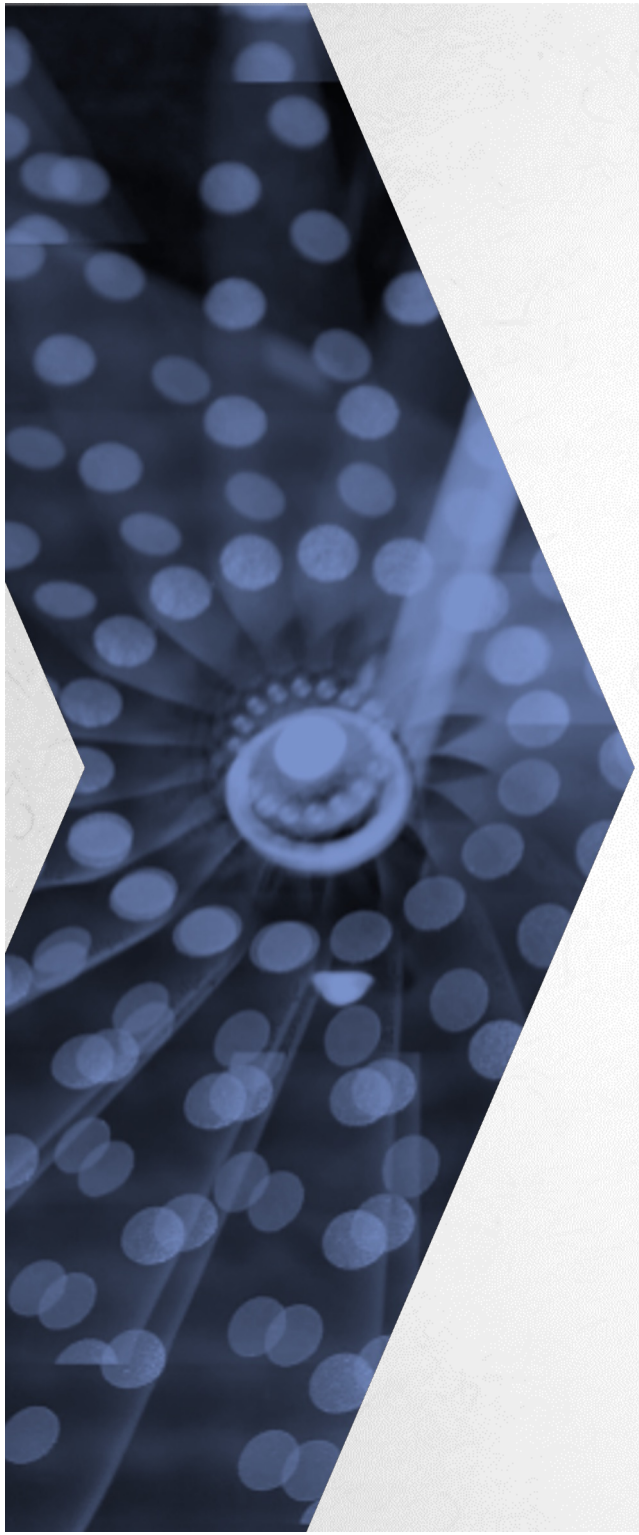


빅데이터 분석이지만 "OpenCV" 강의

강사: 김정호

kimsmap@outlook.com

7.3 영상 분할 및 합성

7.3.1 전경-배경 분할

- 전경(앞쪽에 보이는 경치)과 배경(뒤쪽의 경치)는 연속된 이미지, 즉 비디오에서 움직임이 없는 고정된 영역인 배경과 움직임이 존재하는 영역을 구분하기 위한 기술
- 앞서 웹캠 과제를 통해 수행해 본 바 있음
- 전경 분할을 위해 두 가지 접근 방법을 고려해 볼 수 있음

  1) 공간적 관점 : 최소한의 픽셀 크기를 지정하는 것

  ex) 사람은 가로•세로가 각각 30, 60 픽셀 이상을 차지해야 한다.

