가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA

비디오처리

미디어기술콘텐츠학과
강호철

평균 배경 차영상

- 차영상을 이용한 분할
  - 배경으로 부터 전경(객체) 분할
  - 배경 차영상 사용
    - 입력 프레임 영상 사이의 화소 차이 계산
    - 임계값 이상의 화소 위치 변화가 있는 경우

- 구현
  - 누적 영상을 구하는 OpenCV 함수 사용
  - 차영상 절대값이 임계값보다 큰 화소는 마스크 0으로 설정
  - 차영상 절대값이 임계값보다 작은 화소는 마스크 1로 설정

평균 배경 차영상

- OpenCV 함수

`cv2.accumulate(src, dst[, mask])` : 화소 누적 합

$$dst(x,y) \leftarrow dst(x,y) + src(x,y) \quad \text{if } mask(x,y) \neq 0$$

parameter

- src: input image
- dst: output image
- mask: Optional operation mask

`cv2.accumulateSquare(src, dst[, mask])` : 화소 제곱 합

$$dst(x,y) \leftarrow dst(x,y) + src(x,y)^2 \text{ if } mask(x,y) \neq 0$$

`cv2.accumulateProduct(src1, src2, dst[, mask])` : 화소2개의 product 후 누적 합

$$dst(x,y) \leftarrow dst(x,y) + src1(x,y) \cdot src2(x,y) \text{ if } mask(x,y) \neq 0$$

parameter

- src1: input image1
- src2: input image2
- dst: output image
- mask: Optional operation mask

https://wjddyd66.github.io/opencv/OpenCV(9)/

평균 배경 차영상

- OpenCV 함수

`cv2.accumulateWeighted(src, dst, alpha[, mask])` : 가중치를 적용한 화소 누적 합

$$dst(x,y) \leftarrow (1-\alpha) \cdot dst(x,y) + \alpha \cdot src(x,y) \text{ if } mask(x,y) \neq 0$$

parameter

- src: input image
- dst: output image
- alpha: weight
- mask: Optional operation mask

`cv2.convertScaleAbs(src[, dst[, alpha[, beta]]])` : 절대값 계산후 8bit로서 표현

$$dst(I) = saturate - cast(unchar(8bit))(|src(I) * alpha + beta|)$$

parameter

- src: input array
- dst: output array
- alpha: optional scale facotr
- mask: optional delta added to the scaled values

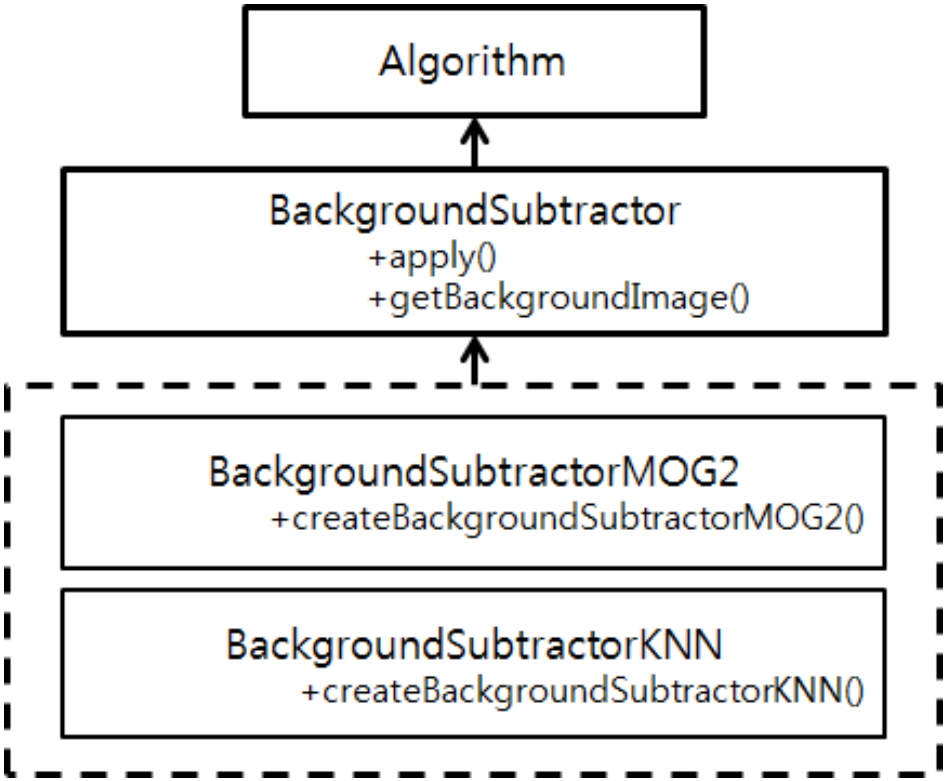
https://wjddyd66.github.io/opencv/OpenCV(9)/

