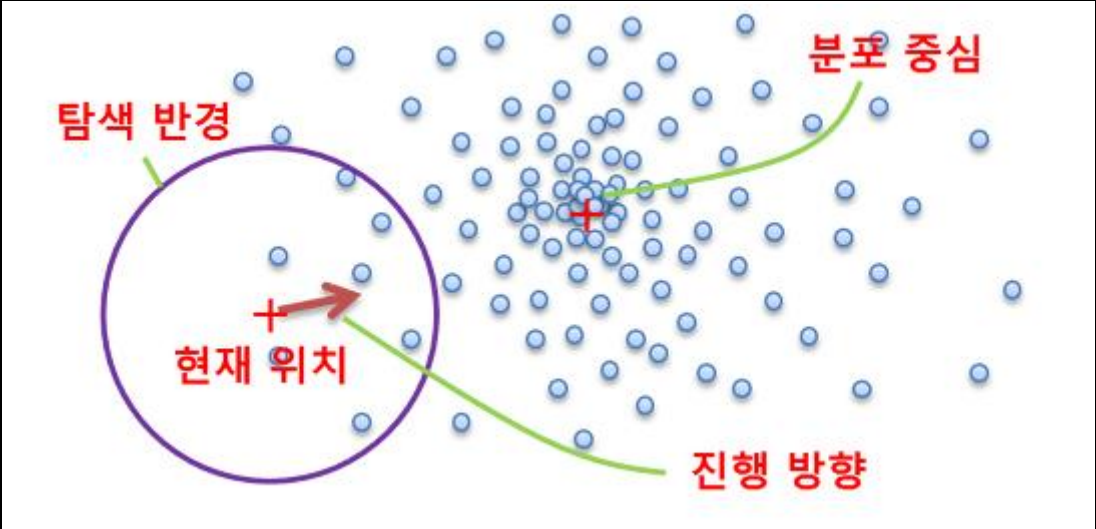
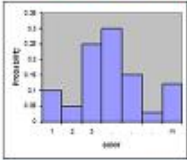


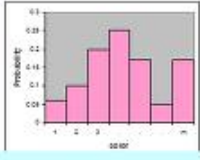
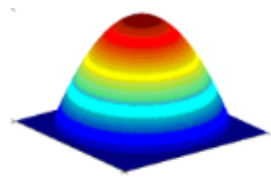


문) Meanshift 추적에 대해 (무엇인지, 장점, 단점 등) 정리하시오.

답)

Question	Answer
무엇인지	<div> <ul style="list-style-type: none"> - 관심영역(ROI: Region Of Interest) 객체의 데이터 집합의 밀도분포(특징점, 코너, 색상)를 기반으로 고속으로 추적하는 방법 - 사용 알고리즘: 평균 이동 알고리즘 <div>: 영상에서 일정 반경 크기의 커널 원도로 픽셀 값의 평균 값을 커널의 중심으로 바뀌서 이동하는 것을 반복해 그 주변의 가장 밀집한 곳(peak)을 찾는 방법</div> - Mode seeking 알고리즘: 특정 데이터들이 중심(mean)으로 이동(shift) - Hill Climb 탐색 방법의 일종 - 기본 아이디어 <div>  </div> - 어떤 데이터 분포의 peak 또는 무게중심을 찾는 한 방법으로서, 현재 자신의 주변에서 가장 데이터가 밀집된 방향으로 이동 그러다 보면 언젠가는 분포 중심을 찾을 수 있을 거라는 방법 </div> <div> <div> <div>Object Model</div> <div>  $\bar{q} = \{q_u\}_{u=1..m} \quad \sum_{u=1}^m q_u = 1$ </div> <div>  </div> <div> <div>입력 영상</div> <div>  <div>  $\bar{p}(y) = \{p_u(y)\}_{u=1..m} \quad \sum_{u=1}^m p_u = 1$ </div> </div> </div> <div> <div>Bhattachayya Coefficient</div> $\sum \sqrt{p_i q_i}$ <div>(히스토그램 유사도 측정)</div> </div> </div> <div> <ul style="list-style-type: none"> - 추적하고자 하는 대상 물체에 대한 색상 히스토그램(histogram)과 현재 입력 영상의 히스토그램을 비교해서 가장 유사한 히스토그램을 갖는 윈도우 영역을 찾는 것 </div> </div>