  2) 시간적 관점 : 직전 이미지(프레임)과 비교해 이동 거리, 진행 방향의 변경 등 물리적 한계를 지정하는 것

  ex) 갑작스런 이동이나 진행 방향의 급격한 변화 등이 없어야 한다.

7.3 영상 분할 및 합성

7.3.1 전경-배경 분할

- 일반적으로 발생할 수 있는 문제점은 다음을 예로 들 수 있음

항 목	문제점
조명 변화	천천히 혹은 급격한 조명 변화에 의해 잘못된 전경 분할
그림자	그림자를 움직이는 객체로 인식
비, 눈	비나 눈에 의해 화면이 일시적으로 크게 바뀌거나 지속적인 잡음에 의해 잘못된 전경 분할
바람	나뭇가지나 깃발 또는 지형물이 바람에 의해 움직이는 객체로 인식
배경의 갱신	배경이 움직여서 생기는 잘못된 전경 분할
카메라 움직임	카메라가 움직여서 잘못된 전경 분할

7.3 영상 분할 및 합성

7.3.1 전경-배경 분할

- 이전에 수행해봤던 예제를 다시 해봅시다.
- 그리고 앞선 문제점을 확인해 봅시다.

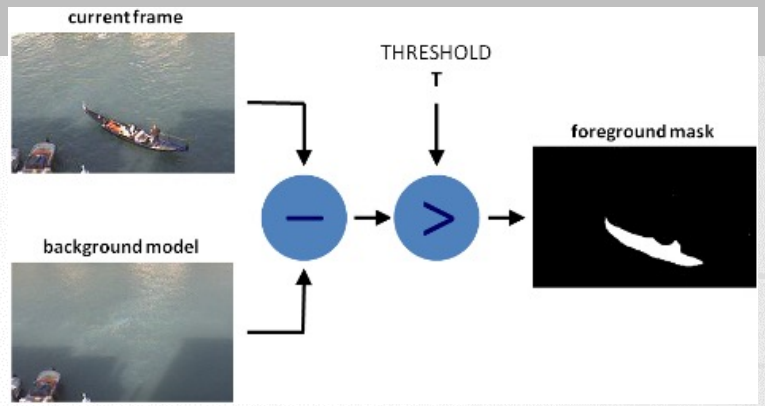
항 목
조명 변화
그림자
비, 눈
바람
배경의 갱신
카메라 움직임

```python
import cv2
import numpy as np

# camera setting
cam = cv2.VideoCapture(0)
cam.set(cv2.CAP_PROP_FRAME_WIDTH,  960)
cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 540)

# q키를 눌러서 순서대로 배경 사진 촬영하기
winName1 = 'background capture'
while cam.isOpened():
    status, frame = cam.read()
    if status:
        cv2.imshow(winName1, frame)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        img1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        break

winName2 = 'foreground capture'
while cam.isOpened():
    status, frame = cam.read()
    if status:
        img2 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        img3 = cv2.absdiff(img1, img2)
        cv2.imshow(winName2, img3)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

