

# Recognition and Tracking

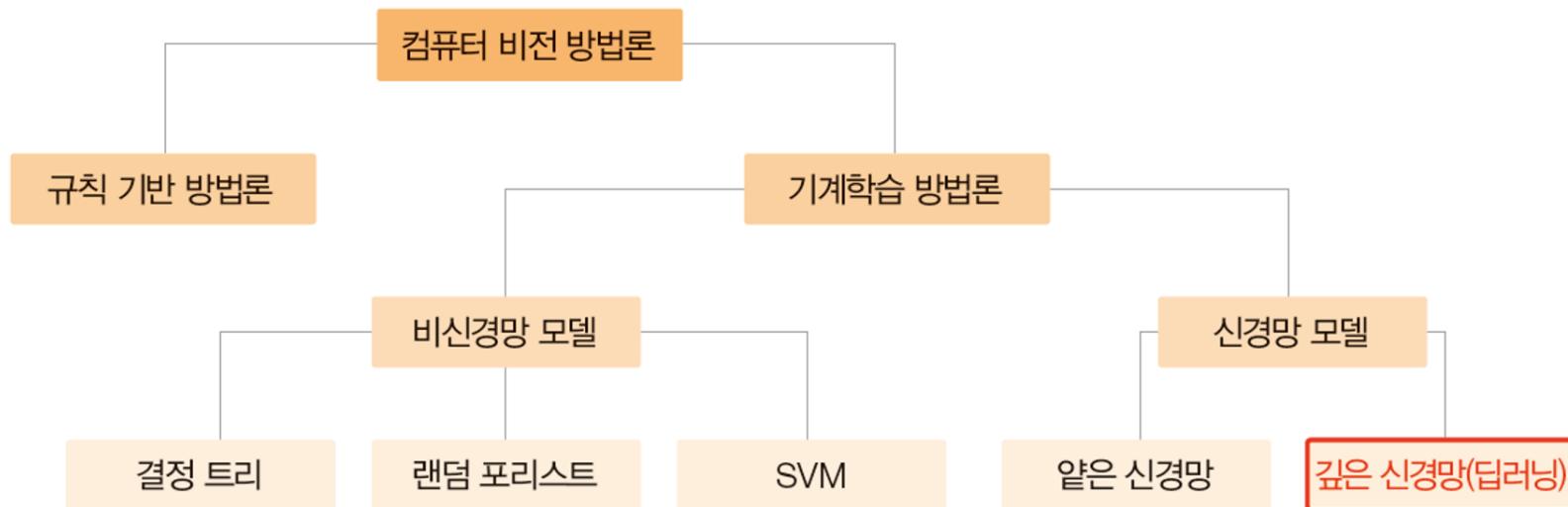
Byeongjoon Noh

Dept. of AI and Bigdata, SCH Univ.

[powernoh@sch.ac.kr](mailto:powernoh@sch.ac.kr)

# Contents

1. Machine learning for recognition
2. Tracking and motion vector
3. AI for recognition
4. SOTA AI for recognition



# 1. ML for Recognition

# Recognition

사람의 영상 해석 능력

- → 심한 혼재(Clutter), 가림(Occlusion)에 강인하고 조명 변환에 불변

인식

- 분류, 검출, 추적 등 모든 세부 문제를 포함/구분
- 각 세부 문제는 별도의 데이터셋 및 성능기준으로 평가 및 개발됨

In this chapter

- 간단한 고전 알고리즘 소개 → (다음 장) AI/딥러닝 approach

# Recognition problems

## Classification

- 영상에 있는 물체의 부류를 알아내는 문제 (Class 확률 벡터 출력)
  - COCO dataset, etc.
- Instance classification, categorical classification

