못함.
- [2] 동기화 기능 추가하기 // (실행 시 n 초간 화면의 정중앙을 응시하게 강제하여 시선좌표와 마우스좌표를 영점보정.)
  장점: 위 1,2 번의 기능의 밑바탕으로 들어갈 수 있음. 안면의 거리, 각도 등 환경요인이 바뀌어도 matching point 의 정확도 챙기는 게 가능
  단점: 실용적이지 못함(불편함)

개선 연구 과정

1. 눈동자 검출과정 수정

관련 연구자료를 참고하여, 기존의 전처리+HoughCircles 검출 과정에서 ->
전처리+이미지이진화+검정색영역(0)에 그릴 수 있는 최대 크기의
타원(ellipse)검출과정으로 변경하였다.

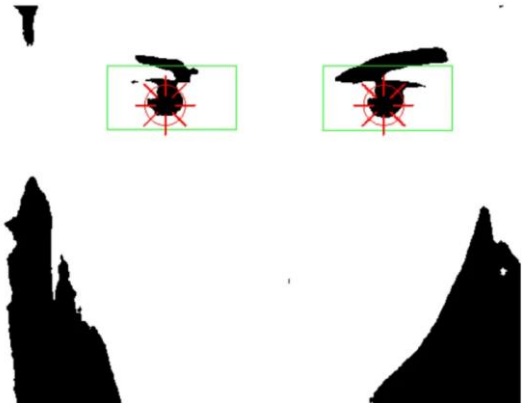

Figure 1: 관련 연구지의 그림자료

Figure 2: 실행 후 crop&binarization 된 우측 눈 이미지

위 figure2 는 dlib 로 눈 영역을 사각형으로 자른 후 이진화를 진행하여 얻은 이미지이다.

이 후, 흑과 백으로만 이루어진 위 **이진화이미지에서 검정색(0)에 해당하는 영역에 포함될 수 있는 가장 큰 사이즈의 타원(원)객체를** 찾도록 하였다. 지난 연구과정까지의 검출법은 오직 원(Circles)만 찾는 것을 기준으로 하였으나 시선이 위,아래를 향할 경우에 원모양보다 타원모양에 가까워지기 때문에 이번 연구부터 타원(Ellipses)객체를 찾는 것을 기준으로 세웠다.

2. 마우스 좌표 매칭을 벡터지향적으로 수정

매 frame 마다 <시선의 좌표 ⇔ 화면 上 마우스의 좌표> 처럼 매칭시키는 방식에서, 이전 frame 에서 현 frame 간 시선좌표가 이동한 정도 ⇔ 마우스의 이동거리와 방향 으로 수정하였다. 현재 frame 의 시선좌표 - 이전 frame 의 시선좌표 = (dx, dy) 라는 벡터정보를 저장하여 벡터의 길이가 최소 임계값(threshold_dis)미만일 경우 이전 frame 의 마우스 좌표를 그대로 이어가며 어떤 동작도 하지 않은 채 정지하고, 임계값 이상일 경우 현 frame 의 마우스 좌표를 갱신하여 마우스를 동작하는 방식으로 코드를 고쳤다. o

3. 동기화 함수 & 민감도 등 파라미터 추가

```
def init_point():
    pyautogui.moveTo(target_x_init, target_y_init)
    time.sleep(1)
```

동기화 기능은 위와 같이 단순하게 마우스의 현재 좌표를 화면의 정 중앙으로 위치시키는 함수이다. 프로그램 실행 중 키보드로 'I' 를 누르면 1 초동안 마우스 좌표가 초기화된다.

민감도(sensitivity)파라미터는 컴퓨터의 마우스 민감도와 동일한 기능을 수행한다.

```
if left_center is not None and right_center is not None :
    # 좌표는 crop이미지 기준 -> 전체frame이미지 기준 좌표로 transform
    left_new_center = (left_center[0] + left_eye_pts_np[0][0], left_center[1] +
    right_new_center = (right_center[0] + right_eye_pts_np[0][0], right_center[1

    # 이전좌표 차이값과 임계값 비교
    px_prev, py_prev = px, py # 이전 frame의 시선좌표 저장
    px = (left_new_center[0] + right_new_center[0]) // 2 # 현제 frame의 시선좌표
    py = (left_new_center[1] + right_new_center[1]) // 2

    dx, dy = px - px_prev, py - py_prev
    distance = dx**2 + dy**2

    if(distance < threshold_dis):
        dx, dy = 0, 0

current_x, current_y = pyautogui.position()
new_x = current_x + (dx * sensitivity)
new_y = current_y + (dy * sensitivity)
pyautogui.moveTo(new_x, new_y)
```

Figure 3: 시선벡터:마우스벡터의 mapping 과정

위 코드는 양 눈의 시선좌표가 모두 검출됐을 경우 이전 frame 과의 차이값을 구하고 최소 거리 임계값 이상일 경우에만 갱신하여 마우스 좌표계로 보낸다. 이전 연구처럼 시선 바운더리와 모니터 사이즈의 비율을 계산하는 것보다 더 단순하게 마우스 좌표값을 얻을 수 있다. 민감도 파라미터를 사용자가 적절히 조절하여 정확도를 향상시킬 수 있을 것이다.

```
threshold_bin = 100 # 이진화 임계값 설정
minimum_size, maximum_size = 5, 1000 # 눈동자 크기 바운더리값
threshold_dis = 100 # 좌표 차이 임계값
sensitivity = 15 # 민감도
output_width, output_height = 1920, 1080 # 출력창 크기 설정
target_x_init, target_y_init = int(output_width/2) , int(output_height/2) # target 좌표
dx = dy = px = py = px_prev = py_prev = 0

1.5s
```

Figure 4: 코드의 여러 파라미터를 추가하였다.

참고 문헌

[1] A 2D eye gaze estimation system with low-resolution webcam images / Ibrahim Furkan Ince & Jin Woo Kim / https://asp-eurasipjournals.springeropen.com/articles/10.1186/1687-6180-2011-40