```

7.3 영상 분할 및 합성

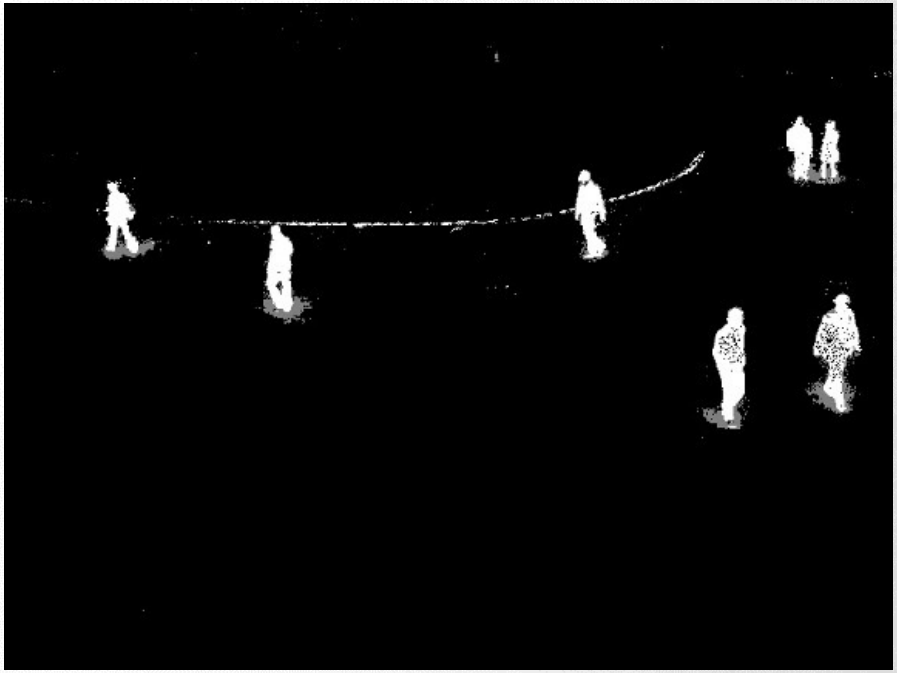
7.3.1 전경-배경 분할

- 배경을 제거하는 방법은 개념 상으로는 쉬움



차영상의 개념

입력 영상



KNN을 이용한 추출 결과

7.3 영상 분할 및 합성

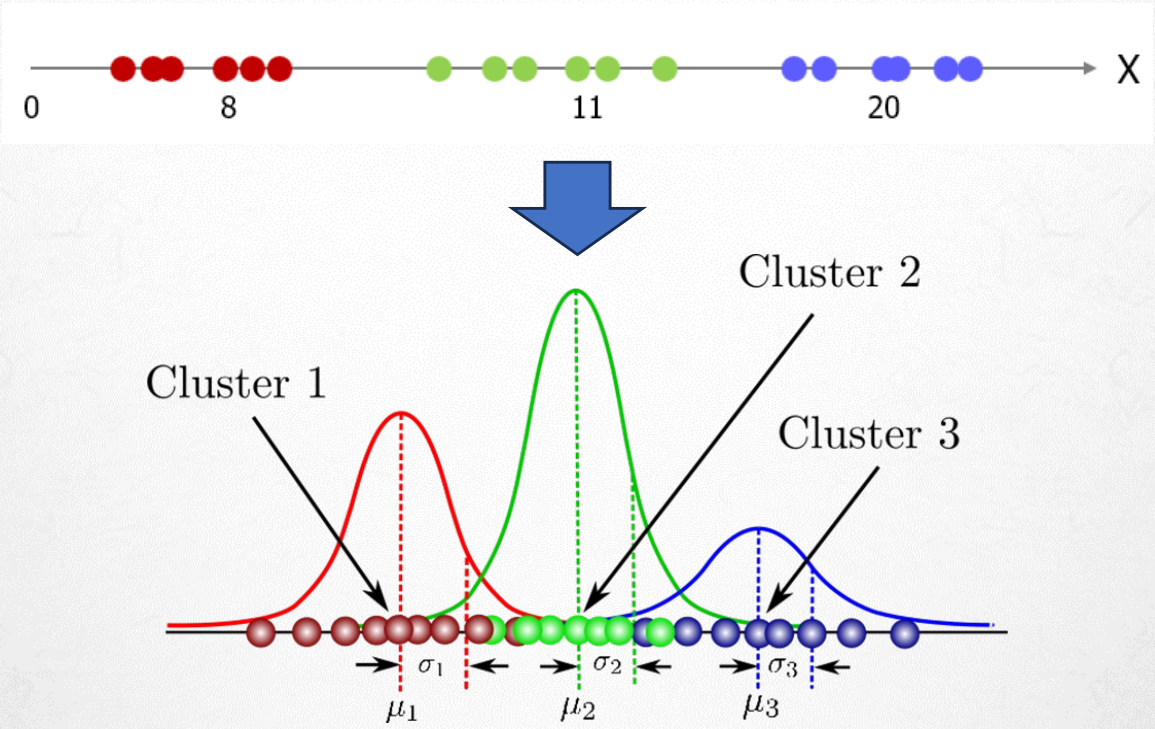
7.3.1 전경-배경 분할

- OpenCV에서는 전경과 배경을 분할하는 데 MOG와 KNN 기법을 제공함

  MOG : Mixture of Gaussian

  KNN : K-Nearest Neighbor

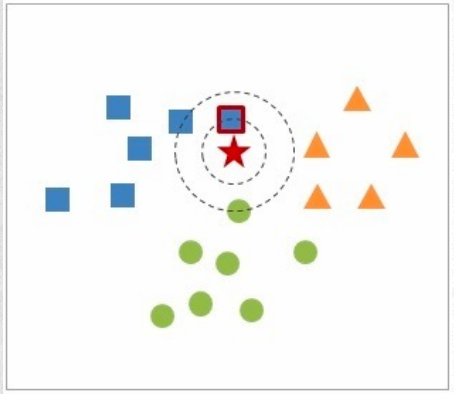
- GMM은 다음 그림과 같이 분포한 점이 있을 때, 각각은 Gaussian 분포를 가진다는 개념에서 시작



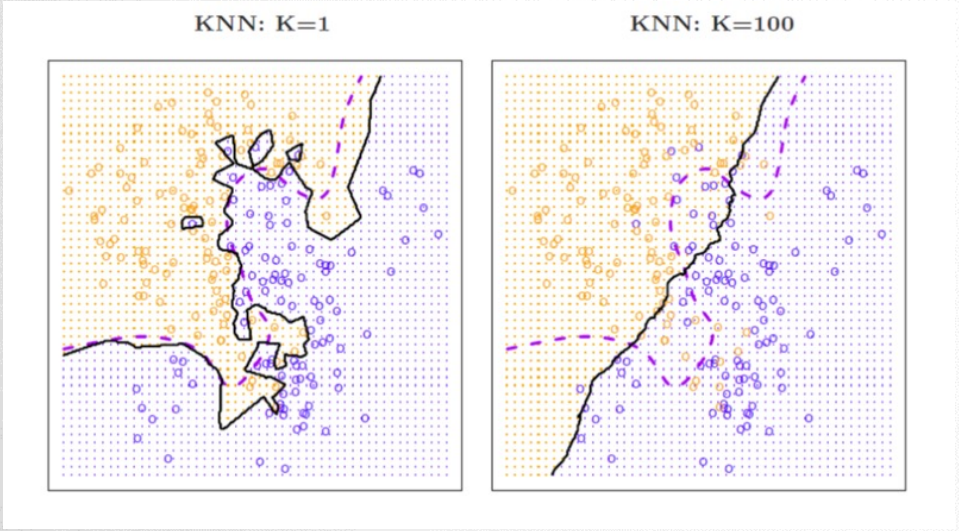
7.3 영상 분할 및 합성

7.3.1 전경-배경 분할

- KNN은 '새로 추가된 별모양이 사각형에 가까우니 사각형일 것이다'라고 분류하는 방법을 의미함



- 여기서 K는 몇 번째 가까운 점까지 살펴볼 것인가를 의미하며 K에 따라 다음과 같은 차이를 보인다.



7.3 영상 분할 및 합성

7.3.1 전경-배경 분할

OpenCV: Motion Analysis

- GMM의 함수 원형은 다음과 같음

```
cv.createBackgroundSubtractorMOG2( [, history[, varThreshold[, detectShadows]]] ) -> retval
```

history	시간 상 몇 프레임까지 고려할 것인지를 나타내는 척도 (기본값=500)
varThreshold	모델과 픽셀의 관계를 결정하기 위한 상대거리 계산 시 사용하는 경계값(threshold) (기본값=400)
detectShadows	이 기능을 켜면 그림자도 인식해서 마킹해줌 (기본값=True) 다만 연산량이 늘어나므로 필요 없다면 끄는 것이 좋음