## Detection

- 영상에서 물체의 위치를 찾는 문제 (위치: Bounding box)
- 보통 class 확률도 함께 추론

## Segmentation

- 물체가 점유하는 영역 (화소 집합) 출력
- Sementic segmentation, instance segmentation

# Recognition problems

## Tracking

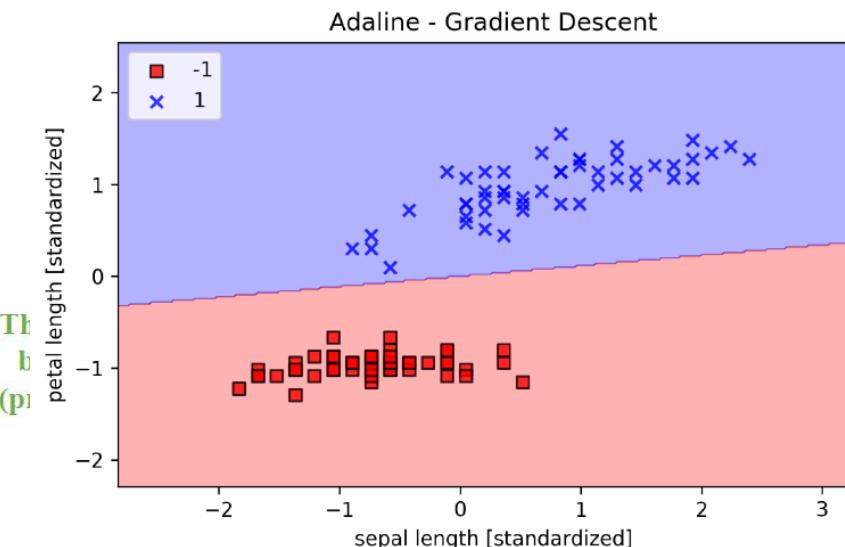
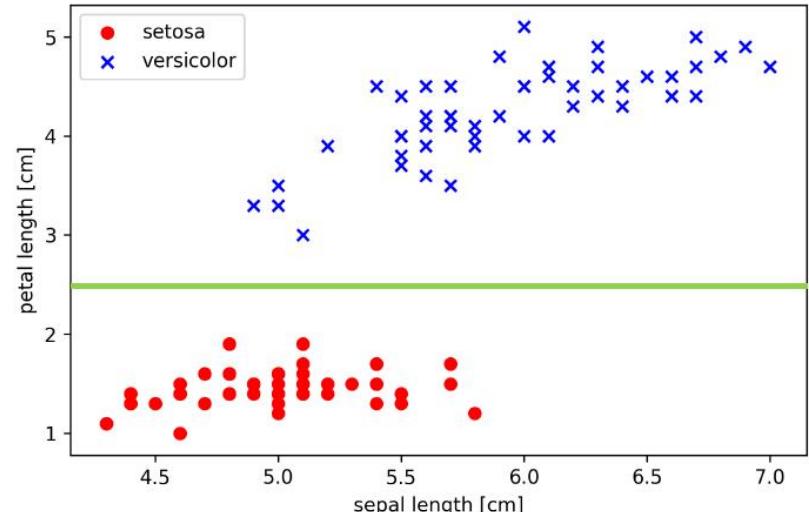
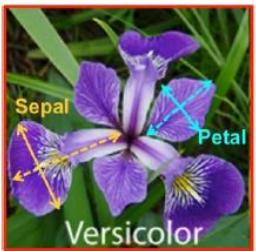
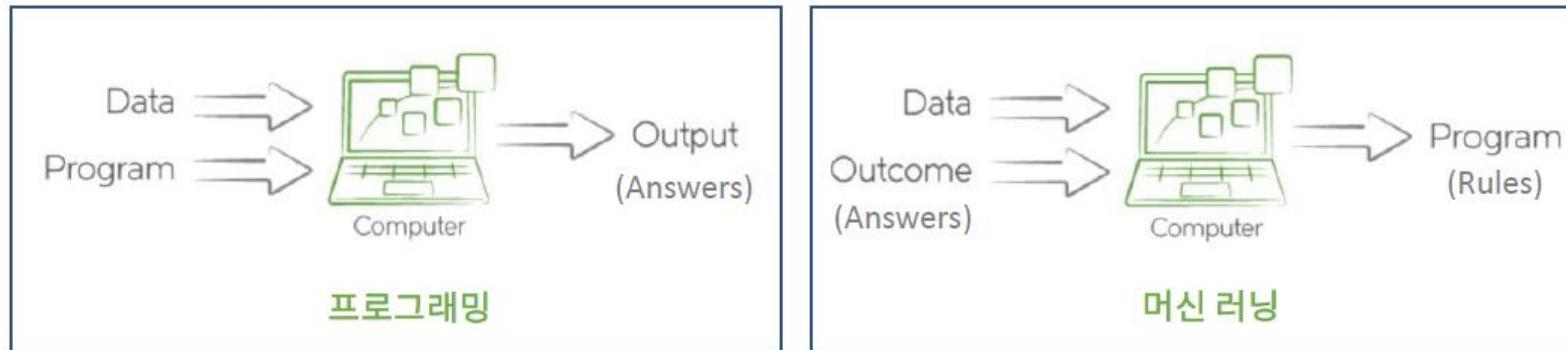
- 비디오에 나타난 물체의 이동 궤적 계산
- Visual Object Tracking (VOT), Multiple Object Tracking (MOT)

