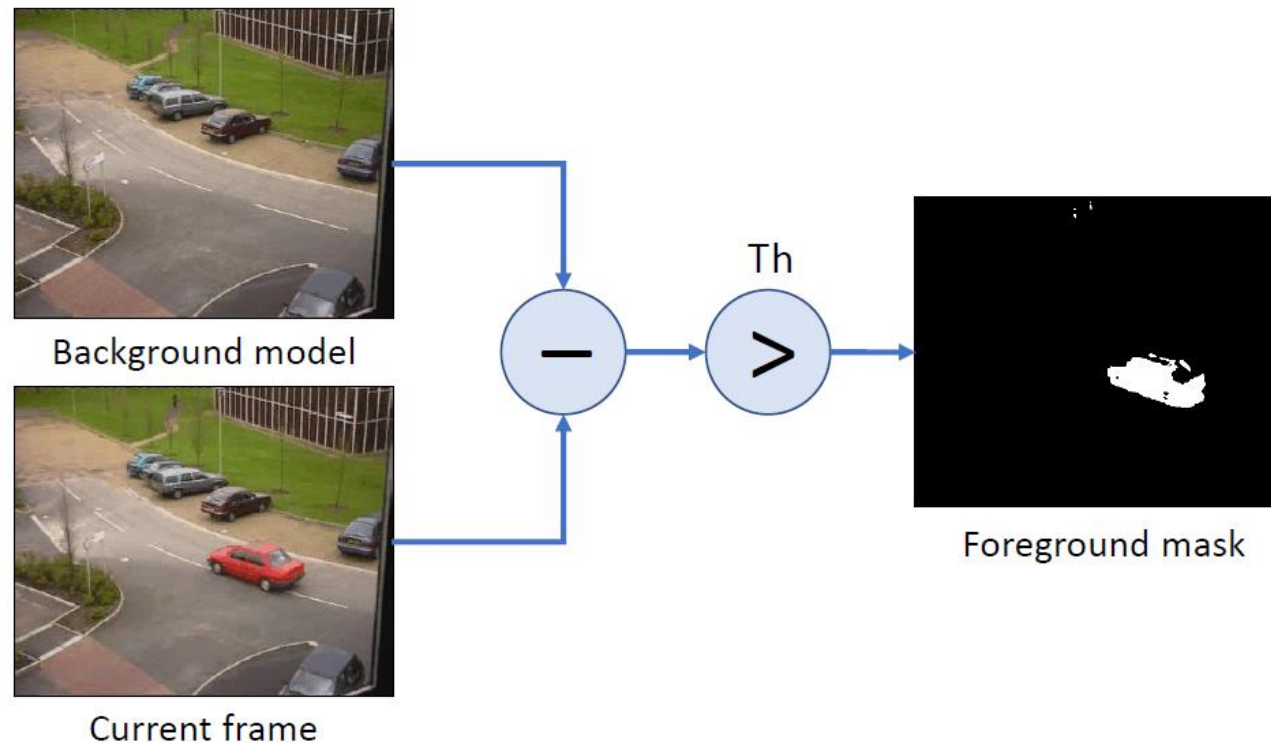


객체 추적과 모션 벡터

멀티미디어

배경 차분 : 정적 배경 차분

- 배경 차분(Background Subtraction: BS)
 - 등록된 배경 모델과 현재 입력 프레임과의 차의 영상을 이용하여 전경 객체를 검출
 - 움직이는 전경 객체 검출을 위한 기본적인 방법



배경 차분 : 정적 배경 차분

- 정적 배경을 이용한 전경 객체 검출 실행 결과
 - 배경 영상과 현재 프레임에 대해 각각 가우시안 필터링을 수행하여 잡음 제거



배경 차분 : 정적 배경 차분

- 정적 배경을 이용한 전경 객체 검출 후 주요 객체 바운딩 박스 표시
 - 레이블링 수행 후 픽셀 개수가 100 이상인 객체에 바운딩 박스 표시



배경 차분 : 이동 평균 배경

- 정적 배경 모델 사용 시 문제점
 - 미리 등록된 기준 영상이 실제 배경과 크게 달라질 경우 오동작
 - Ex_) 그림자 등의 영향으로 인한 조도 변경, 새로운 객체가 화면에 고정될 경우



