

Computer Graphics Vision
Key Point Matching 기반 객체추적

E조

2016125045 윤나라

2016125055 이우리

2016125008 김수연

Key Point Matching 기반 객체추적 개요

GOAL

- 마우스로 추적할 객체 선택
 - 실시간으로 영상 속 객체
노란색으로 표시
 - 실행 속도 향상을 위한 최적화
-
- ✓ 클릭된 위치 주변의 key point 기반으로
객체 추적
 - ✓ 탐색된 key point와 매치된 key point에
대하여 각각 파랑색과 빨강색
원으로 표시

01 소스코드

02 코드분석

02-1 실행속도 최적화

04 결과화면

Key Point Matching 기반 객체추적 소스코드

```
import cv2
import numpy as np
import math
```

```
def mouse_callback(event, x, y, flags, param): #추적하고 싶은 물체 지정
    global pause, frame, sx, sy, roi
```

```
    if pause==False and event==1: #멈추고 있고, 마우스 눌렀을때 시작위치
        sx, sy = x, y
```

```
    elif pause==False and event == 4: #멈추고 있고 마우스 땔때 마지막 위치
        roi = frame[sy:y, sx:x]
        cv2.imshow('roi', roi) #지정한 부분 새창으로 띄움
        pause = True #동영상 시작하도록 변수 True로 바꾸어줌
```

```
windowName = "Find Matching Video"
cv2.namedWindow(windowName)
pause = False
```

```
def main():
    global frame, roi, pause
```

```
    # SIFT선언, 루프 안에서 선언하면 계산의 낭비이므로 밖에서 미리 정해주고 들어감
    sift = cv2.xfeatures2d.SIFT_create()
    FLANN_INDEX_KDTREE = 0
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
    search_params = dict(checks=50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)
```

```
    #동영상 읽기
    filename = "1st_school_tour.mp4"
    cap = cv2.VideoCapture(filename)
```

```
    if cap.isOpened():
        ret, frame = cap.read()
    else:
        ret = False
```

```
    # 동영상 크기 조정
```

```
    ret, frame = cap.read()
    frame = cv2.resize(frame, (640, 360), interpolation=cv2.INTER_AREA)
    cv2.imshow(windowName, frame)
```

```
    while ret:
        if pause: #pause가 true일때만 동영상 재생
            ret, frame = cap.read()
            frame = cv2.resize(frame, (640, 360), interpolation=cv2.INTER_AREA)
```

```
        # 매칭 계산 시작
        kp1, des1 = sift.detectAndCompute(roi, None)
        kp2, des2 = sift.detectAndCompute(frame, None)
```

```
        matches = flann.knnMatch(des1, des2, k=2)
```

```
        good = []
        for m,n in matches:
            if m.distance < 0.6*n.distance:
                good.append(m)
```

```
        if len(good)==0: # 만약 재생중인 동영상에 지정한 부분과 매칭되는 객체가
            더이상 없으면
```

```
            pause=False # pause가 False이므로 영상이 멈추고 다시 지정할 수 있음
            cv2.destroyWindow('roi') #멈추면 roi창과
            cv2.destroyWindow('result') #매칭result창이 꺼지도록
```

```
        if len(good)>4:
            src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)
            dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)
```

```
            M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
```

```
            if M is not None :
                h, w = roi.shape[:2]
                pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
                dst = cv2.perspectiveTransform(pts,M)
                pt1=(dst[0][0][0], dst[0][0][1]) # 구한 매칭의 양 대각선 꼭짓점
                pt2=(dst[2][0][0], dst[2][0][1])
                mid = ((int)((pt1[0] + pt2[0])/2), (int)((pt1[1] + pt2[1])/2)) #중점
                rad = (int)(math.sqrt(math.pow(pt1[0]-pt2[0], 2) + math.pow(pt1[1]-
                pt2[1], 2)) / 2) #반지름
                #area = w*h
```

```
                if abs(pt1[0]-pt2[0]) > (1.2*w) or abs(pt1[1]-pt2[1])>(1.2*h): #기존의 매
                    치와 너무 다른 비율이 detection되면 오류로 처리하고 표시하지 않음
```

```
                    pass
                else:
                    frame = cv2.circle(frame, mid, rad, (0, 255, 255), 2) # detection 성
                    공
                else:
                    pass
```

```
            img3 = cv2.drawMatches(roi, kp1, frame, kp2, good, None, (0, 0, 255),
            flags=2) #매칭 키포인트 연결선 보이게 출력
            cv2.imshow('result', img3)
            cv2.imshow(windowName, frame)
```

```
            if cv2.waitKey(1) == 27: #종
                break
```

```
        cv2.destroyAllWindows()
        cap.release()
```

```
    cv2.setMouseCallback(windowName, mouse_callback)
```

```
if __name__ == "__main__":
    main()
```