평균 배경 차영상

- 실습
  - 평균에 의한 배경영상
  - 배경 차영상 이동물체 검출
  - 이동평균 배경 차영상

배경과 전경 분할

- BackgroundSubtractor
  - GMM 혹은 K-NN 등 통계적 방법으로 화소 모델링
    - 배경과 전경 화소 분할



배경과 전경 분할

- OpenCV 함수

`cv2.BackgroundSubtractor.apply(image,fgmask,learningRate)` : image를 모델에 적용하여 8비트 전경 마스크 fgmask를 계산한다.

parameter

- image:Next video frame
- fgmask: foreground mask as an 8-bit binary image(Current Frame - Background model)
- leaningRate: 0 ~ 1 값으로 Background Model갱신을 어느정도 비율로 할지 정한다.
- 0: BackGround Model을 갱신하지 않는다.
- 1: BackGround Model을 마지막 프레임으로 갱신한다.

`cv2.BackgroundSubtractor.getBackgroundImage(backgroundImage)` : Background Image 반환

https://wjddyd66.github.io/opencv/OpenCV(9)/

배경과 전경 분할

- OpenCV 함수

`cv2.creteBackgroundSubtractorMoG(history, varThreshold, detectShadows)` : Gaussian Mixture-based Background/Foreground Segmentation Algorithm을 사용하여 Background와 Object를 분리한다.

parameter

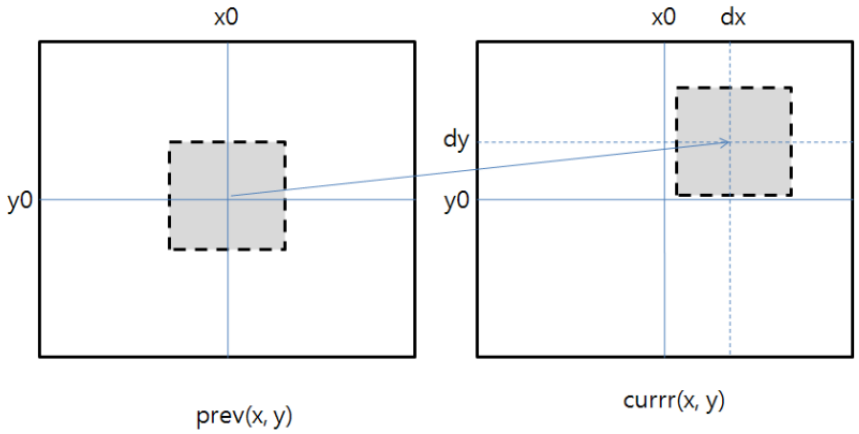
- history: Background Model 검출에 영향을 주는 최근 프레임의 길이
- varThreshold: 배경을 판단하기 위한 Mahalanobis 거리 제곱의 임계치(작은 값을 사용하면 많은 화소를 전경(foreground)로 검출한다.)
- detectShadows: 그림자 검출 여부
- True: 검출 O
- False: 검출 X