배경 차분 : 이동 평균 배경

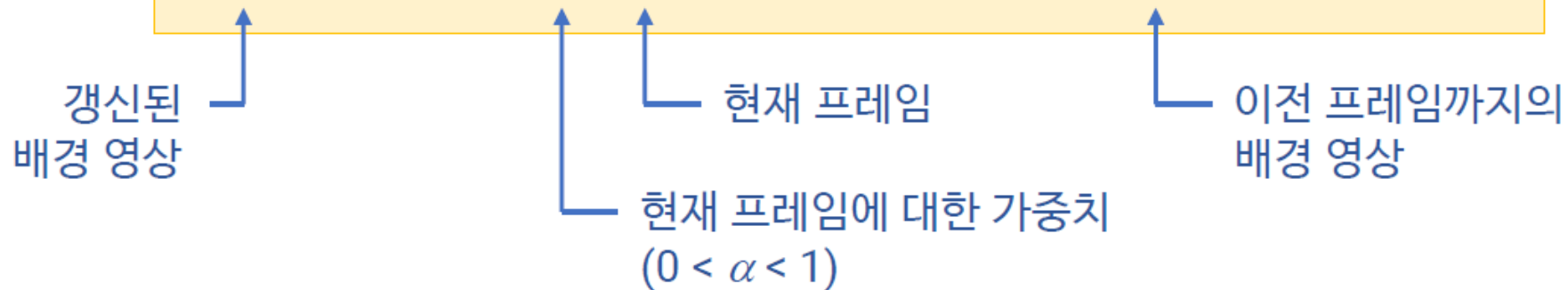
- 평균 연산에 의한 배경 영상 생성
 - 움직이는 객체가 존재하는 수백장의 입력 영상으로부터 평균 영상을 구하면?

$$\left[\begin{array}{c} \text{Image 1} \\ + \\ \text{Image 2} \\ + \dots \\ \hline N \end{array} \right] = \text{Average Image}$$


- 수백장의 이전 프레임을 버퍼에 저장하려면 대용량 메모리가 필요

배경 차분 : 이동 평균 배경

- 이동 평균(Moving average)이란?
 - 수백 장의 영상을 저장하는 대신 매 프레임이 들어올 때마다 평균 영상을 갱신

$$B(x, y, t) = \alpha \cdot I(x, y, t) + (1 - \alpha) \cdot B(x, y, t - 1)$$


갱신된
배경 영상

현재 프레임

현재 프레임에 대한 가중치
($0 < \alpha < 1$)

이전 프레임까지의
배경 영상

- 대용량 버퍼 메모리가 필요하지 않음
- uint8의 정수가 아닌 float32의 실수형으로 계산해줘야함

배경 차분 : 이동 평균 배경

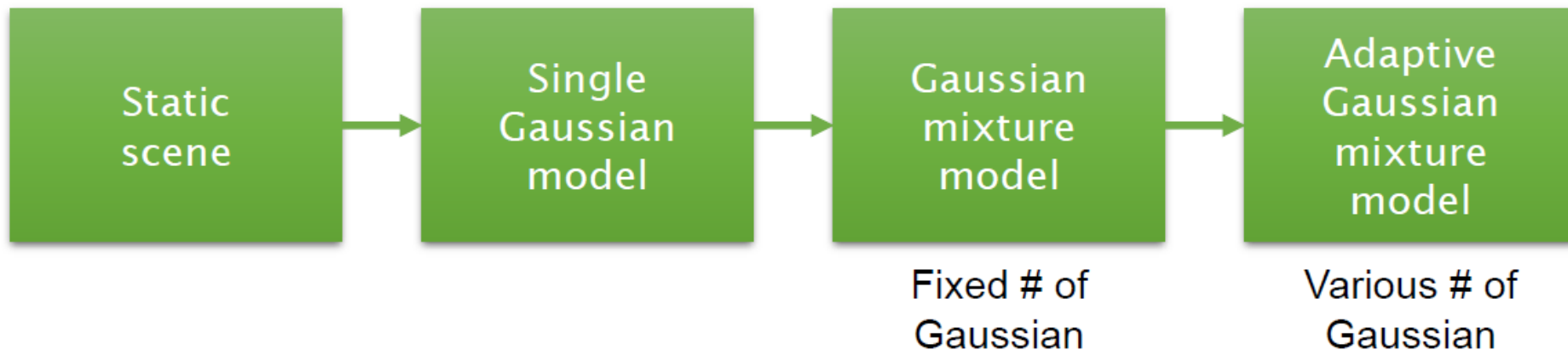
- 이동평균 계산을 위한 가중치 누적 함수

```
cv2.accumulateWeighted(src, dst, alpha, mask=None) -> dst
```