- 절차

Step	Process
히스토그램 구하기	<ul style="list-style-type: none"> - 색상 모델은 RGB, HSV, YCbCr 등 어느것을 써도 무방 - 또는 그레이(gray)를 사용하거나 HSV의 H(Hue)만 사용해도 됨 - 히스토그램은 윈도우에 들어오는 픽셀들에 대해 각 색상별로 픽셀 개수를 센 다음에 확률적 해석을 위해 전체 픽셀수로 나눠줌
히스토그램 백프로젝션	<ul style="list-style-type: none"> - 현재 입력 영상에 있는 픽셀 색상값이 추적하고자 하는 객체 모델에 얼마나 많이 포함되어 있는 색인지를 수치화하는 과정 - 모델 히스토그램을 H_m, 입력 이미지 I의 픽셀 x에서의 색상값을 $I(x)$라 하면 백프로젝션 값은 $w(x) = H_m(I(x))$ - 보통은 현재 입력 영상에 대한 히스토그램 H를 구한 후 $w(x) = \sqrt{H_m(I(x))/H(I(x))}$ 와 같이 모델 히스토그램 값을 현재 영상 히스토그램 값으로 나누는 것이 일반적
Mean Shift 적용	<ul style="list-style-type: none"> - 히스토그램 백프로젝션을 통해 얻은 w값들을 일종의 확률값처럼 생각하고 mean shift를 적용하는 것 - 이전 영상 프레임(frame)에서의 물체의 위치를 초기 위치로 해서 다음과 같이 mean shift를 적용 - w를 가중치(weight)로 해서 현재 윈도우(window)내에 있는 픽셀 좌표들의 가중평균(무계중심) 위치를 구하는 것 - 이렇게 구한 x_{new}가 새로운 윈도우의 중심이 되도록 윈도우를 이동시킨 다음에 수렴할 때까지 이 과정을 반복 - 커널함수는 배경의 영향을 줄이기 위한 목적으로 사용하는데, 윈도우 중심에서 가장 높은 값을 갖고 중심에서 멀어질수록 값이 작아지는 방사형의 symmetric 함수가 주로 사용됨 <ul style="list-style-type: none"> - 실제 커널 함수로는 Epanechnikov 함수가 주로 사용 $x_{new} = \frac{\sum w(x_i)x_i K(r_i)}{\sum w(x_i)K(r_i)}$ $K_E(x) = \begin{cases} c(1 - \ x\ ^2) & \ x\ \leq 1 \\ 0 & \text{otherwise} \end{cases}$ 
히스토그램 유사도 측정	<ul style="list-style-type: none"> - 두 히스토그램 $H_1 = \{p_1, \dots, p_n\}$, $H_2 = \{q_1, \dots, q_n\}$ 사이의 유사도는 보통 Bhattacharyya 계수(coefficient)를 이용해서 계산 - $Bhattacharyya(H_1, H_2) = \sum \sqrt{p_i q_i}$ - Bhattacharyya 계수는 두 히스토그램이 일치할 때 최대값 1, 상관성이 하나도 없으면 최소값 0
물체의 크기(scale) 결정	<ul style="list-style-type: none"> - mean shift로 위치를 찾은 다음에 윈도우의 크기를 조금씩 변경시켜 보면서 모델 히스토그램과 현재 윈도우 영역에 대한 히스토그램을 비교 - Bhattacharyya 계수가 가장 큰 경우를 찾으면 됨

장점	<ul style="list-style-type: none"> - 고속 추적이 가능한 알고리즘 - 히스토그램의 특성상 위치정보는 고려하지 않기 때문에 물체의 형태가 변해도 추적이 가능 - 단순한 환경(공장자동화 응용, 배경도 단색, 물체도 단색)에서는 최고의 tracker
단점	<ul style="list-style-type: none"> - 효과적이기는 하지만 local minimum에 빠지기 쉬움 - 초기위치(출발위치)에 따라서 최종적으로 수렴하는 위치가 달라질 수 있음 - 탐색 윈도우(탐색 반경)의 크기를 정하는 것이 쉽지 않음 (특히 영상 추적의 경우 대상 물체의 크기, 형태 변화에 따라 탐색 윈도우의 크기나 형태를 적절히 변경해 주어야 하는데 이게 적절히 변경되지 않으면 추적 성능에 많은 영향을 끼치게 됨) - 색상을 기반으로 하므로 추적하려는 객체의 색상이 주변과 비슷하거나 여러가지 색상으로 이루어진 경우 효과를 보기 어려움 - 객체의 크기와 방향과는 상관없이 항상 같은 원도를 반환 - 더 많은 밀도를 가진 지역이 있어도 원도 중심에 머물러 원하는 대로 추적되지 않을 수 있음 - 히스토그램의 특성상 위치정보는 고려하지 않기 때문에 색이 배치된 위치정보를 잃어버리기 때문에 물체의 색 구성이 배경과 유사한 경우에는 추적에 실패하기 쉬움
OpenCV 라이브러리	<pre>retval, window = cv.meanshift(problImage, window, creteria)</pre> <p>problImage: 검색할 히스토그램의 역투영 결과 window: 검색 시작 위치, 검색 결과 위치(x, y, w, h) creteria: 검색 중지 요건, 튜플 객체로 전달</p> <p>type</p> <p>cv2.TERM_CRITERIA_EPS: 정확도가 epsilon 보다 작으면 cv2.TERM_CRITERIA_MAX_ITER: max_iter 횟수를 채우면 cv2.TERM_CRITERIA_COUNT: MAX_ITER와 동일</p> <p>max_iter: 최대 반복 횟수 epsilon: 최소 정확도 retval: 수렴한 반복 횟수</p>
출처	<p>[도서] 파이썬으로 만드는 OpenCV 프로젝트</p> <p>https://techlog.gurucat.net/146</p> <p>https://darkpgmr.tistory.com/</p>