- KNN의 함수 원형은 다음과 같음

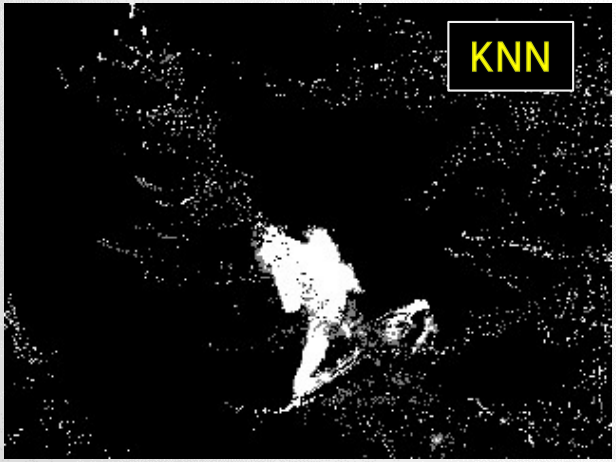
```
cv.createBackgroundSubtractorKNN( [, history[, dist2Threshold[, detectShadows]]] ) -> retval
```

history	시간 상 몇 프레임까지 고려할 것인지를 나타내는 척도 (기본값=500)
dist2Threshold	모델과 픽셀의 관계를 결정하기 위한 상대거리 계산 시 사용하는 경계값(threshold) (기본값=16)
detectShadows	이 기능을 켜면 그림자도 인식해서 마킹해줌 (기본값=True) 다만 연산량이 늘어나므로 필요 없다면 끄는 것이 좋음

7.3 영상 분할 및 합성

7.3.1 전경-배경 분할

- OpenCV에서 제공하는 MOG
  KNN을 이용해 전경-배경 분할 수행



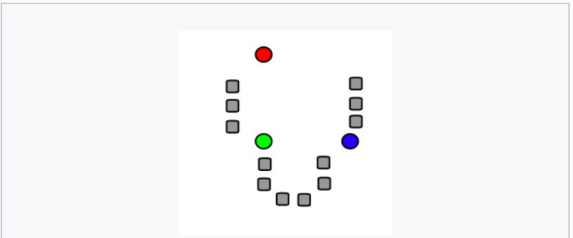

```python
import cv2
import numpy as np

cap = cv2.VideoCapture("./../images/video2.mp4")

# 전경-배경 분할 수행자 선언
bgMethod1 = cv2.createBackgroundSubtractorMOG2()
bgMethod2 = cv2.createBackgroundSubtractorKNN()
bgMethod1_blur = cv2.createBackgroundSubtractorMOG2()
bgMethod2_blur = cv2.createBackgroundSubtractorKNN()

imgIndex = 1
while(cap.isOpened()):
    # 영상이 읽어오고, 존재하지 않으면 while 종료
    ret, frame = cap.read()
    if frame is None:
        break

    # 전경 배경 분리 수행
    frame = cv2.resize(frame, (320, 240))
    if imgIndex != 1:
        bgMOG = bgMethod1.getBackgroundImage()
        bgKNN = bgMethod2.getBackgroundImage()
        bgMOG_blur = bgMethod1_blur.getBackgroundImage()
        bgKNN_blur = bgMethod2_blur.getBackgroundImage()
    fgMOG = bgMethod1.apply(frame, learningRate = -1)
    fgKNN = bgMethod2.apply(frame)
    fgMOG_blur = bgMethod1_blur.apply(cv2.blur(frame, (5,5)), learningRate=-1)
    fgKNN_blur = bgMethod2_blur.apply(cv2.blur(frame, (5,5)))

    print(imgIndex)

    # 영상 저장
    cv2.imwrite("output/" + "mog2_" + str(imgIndex) + ".jpg", fgMOG)
    cv2.imwrite("output/" + "knn_" + str(imgIndex) + ".jpg", fgKNN)
    cv2.imwrite("output/" + "mog2_blur_" + str(imgIndex) + ".jpg", fgMOG_blur)
    cv2.imwrite("output/" + "knn_blur_" + str(imgIndex) + ".jpg", fgKNN_blur)

    imgIndex += 1
    cv2.waitKey(30)

    if imgIndex > 100:
        break
```

