컴퓨터그래픽스 7주차 과제 보고서
201300995 이상건 물리학과

구현 내용
color histogram을 이용해 tracking을 하는 것이다.

이유(구현 방법)
일단 각각의 R,G,B를 8등분의 historgram으로 나눠 총 24개의 배열로 만들어서 계산하는 법부터 살펴보겠다.

```
# histogram을 계산하고, 각 histogram간 거리를 계산.
# 거리가 최소가 되는 지점의 좌표 4개를 coord에 저장한다.
srcoffsetimageyxhist = np.zeros((src.shape[0],src.shape[1],8*3),dtype=int)
srcoffsetimageyxhist[offset_y-5:offset_y+5+ty,offset_x-5:offset_x+5+tx,:] = srcimageyx_hist(src[offset_y-5:offset_y+5+ty,offset_x-5:offset_x+5+tx,:])
```

```
#color histogram 생성.
def srcimageyx_hist(fr):
    '''
    :param fr: histogram을 구하고자 하는 대상 영역
    :return: fr의 color histogram
    '''

    # Histogram을 계산해 주세요.

    imagehist = np.zeros((fr.shape[0],fr.shape[1],8*3),dtype=int)
    for i in range(0,imagehist.shape[0]):
        for j in range(0,imagehist.shape[1]):
            histogramR = np.histogram(fr[i,j,0],bins=8,range=(0,255))[0]
            histogramG = np.histogram(fr[i,j,1], bins=8,range=(0,255))[0]
            histogramB = np.histogram(fr[i,j,2], bins=8,range=(0,255))[0]
            imagehist[i,j] = np.concatenate((histogramR, histogramG, histogramB))

    return imagehist
```

일단 이미지 단위로 수행을 하는게 좋을 것 같아서, offset으로 주변을 검사하는 크기만큼의 이미지 부분의 color histogram을 미리 검사해서 좌표값의 정보도 함께 저장하는 (y좌표크기,x좌표크기,24) 형태의 배열부터 만들었다. 이를 위해 srcimageyx_hist 함수를 따로 만들었다. 함수를 보면 각각의 R,G,B에 대해 8등분의 histogram을 수행하고 합치는 과정을 각 좌표마다 수행하는 것을 알 수 있다. 그래서 검사하는 부분(offsetY-20:offsetY+20,offsetX-20:offsetX+20) 의 color histogram을 각각의 좌표에 대해 구했기 때문에 한 범위 내의 color histogram을 구하고 싶다면 그 부분의 color histogram을 다 더하기만 하면 된다.

```
sourcehist = my_hist(target)
# sourcehist = my_divHist(target)
```

```python
#color histogram 생성.
def my_hist(fr):
    '''
    :param fr: histogram을 구하고자 하는 대상 영역
    :return: fr의 color histogram
    '''

    # Histogram을 계산해 주세요.
    histogramR = np.histogram(fr[:, :, 0], bins=8, range=(0, 255))[0]
    histogramG = np.histogram(fr[:, :, 1], bins=8, range=(0, 255))[0]
    histogramB = np.histogram(fr[:, :, 2], bins=8, range=(0, 255))[0]

    hist = np.concatenate((histogramR, histogramG, histogramB))

    return hist
```

target 부분의 color historgram은 histogram 함수를 이용해서 구한다. target의 color histogram을 위에서 쓴 srcimageyx_hist 로 계산한 검사하는 이미지 부분의 histogram을 사용하고 싶었지만 target에선 해당 부분의 RGB 정보만 있고 전체 이미지의 y,x 좌표는 나와 있지 않기 때문에 쓸 수가 없었다.

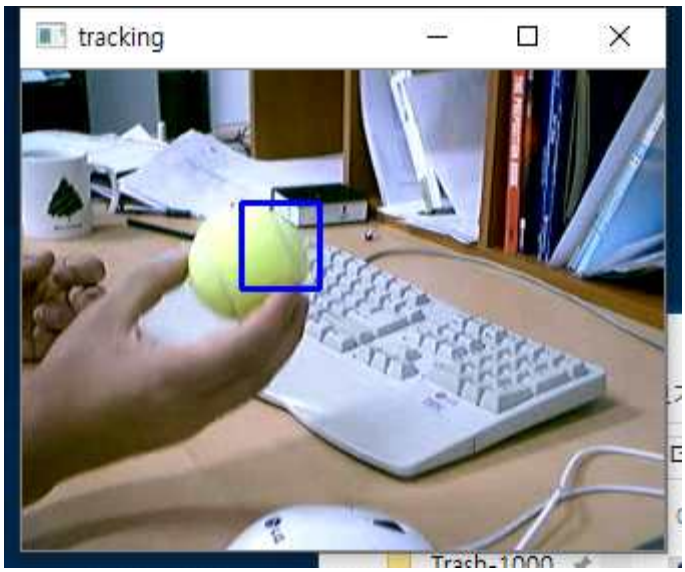
```python
for i in range(offset_y-20, offset_y+20):
    for j in range(offset_x-20, offset_x+20):                #이전
        # my_hist 함수를 쓸 땐 이 밑을 사용
        histr = srcoffsetimageyxhist[i:i+ty,j:j+tx].sum(axis=0)
        calhist = histr.sum(axis=0)
```