## Behavior detection

- 물체가 수행하는 행동의 종류를 알아내는 문제
- 현재는 사람의 행동을 몇 가지로 분류하는 수준

# Machine learning

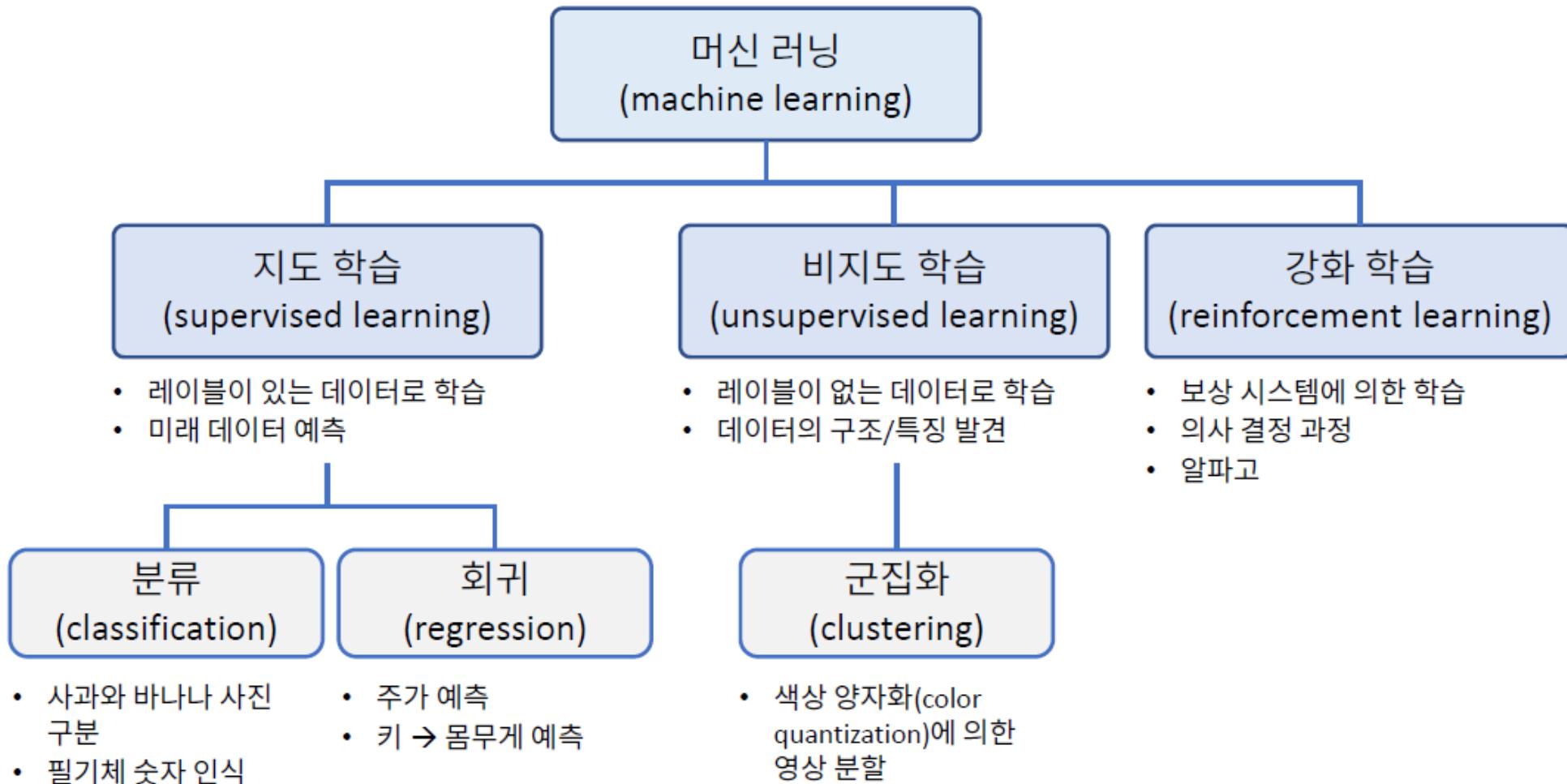
주어진 데이터를 분석하여 규칙성, 패턴 등을 찾고, 이를 이용하여 의미있는 정보를 추출하는 과정



$$\begin{cases} f(x, y) > 0 \rightarrow \text{Versicolor} \\ f(x, y) < 0 \rightarrow \text{Setosa} \end{cases}$$