`cv2.creteBackgroundSubtractorKNN(history, varThreshold, detectShadows)` : Creates KNN Background Subtractor 생성

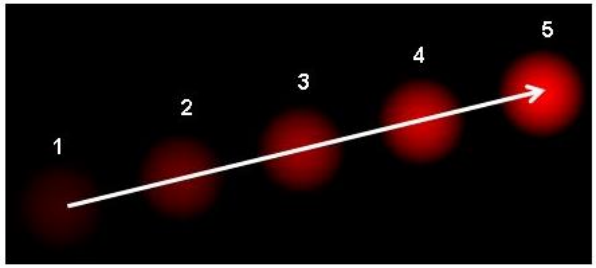
https://wjddyd66.github.io/opencv/OpenCV(9)/

Optical Flow

- 광류(optical flow)
  - 영상 화소 밝기의 움직임을 벡터로 계산하는 방법
    - 물체의 이동 분석, 추적
    - 비디오압축, 화질개선



Optical flow : velx(x0, y0) =dx
vely(x0, y0) =dy



http://www.gisdeveloper.co.kr/?p=6855

Optical Flow

- OpenCV 함수
  - Lucas-Kanade 방법
    - Pros: 다양한 스케일 영상 탐색 → 큰 움직임 검출 가능
    - Cons: Keypoints 몇 개만 계산하므로 정확도가 상대적으로 떨어짐

`cv2.calcOpticalFlowPyrLK(prevImg, nextImg, prevPts[, nextPts[, status[, err[, winSize[, maxLevel[, criteria[, flags[, minEigThreshold]]]]]]]])` : Lucas-Kanade method with pyramids를 통하여 optical flow 를 계산한다.

parameter

- prevImg: first 8-bit input image or pyramid constructed by buildOpticalFlowPyramid().
- nextImg: second input image or pyramid of the same size and the same type as prevImg.
- prevPts: vector of 2D points for which the flow needs to be found; point -coordinates must be single-precision floating-point numbers.
- nextPts: output vector of 2D points

https://wjddyd66.github.io/opencv/OpenCV(9)/

Optical Flow

- OpenCV 함수
  - Farneback 방법
    - Pros: 모든 픽셀에 대한 Keypoints를 통해 계산하므로 정확도가 높음

      Cons: 계산 시간이 오래 걸림

`cv2.calcOpticalFlowFarneback(prev, next, pyr_scale, levels, winsize, iterations, poly_n, poly_sigma, flags[, flow])` : Gunnar Franeback's Algorithm을 통하여 optical flow 를 계산한다.

parameter

- prev: first 8-bit single-channel input image.
- next: second input image of the same size and the same type as prev.
- flow: computed flow image that has the same size as prev and type CV_32FC2.
- pyr_scale: parameter, specifying the image scale (<1) to build pyramids for each image; pyr_scale=0.5 means a classical pyramid, where each next layer is twice smaller than the previous one.

https://wjddyd66.github.io/opencv/OpenCV(9)/

MeanShift/CamShift

- Tracking
  - 특정 물체를 계속 추적하는 기술
  - MeanShift, CamShift 주로 사용
    - 히스토그램 역투영 이용
    - 컬러 영상의 경우 Hue 채널을 주로 사용

- OpenCV 함수

`cv2.meanShift(probImage, window, criteria)` : Back projection image를 통하여 Object 검출