- src : 입력영상, 1또는 3채널. 8비트 또는 32비트 실수형
- dst : 축적 영상. 입력 영상과 동일한 채널 개수. 32비트 또는 64비트 실수형
- alpha : 입력영상에 대한 가중치
- mask : 마스크 영상

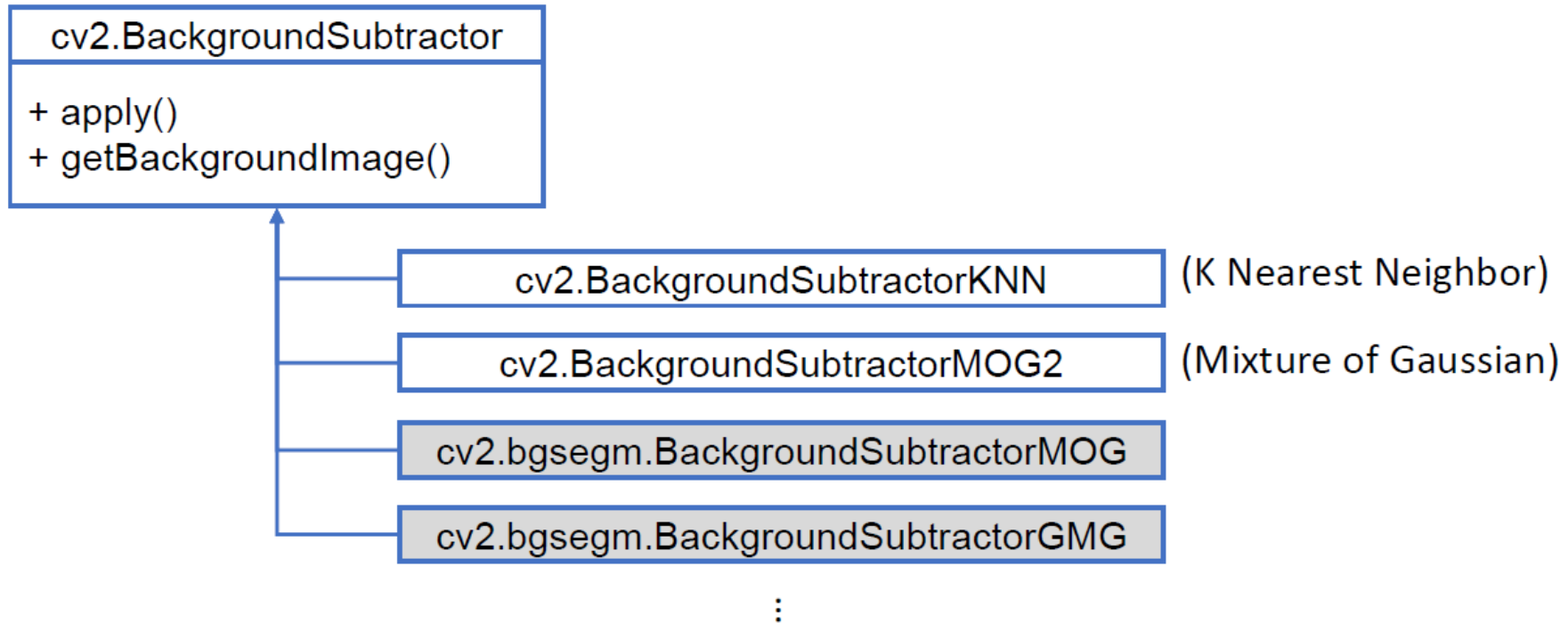
배경 차분 : MOG 배경 모델

- MOG란?
 - Mixture of Gaussian, GMM(Gaussian Mixture Model)
 - 각 픽셀에 대해 MOG 확률 모델을 설정하여 배경과 전경을 구분
- 다양한 배경 모델 구성 방법