검사하는 부분의 histogram은 위에서 srcimageyx_hist 함수에서 계산한 것을 이용한다. 이미 (y좌표,x좌표,24)로 해당 color histogram을 구했으므로 다 더하기만 하면 된다. 그럼 계산된 calhist가 나온다.

```python
diffhistlist = sourcehist - calhist
diffhistlistsquare = diffhistlist ** 2
sumdiffhistlistsquare = sum(diffhistlistsquare)
plushistlist = sourcehist + calhist
sumplushistlist = sum(plushistlist)
divsumhistlistsquaresumplushistlist = sumdiffhistlistsquare / sumplushistlist
if(divsumhistlistsquaresumplushistlist < min):
    min = divsumhistlistsquaresumplushistlist
    coord = (j,i,j+tx,i+ty)
```

이렇게 계산된 calhist와 target의 color histogram간의 차이를 Chi square 방식을 이용한다. 그대로 적용한 것 뿐이다. 그렇게 거리를 구해서 제일 작은 거리가 나왔을 때 그 곳의

coord를 저장한다.



그 결과 잘 실행되었다.

그럼 24크기 배열이 아니라 9등분으로 나눠서 계산하여 216 크기로 계산하는 divHist를 보겠다.

```
    # sourcehist = my_hist(target)
    sourcehist = my_divHist(target)
```

```
def my_divHist(fr):
    '''
    :param fr: 3x3 (9) 등분으로 분할하여 Histogram을 계산할 이미지.
    :return: length (216) 혹은 (216,1) array ( histogram )
    '''
    y, x = fr.shape[0], fr.shape[1]
    div = 3 # 3x3 분할
    divY, divX = y // div, x // div # 3등분 된 offset 계산.

    # cell 단위의 histogram을 계산하기 위해 필요한 작업 및 계산을 수행하세요.
    hist = np.zeros((0,),dtype=int)
    for i in range(div):
        for j in range(div):
            histogramR = np.histogram(fr[divY*i:divY*(i+1), divX*i:divX*(j+1), 0], bins=8, range=(0, 255))[0]
            histogramG = np.histogram(fr[divY*i:divY*(i+1), divX*i:divX*(j+1), 1], bins=8, range=(0, 255))[0]
            histogramB = np.histogram(fr[divY*i:divY*(i+1), divX*i:divX*(j+1), 2], bins=8, range=(0, 255))[0]
            hist = np.concatenate((hist,histogramR, histogramG, histogramB))
    # 여기까지

    return hist
```

일단 target의 div color histogram을 구하는 부분이다. 3*3 구간으로 나누고 각각의 RGB를 앞에서 color histogram처럼 구한다음 이어붙인다.

그런데 이렇게 하면 문제가 생기는데, 앞에서 offset 검사 범위의 color histogram은 24 크기의 color histogram 이라고 가정했기 때문에 앞에서 했던 것처럼 그 범위 전체를 더하기만 해서는 안된다. 검사부분만 따로 분리해서 구역을 3*3으로 나눠 216으로 바꾸고 붙이는 과정이 필요하다. 이걸 수행했다.

```
# my_divHist 함수를 쓸 땐 이 밑을 사용
div = 3  # 3x3 분할
divY, divX = ty // div, tx // div  # 3등분 된 offset 계산.
histd = np.zeros((0,), dtype=int)
for k in range(div):
    for l in range(div):
        histogramRGB = srcoffsetimageyxhist[i+divY*k:i+divY*(k+1),j+divX*k:j+divY*(l+1)].sum(axis=0)
        histd = np.concatenate((histd,histogramRGB.sum(axis=0)))
calhist = histd
```

이미 24크기의 color histogram은 구했으니까 구간 잘라서 붙이기만 했다.

그 뒤는 앞에서 했던 그냥 color histogram으로 구할 때와 같다.

느낀 점
어렵다.
이미지단위가 애매해서 발상하는데 시간이 많이 걸렸다.

과제 난이도
어렵다.