parameter

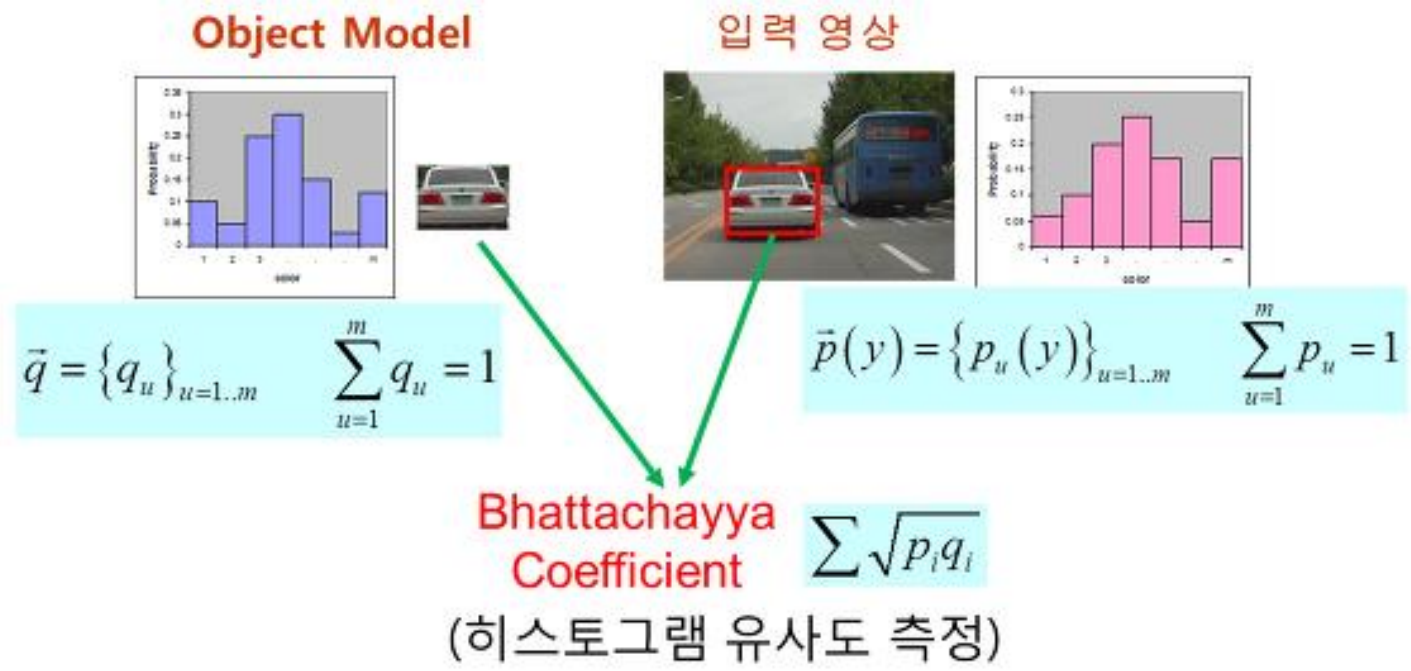
- probImage: Back projection of the object histogram
- window: Initial search window
- criteria: 탐색 종료 조건

`cv2.CamShift(probImage, window, criteria)` : Back projection image를 통하여 Object 검출