배경 차분 : MOG 배경 모델

- OpenCV에서 제공하는 배경 추정 알고리즘

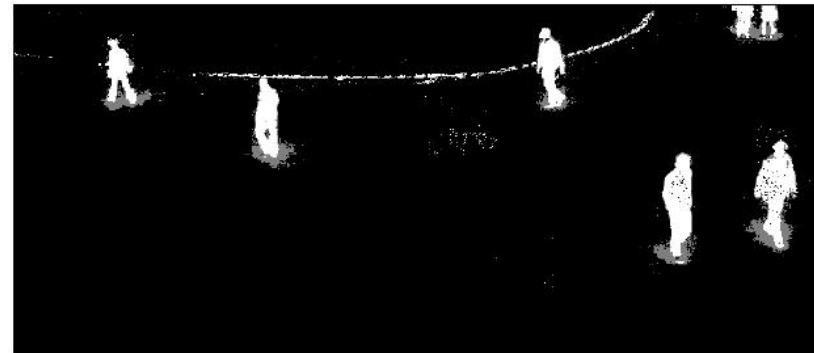


배경 차분 : MOG 배경 모델

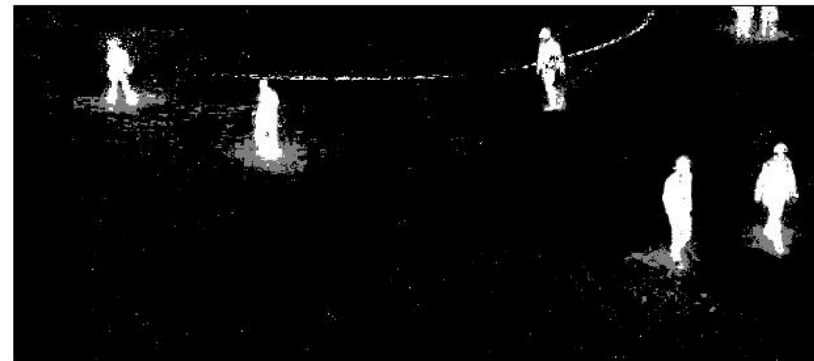
- 움직이는 전경 객체 마스크 영상



입력 프레임 (#200)



BackgroundSubtractorKNN



BackgroundSubtractorMOG2

배경 차분 : MOG 배경 모델

- BackgroundSubtractorMOG2클래스 생성 함수

```
cv2.createBackgroundSubtractorMOG2(, history=None, varThreshold=None,  
                                     detectShadows=None) -> dst
```

- history: 히스토리 길이. 기본값은 500.
- varThreshold: 픽셀과 모델 사이의 마할라노비스 거리(Mahalanobis distance) 제공에 대한 임계값. 해당 픽셀이 배경 모델에 의해 잘 표현되는 지를 판단. 기본값은 16.
- detectShadows: 그림자 검출 여부. 기본값은 True.

배경 차분 : MOG 배경 모델

- 전면 객체 마스크 생성 함수

```
cv2.BackgroundSubtractor.apply(image, fgmask=None, learningRate=None)  
-> fgmask
```

- image: (입력) 다음 비디오 프레임
- fgmask: (출력) 전경 마스크 영상. 8비트 이진 영상.
- learningRate: 배경 모델 학습 속도 지정 (0~1 사이의 실수). 기본값은 -1.

0	배경 모델을 갱신하지 않음
1	매 프레임마다 배경 모델을 새로 만듦
음수	자동으로 결정됨

배경 차분 : MOG 배경 모델

- 배경 영상 반환 함수

```
cv2.BackgroundSubtractor.getBackgroundImage(, backgroundImage=None)  
-> backgroundImage
```

- backgroundImage: (출력) 학습된 배경 영상

추적(Tracking)

- Detection vs Recognition vs Tracking

Detection
(검출)

영상에서 찾고자 하는 대상의 위치와 크기를 알아내는 작업

Recognition
(인식)