7.3 영상 분할 및 합성

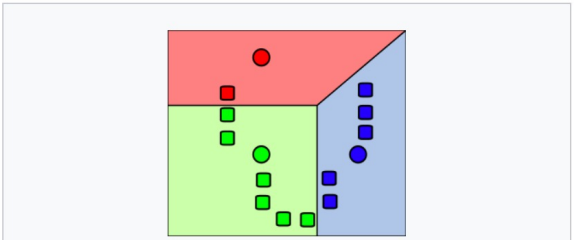
7.3.2 영역 군집화

- 군집화(clustering)은 비슷한 특성을 가지는 픽셀을 한 분류로, 다른 특성을 가지는 픽셀들을 다른 분류로 구분하는 것을 의미함
- 다시 말하면 같은 분류 내에서는 특징의 분산이 최소화되고 (서로 간의 차이가 최소화되고) 분류와 분류 사이의 분산은 최대가 되도록 하는 것을 의미함

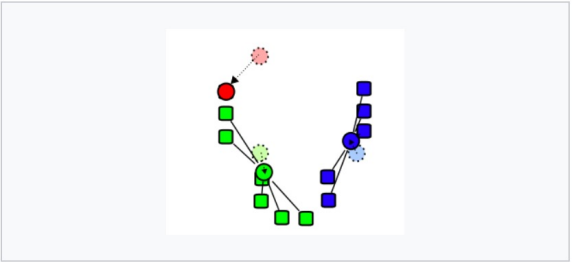
표준 알고리즘의 실행 과정



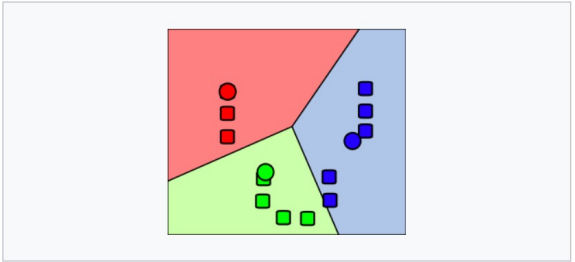
1) 초기 *k* "평균값" (위의 경우 *k*=3) 은 데이터 오브젝트 중에서 무작위로 뽑힌다. (색칠된 동그라미로 표시됨).



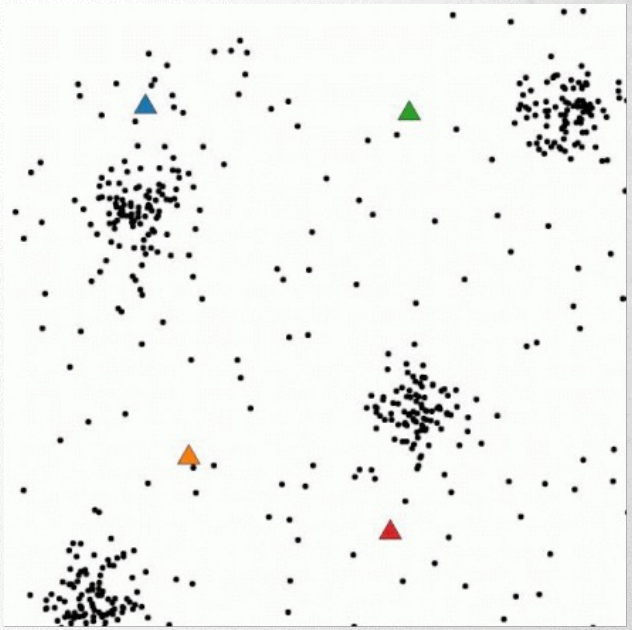
2) *k* 각 데이터 오브젝트들은 가장 가까이 있는 평균값을 기준으로 묶인다. 평균값을 기준으로 분할된 영역은 보로노이 다이어그램으로 표시된다..



3) *k*개의 클러스터의 중심점을 기준으로 평균값이 재조정된다.



4) 수렴할 때까지 2), 3) 과정을 반복한다.