Key Point Matching 기반 객체추적 코드분석, 실행속도 최적화

```
def mouse_callback(event, x, y, flags, param): #추적하고 싶은 물체 지정
    global pause, frame, sx, sy, roi

    if pause==False and event==1: #멈추고 있고, 마우스 눌렀을때 시작위치
        sx, sy = x, y

    elif pause==False and event == 4: #멈추고 있고 마우스 땔때 마지막 위치
        roi = frame[sy:y, sx:x]
        cv2.imshow('roi', roi) #지정한 부분 새창으로 띄움
        pause = True #동영상 시작하도록 변수 True로 바꾸어줌

windowName = "Find Matching Video"
cv2.namedWindow(windowName)
pause = False
```

- mouse_callback():
멈춰있는 영상에서 마우스이벤트로 객체를 선택하고 선택된 부분을 'roi'라는 새 창으로 띄운다.

- Sift를 사용함으로써 이미지의 크기와 회전에 불변하는 특징들을 추출할 수 있다.

```
def main():
    global frame, roi, pause

    # SIFT선언, 루프 안에서 선언하면 계산의 낭비이므로 밖에서 미리 정해주고 들어감
    sift = cv2.xfeatures2d.SIFT_create()
    FLANN_INDEX_KDTREE = 0
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
    search_params = dict(checks=50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)

    #동영상 읽기
    filename = "1st_school_tour.mp4"
    cap = cv2.VideoCapture(filename)

    if cap.isOpened():
        ret, frame = cap.read()
    else:
        ret = False

    # 동영상 크기 조정
    ret, frame = cap.read()
    frame = cv2.resize(frame, (640, 360), interpolation=cv2.INTER_AREA)
    cv2.imshow(windowName, frame)
```

- cv2.resize():
실행속도 향상을 위해 동영상 크기 조정을 한다.
- cv2.INTER_AREA:
가장 많이 사용되는 보간법으로 사이즈를 줄이기 위해 사용한다.

Key Point Matching 기반 객체추적 코드분석

```
while ret:
    if pause: #pause가 true일때만 동영상 재생
        ret, frame = cap.read()
        frame = cv2.resize(frame, (640, 360), interpolation=cv2.INTER_AREA)

        # 매칭 계산 시작
        kp1, des1 = sift.detectAndCompute(roi, None)
        kp2, des2 = sift.detectAndCompute(frame, None)

        matches = flann.knnMatch(des1, des2, k=2)

        good = []
        for m,n in matches:
            if m.distance < 0.6*n.distance:
                good.append(m)

        if len(good)==0: # 만약 재생중인 동영상에 지정된 부분과 매칭되는 객체가 더이상 없으면
            pause=False # pause가 False이므로 영상이 끝나고 다시 지정할 수 있을
            cv2.destroyWindow('roi') #얼추면 roi창과
            cv2.destroyWindow('result') #매칭result창이 꺼지도록

        if len(good)>4:
            src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape((-1,1,2))
            dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape((-1,1,2))

            M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)

            if M is not None :
                h, w = roi.shape[:2]
                pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape((-1,1,2))
                dst = cv2.perspectiveTransform(pts,M)
                pt1=(dst[0][0][0], dst[0][0][1]) # 구한 매칭의 양 대각선 꼭짓점
                pt2=(dst[2][0][0], dst[2][0][1])
                mid = ((int)((pt1[0] + pt2[0])/2), (int)((pt1[1] + pt2[1])/2)) #중점
                rad = (int)(math.sqrt(math.pow(pt1[0]-pt2[0], 2) + math.pow(pt1[1]-pt2[1], 2)) / 2) #반지름
                #area = w*h
                if abs(pt1[0]-pt2[0]) > (1.2*w) or abs(pt1[1]-pt2[1])>(1.2*h): #거론의 매칭과 너무 다른 비율이 detection되면 오류로 처리하고 표시하지 않음
                    pass
                else:
                    frame = cv2.circle(frame, mid, rad, (0, 255, 255), 2) # detection 성공
            else:
                pass

            img3 = cv2.drawMatches(roi, kp1, frame, kp2, good, None, (0, 0, 255), flags=2) #매칭 키포인트 연결선 보이게 출력
            cv2.imshow('result', img3)
            cv2.imshow(windowName, frame)

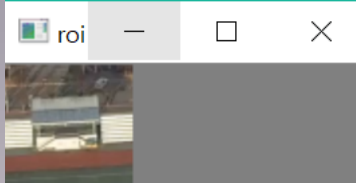
        if cv2.waitKey(1) == 27: #종
            break
```