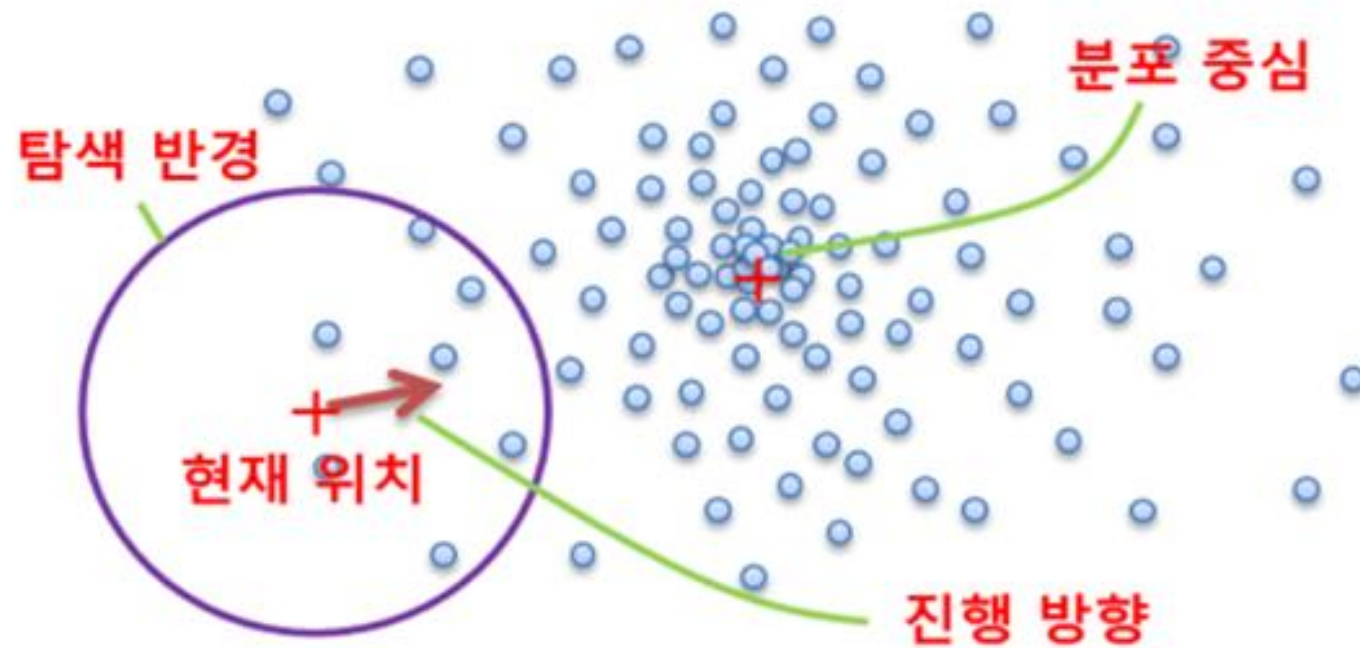
주어진 영상이 무엇인지 판별하는 작업
(classification, identification)

Tracking
(추적)

동영상에서 특정 대상의 위치 변화를 알아내는 작업

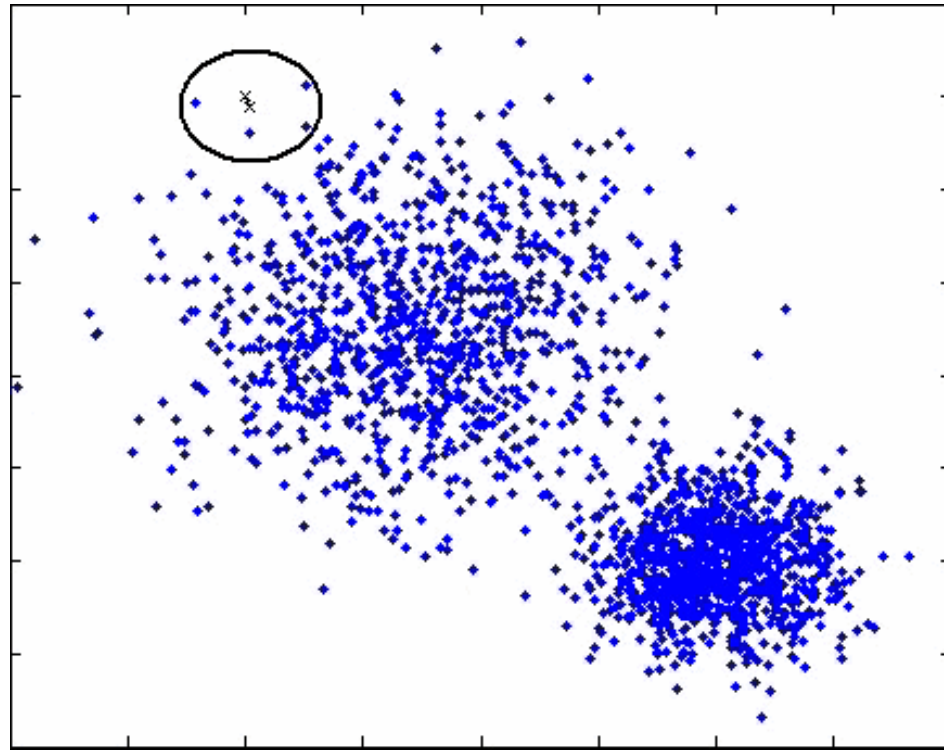
평균 이동 알고리즘

- 평균 이동(Mean shift) : 데이터의 분포를 분석하는 기법 중 하나
 - 국지적 평균을 탐색하면서 이동
 - 데이터가 가장 밀집되어있는 부분을 찾아내기 위한 기법
 - 모드 검출(mode seeking) 알고리즘 이라고도 함