# Machine learning

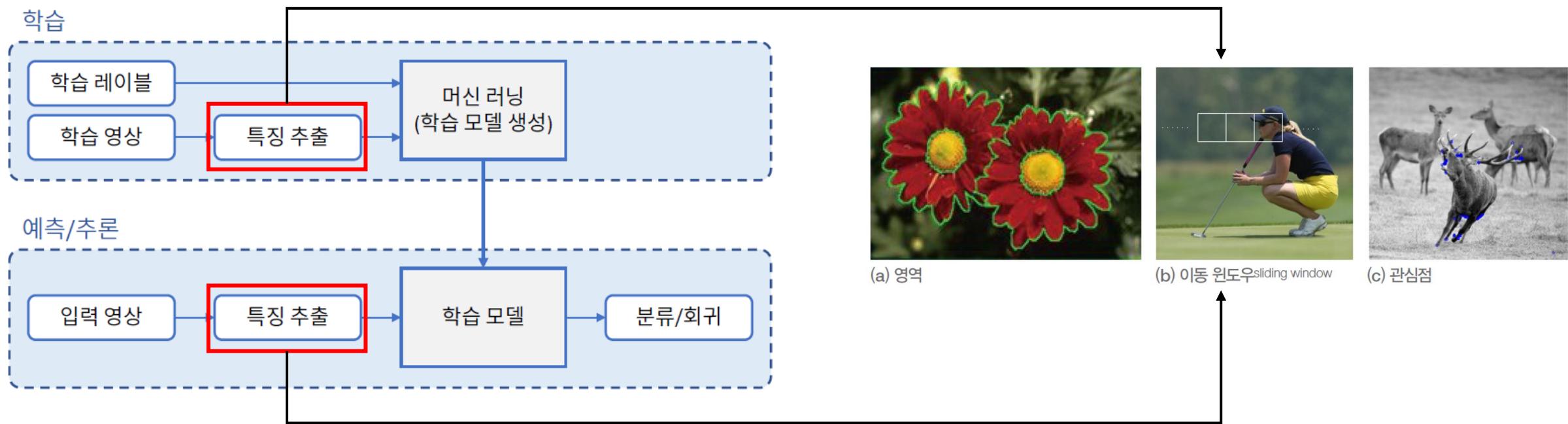
## 머신러닝의 종류



# Machine learning

## 머신러닝의 단계

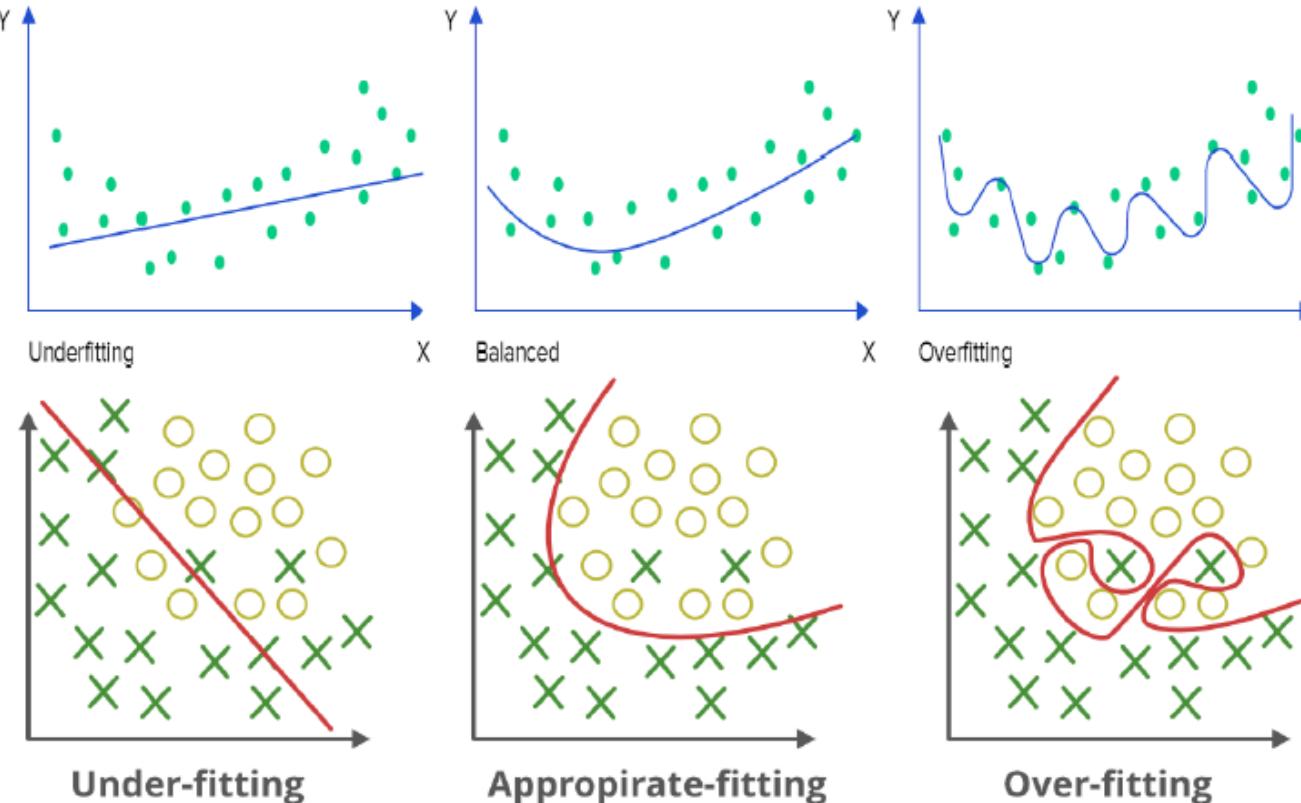
- Training: 학습 데이터를 활용하여 모델을 학습하는 과정
- Prediction/Inference (test): 학습된 모델을 이용하여 새로운 데이터의 적절한 값 예측 (추론)