7.3 영상 분할 및 합성

7.3.2 영역 군집화

- OpenCV에서는 K-means clustering 알고리즘을 제공함

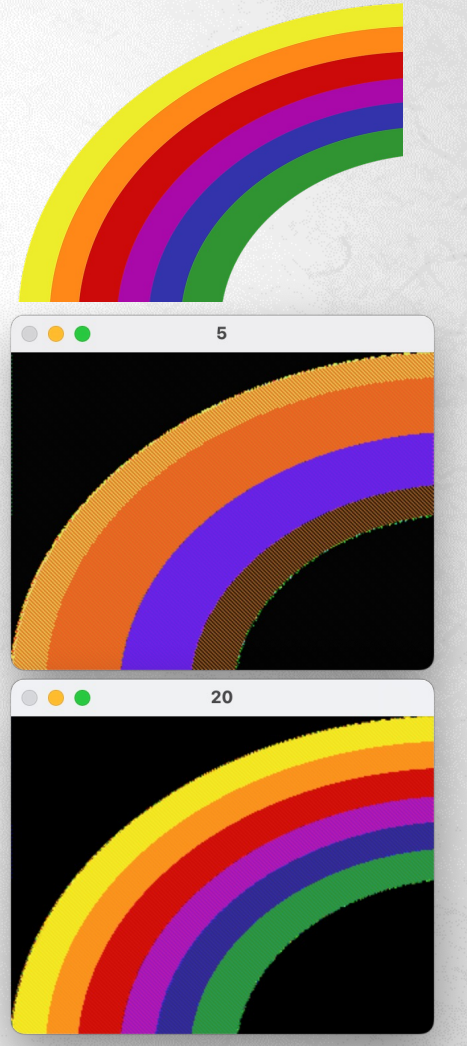
```
cv.kmeans( data, K, bestLabels, criteria, attempts, flags[, centers] ) -> retval, bestLabels, centers
```

data	군집화를 위한 실수형의 입력 데이터 행렬
K	군집화 개수
bestLabels	입력과 출력이 대응하는 레이블(label)이 저장된 행렬
criteria	군집화를 종료하기 위한 조건 (최대 반복횟수나 정확도 중 하나를 지정)
attempts	초기 군집 위치를 다르게 하여 알고리즘을 실행하는 횟수 지정 최종적으로는 가장 우수한 군집을 반환함
flags	초기 중심 위치를 설정하기 위한 플래그
centers	군집화 중심점의 결과 행렬

7.3 영상 분할 및 합성

7.3.2 영역 군집화

```python
import cv2
import numpy as np

# 영상 읽기
img1_src = cv2.imread("./../images/img25.png",
                      cv2.IMREAD_UNCHANGED)
img1 = cv2.resize(img1_src,(320,240))

# knn 입력 생성
data = img1.reshape((-1,3))
data = np.float32(data)

# knn 변수 설정
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
attempts = 10
flags = cv2.KMEANS_RANDOM_CENTERS
for i in range(1, 5):
    # knn 수행
    numK = i * 5
    ret, label, center = cv2.kmeans(data, numK, None, criteria, attempts, flags)

    # 결과 영상 출력 및 저장
    center = np.uint8(center)
    res = center[label.flatten()]
    res = res.reshape((img1.shape))
    cv2.imshow(str(numK),res)
    cv2.waitKey(0)

# 키보드 입력을 기다린 후 모든 영상창 닫기
cv2.waitKey(0)
cv2.destroyAllWindows()
```



7.3 영상 분할 및 합성

도전과제

- 카메라 영상을 입력받고 화면에 출력되도록 합니다.
- q키를 눌러 배경 영상을 촬영합니다.
- 1~5까지의 숫자를 키보드로 입력받도록 합니다.(기본값=1)
- k키를 누르면 입력받은 숫자 * 5한 만큼의 숫자만큼 배경 영상과 카메라 영상에 K-means clustering을 적용합니다.
- b키를 누르면 원래의 배경 제거를 적용하도록 합니다.
- o키를 누르면 입력된 영상이 그대로 출력되도록 합니다.

- K-means clustering을 적용한 차영상과 기본 차영상은 어떻게 다른지 생각해봅시다.
- 또한, 1~5까지 키를 눌러 기본값을 차이를 비교해 봅시다.