평균 이동 알고리즘

- 평균 이동(Mean shift) : 데이터의 분포를 분석하는 기법 중 하나
 - 국지적 평균을 탐색하면서 이동
 - 데이터가 가장 밀집되어있는 부분을 찾아내기 위한 기법
 - 모드 검출(mode seeking) 알고리즘 이라고도 함



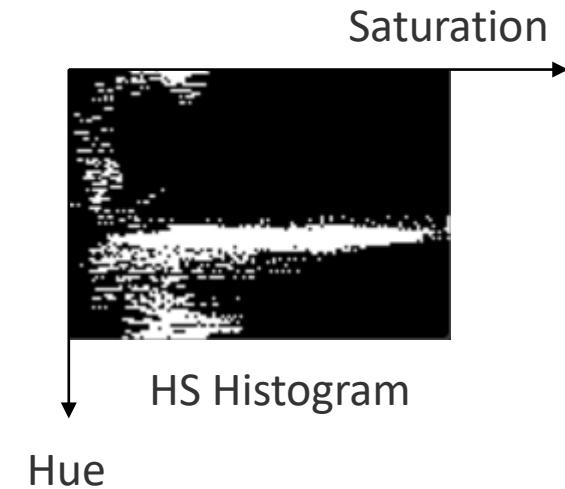
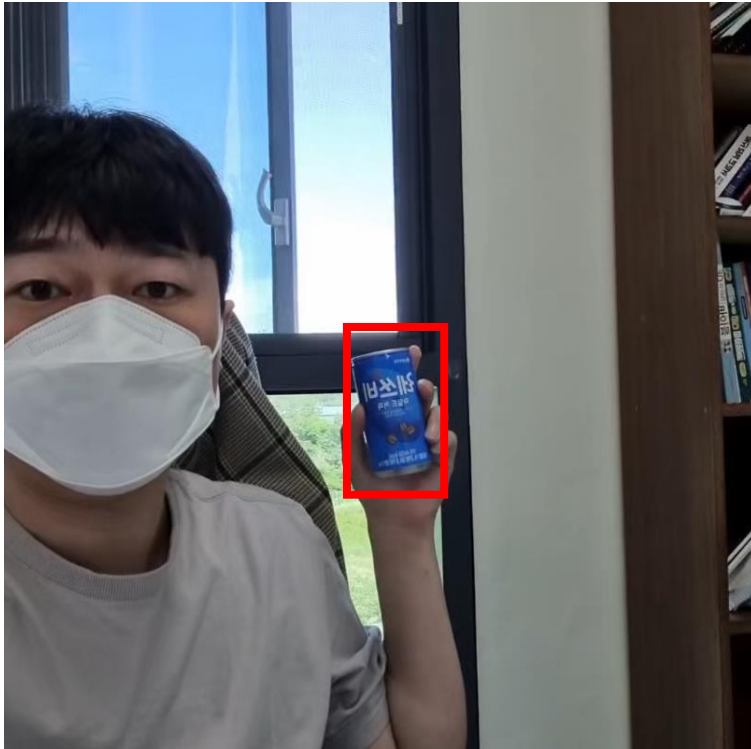
평균 이동 알고리즘



[Camshift — Wikipédia \(wikipedia.org\)](http://en.wikipedia.org/wiki/Camshift)

평균 이동 알고리즘

- 프로그램 동작 방식
 - 1) 추적할 객체 등록
 - 첫 번째 프레임에서 추적할 객체의 위치를 지정
 - HSV 색 공간에서 HS 히스토그램을 구함 H는 Hue색상, S는 Saturation 채도 ~~V는 value 명도~~

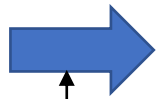


평균 이동 알고리즘

- 프로그램 동작 방식
 - 2) 평균 이동 추적
 - 매 프레임마다 히스토그램 역투영을 수행
 - 여기에 평균 이동 알고리즘을 적용하여 객체를 추적!



Nth Frame



히스토그램 역투
영



객체 추적

평균 이동 알고리즘

- 평균 이동 알고리즘을 이용한 트래킹

```
cv2.meanShift(probImage, window, criteria) -> retval, window
```

- probImage: 관심 객체에 대한 히스토그램 역투영 영상 (확률 영상)
- window: 초기 검색 영역 윈도우 & 결과 영역 반환
- criteria: 알고리즘 종료 기준. (type, maxCount, epsilon) 튜플.
 - (e.g.) `term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)`
→ 최대 10번 반복하며, 정확도가 1이하이면 (즉, 이동 크기가 1픽셀보다 작으면) 종료.
- retval: 알고리즘 내부 반복 횟수.