https://wjddyd66.github.io/opencv/OpenCV(9)/

MeanShift/CamShift

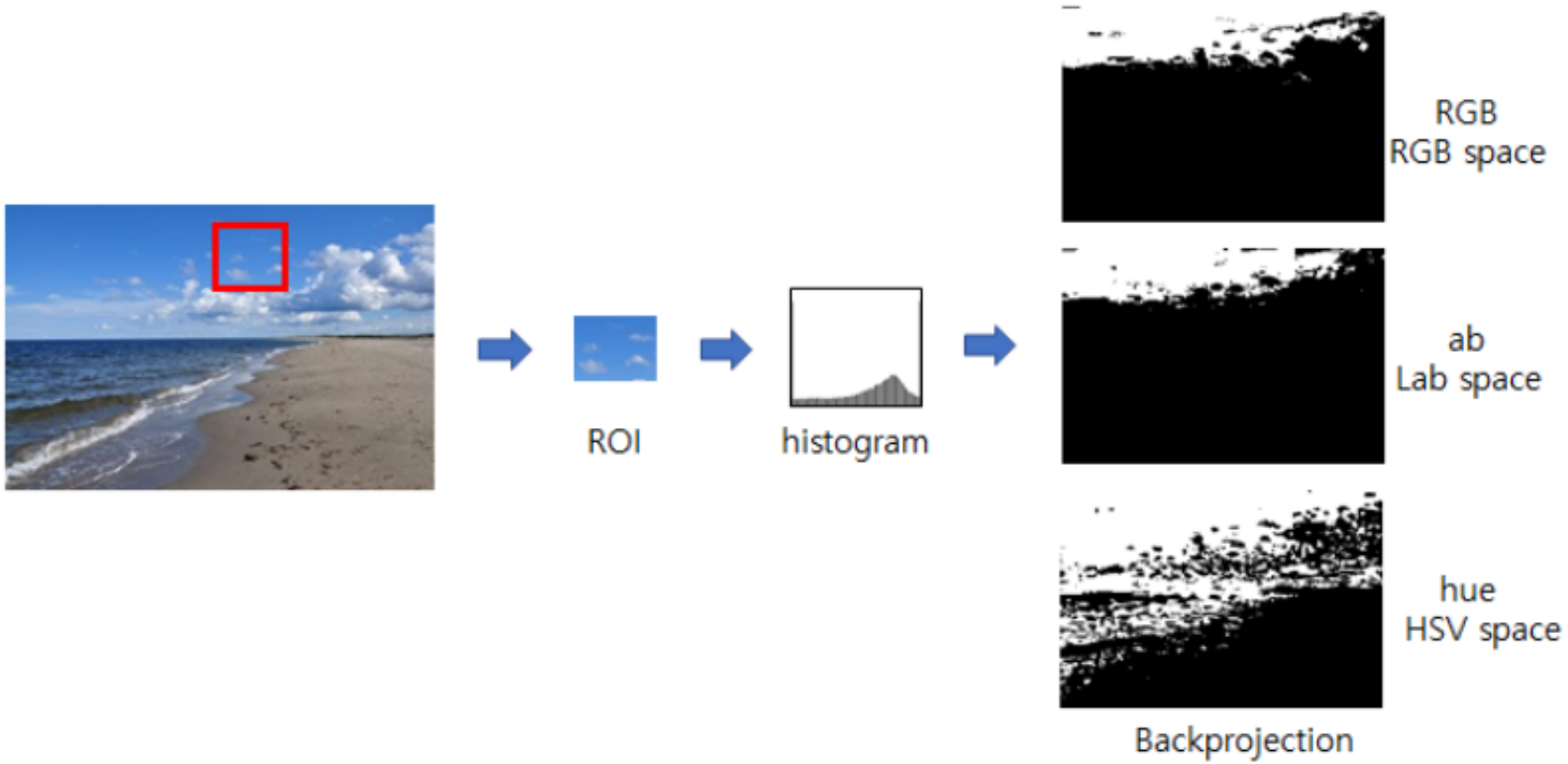
- Histogram Backprojection
  - 해당 픽셀의 값을 확률로 표현