- `sift.detectAndCompute()`:
roi창과 frame창의 매칭 계산을 실시한다.
- `if len(good)==0:`
재생중인 동영상에 선택된 객체가 감지되지 않으면 다시 지정하도록 한다.
- `cv2.findHomography()`:
두개의 영상에서 객체의 투영변환을 찾고 `cv2.RANSAC`으로 데이터군집의 노이즈를 정리한다.
- `cv2.drawMatches()`:
매칭된 키포인트의 연결선을 출력

Key Point Matching 기반 객체추적 결과화면



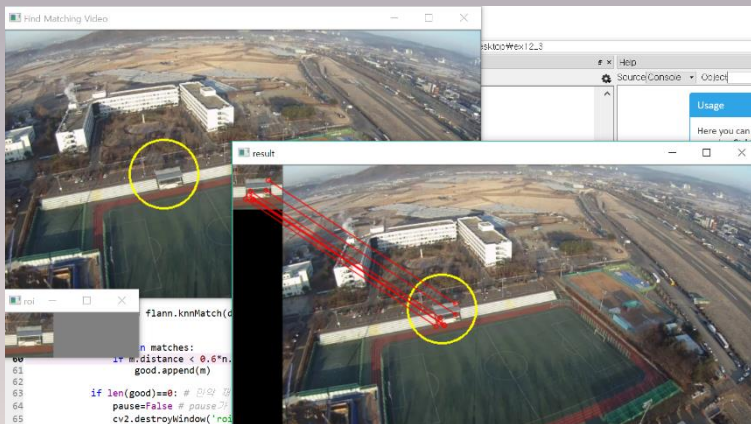
- 마우스로 추적할 객체 선택



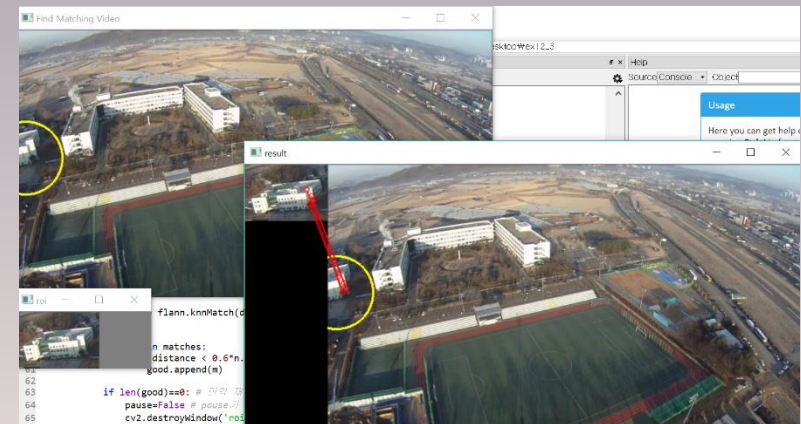
- 선택한 객체



- 실시간 영상 속 객체 및 매칭된 keypoint 표시




- 예시 1



예시 2 -

Key Point Matching 기반 객체추적 결과화면

Find Matching Video



```
():  
    frame, roi, pause  
  
    # 선언, 루프 안에서 선언하면 계산의 낭비이므로 밖에서 미리 정해주고 들어감  
    k = cv2.xfeatures2d.SIFT_create()  
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)  
    search_params = dict(checks=50)  
    matcher = cv2.FlannBasedMatcher(index_params, search_params)  
  
    # 영상 읽기  
    filename = "1st_school_tour.mp4"  
    cap = cv2.VideoCapture(filename)  
  
    while cap.isOpened():  
        ret, frame = cap.read()  
        if not ret:  
            ret = False  
  
    # 영상 크기 조정  
    frame = cap.read()  
    frame = cv2.resize(frame, (640, 360), interpolation=cv2.INTER_AREA)  
    cv2.imshow(windowName, frame)
```

Key Point Matching 기반 객체추적 결과화면