캠시프트 알고리즘

- 캠시프트(CamShift)란?
 - Continuously Adaptive Mean Shift(연속적인 조정 평균 이동)
 - 추적하는 객체의 크기가 변하더라도 검색 윈도우의 크기가 고정되어 있는 평균 이동 알고리즘의 단점을 보완
- 캠시프트 동작 방법
 - 평균 이동 알고리즘으로 이동 위치 계산
 - 윈도우 크기를 조정
 - 특징 공간을 가장 잘 표현하는 타원 검출
 - 새로운 크기의 윈도우를 이용하여 다시 평균 이동 수행

캠시프트 알고리즘



Mean shift window
initialization

[Camshift — Wikipédia \(wikipedia.org\)](http://en.wikipedia.org/wiki/Camshift)

캠시프트 알고리즘

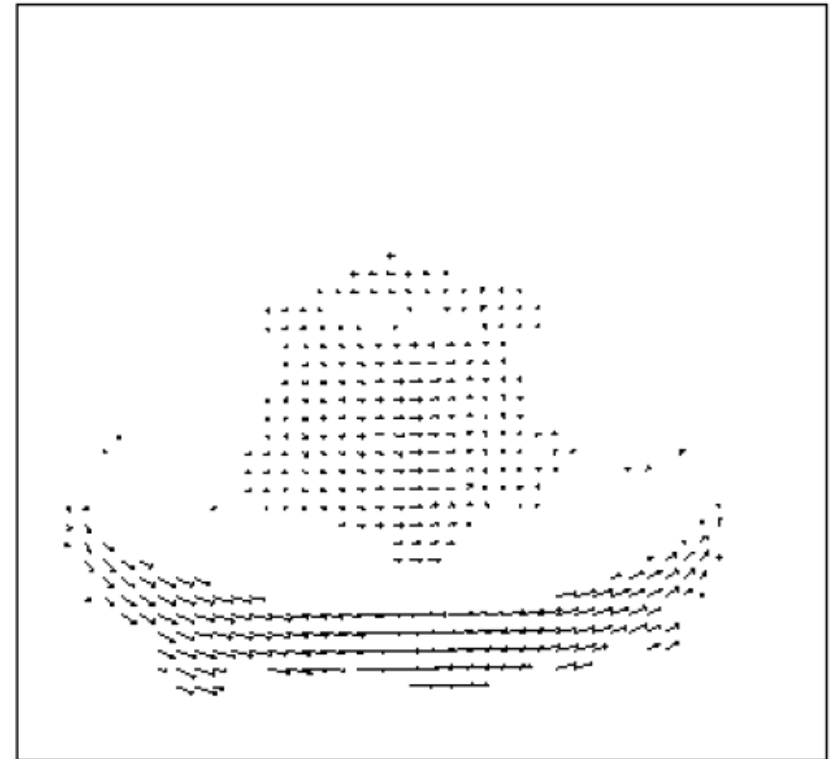
- 캠시프트 추적 함수

```
cv2.CamShift(probImage, window, criteria) -> retval, window
```

- probImage: 관심 객체에 대한 히스토그램 역투영 영상 (확률 영상)
- window: 초기 검색 영역 윈도우 & 결과 영역 반환
- criteria: 알고리즘 종료 기준. (type, maxCount, epsilon) 튜플.
 - (e.g.) `term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)`
→ 최대 10번 반복하며, 정확도가 1이하이면 (즉, 이동 크기가 1픽셀보다 작으면) 종료.
- retval: 추적하는 객체의 모양을 나타내는 회전된 사각형 정보를 반환.
((cx, cy), (width, height), angle) 튜플.