https://wjddyd66.github.io/opencv/OpenCV(9)/

MeanShift/CamShift

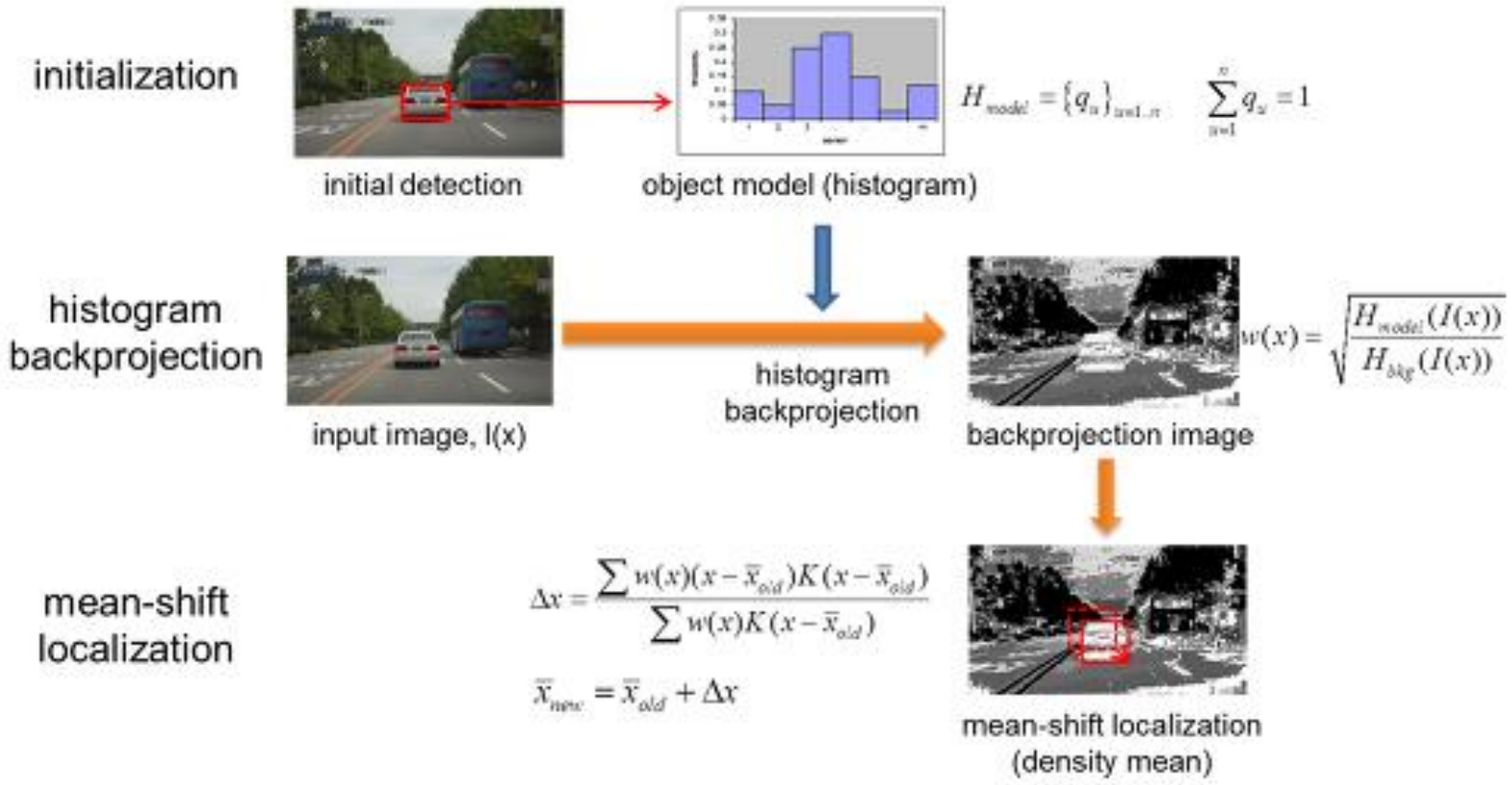
- Histogram Backprojection
  - Image Segmentation



https://preventionyun.tistory.com/37

MeanShift/CamShift

- Histogram Backprojection
  - MeanShift에 적용



https://wjddyd66.github.io/opencv/OpenCV(9)/

Tracker

- OpenCV 추적기
  - ***#pip install opencv-contrib-python***
  - AdaBoost
  - KCF (Kernelized Correlation Filters)
  - Multiple Instance Learning
  - MOSSE (Minimum Output Sum of Squared Error)
  - Median Flow
  - TLD (Tracking-Learning-Detection)
  - GOTURN (Generic Object Tracking Using Regression Networks)
    - goturn.caffemodel
    - goturn.porotxt

참고자료

- Python으로 배우는 OpenCV 프로그래밍
  - 김동근 지음
  - 가메출판사, 2018

- OpenCV4로 배우는 컴퓨터 비전과 머신러닝
  - 황선규 지음
  - 길벗출판사, 2019