옵티컬플로우 (동영상 움직임 계산)

- 옵티컬플로우(Optical flow)란?
 - 연속하는 두 프레임(영상)에서 카메라 또는 객체의 움직임에 의해 나타나는 객체의 이동 정보 패턴



옵티컬플로우

- OpenCV 옵티컬플로우 계산 함수
 - 루카스-카나데 알고리즘
 - `cv2.calcOpticcalFlowPyrLK(...)`
 - 주로 Sparse Points에 대한 이동 벡터 계산
 - 특정 픽셀에서 옵티컬플로우 벡터 계산
 - 파네백 알고리즘
 - `cv2.calcOpticcalFlowFarneback(...)`
 - 주로 Dense Points에 대한 이동 벡터 계산
 - 모든 픽셀에서 옵티컬플로우 벡터 계산

옵티컬플로우

- 루카스-카나데 옵티컬플로우 계산 함수

```
cv2.calcOpticalFlowPyrLK(prevImg, nextImg, prevPts, nextPts, status=None,  
    err=None, winSize=None, maxLevel=None, criteria=None, flags=None,  
    minEigThreshold=None) -> nextPts, status, err
```

- prevImg, nextImg: 이전 프레임과 현재 프레임. 8비트 입력 영상.
- prevPts: 이전 프레임에서 추적할 점들. `numpy.ndarray`. shape=(N, 1, 2), dtype=np.float32.
- nextPts: (출력) prevPts 점들이 이동한 (현재 프레임) 좌표.
- status: (출력) 점들의 매칭 상태. `numpy.ndarray`. shape=(N, 1), dtype=np.uint8. i번째 원소가 1이면 prevPts의 i번째 점이 nextPts의 i번째 점으로 이동.
- err: 결과 오차 정보. `numpy.ndarray`. shape=(N, 1), dtype=np.float32.
- winSize: 각 피라미드 레벨에서 검색할 윈도우 크기. 기본값은 (21, 21).
- maxLevel: 최대 피라미드 레벨. 0이면 피라미드 사용 안 함. 기본값은 3.
- criteria: (반복 알고리즘의) 종료 기준
- ...

옵티컬플로우

- 밀집 옵티컬플로우 계산 함수

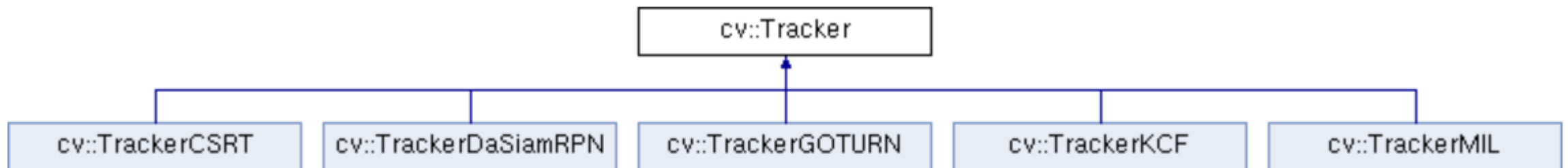
```
cv2.calcOpticalFlowFarneback(prev, next, flow, pyr_scale, levels, winsize,  
                             iterations, poly_n, poly_sigma, flags) -> flow
```

- prev, nex: 이전 영상과 현재 영상. 그레이스케일 영상.
- flow: (출력) 계산된 옵티컬플로우. `np.ndarray`. `shape=(h, w, 2)`, `dtype=np.float32`.
- pyr_scale: 피라미드 영상을 만들 때 축소 비율. (e.g.) 0.5
- levels: 피라미드 영상 개수. (e.g.) 3
- winsize: 평균 윈도우 크기. (e.g.) 13
- iterations: 각 피라미드 레벨에서 알고리즘 반복 횟수. (e.g.) 10
- poly_n: 다항식 확장을 위한 이웃 픽셀 크기. 보통 5 또는 7.
- poly_sigma: 가우시안 표준편차. 보통 `poly_n = 5`이면 1.1, `poly_n = 7`이면 1.5.
- flags: 0, `cv2.OPTFLOW_USE_INITIAL_FLOW`, `cv2.OPTFLOW_FARNEBACK_GAUSSIAN`.

OpenCV트래커

- OpenCV 트래커
 - OpenCV 3.0 버전 부터 트래커 클래스 제공
 - 트래커 클래스는 OpenCV extra modules에서 지원하므로 opencv-contrib-python 패키지를 설치해야함

```
> pip uninstall opencv-python  
> pip install opencv-contrib-python==4.1.0.25
```



OpenCV트래커

- Tracker 클래스 사용 방법

```
cv2.TrackerXXX_create() -> <TrackerXXX object>
```



```
cv2.Tracker.init(image, boundingBox) -> retval
```

boundingBox: 초기 사각형 ROI
(실수형 (x, y, w, h) 튜플)



```
cv2.Tracker.update(image) -> retval, boundingBox
```

retval: 추적에 성공하면 True,
실패하면 False.