# Machine learning

## 머신러닝 학습

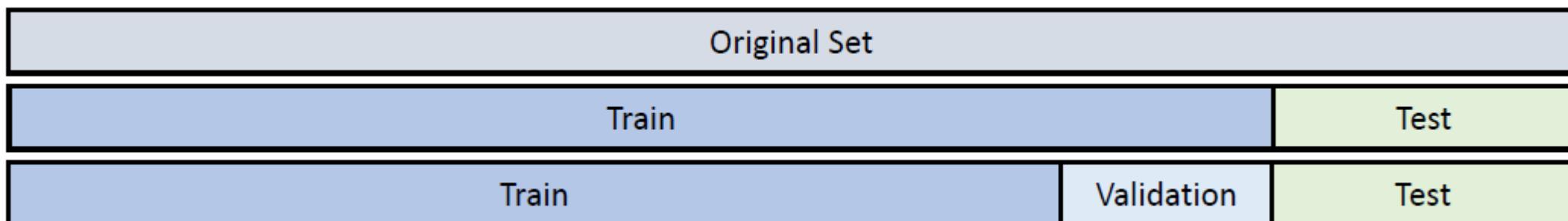
- 미래의 새로운 데이터를 더 정확하게 예측하기 위함
- → 모델의 일반화(Generalization) 성능을 향상 시키는 방향으로 학습해야 함.



# Machine learning

## 머신러닝 학습

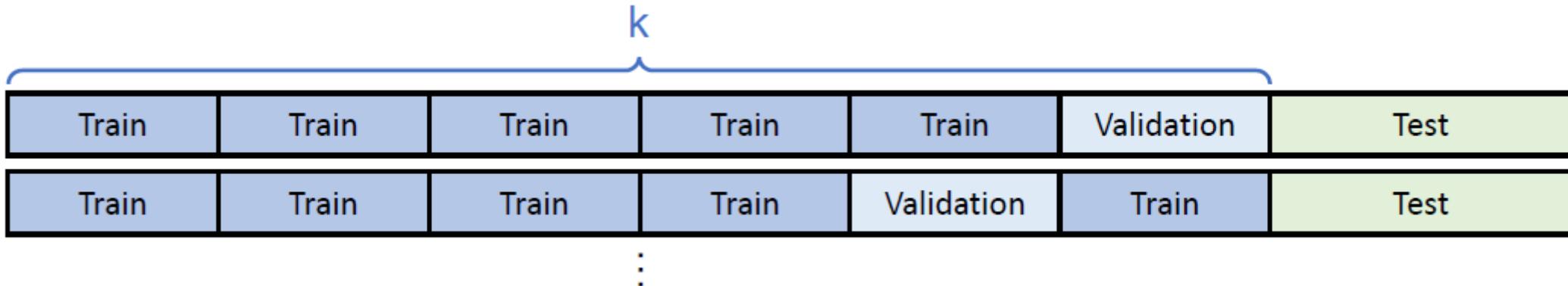
- Overfitting (과적합)
  - 학습 데이터셋을 지나치게 정확하게 구분하도록 학습하여 모델의 일반화 성능이 저하되는 현상
  - 원인 → 데이터 셋 vs 모델?
- 학습 데이터의 분할
  - 학습 가능한 데이터를 학습, 검증, 테스트 데이터셋으로 분할하여 사용



# Machine learning

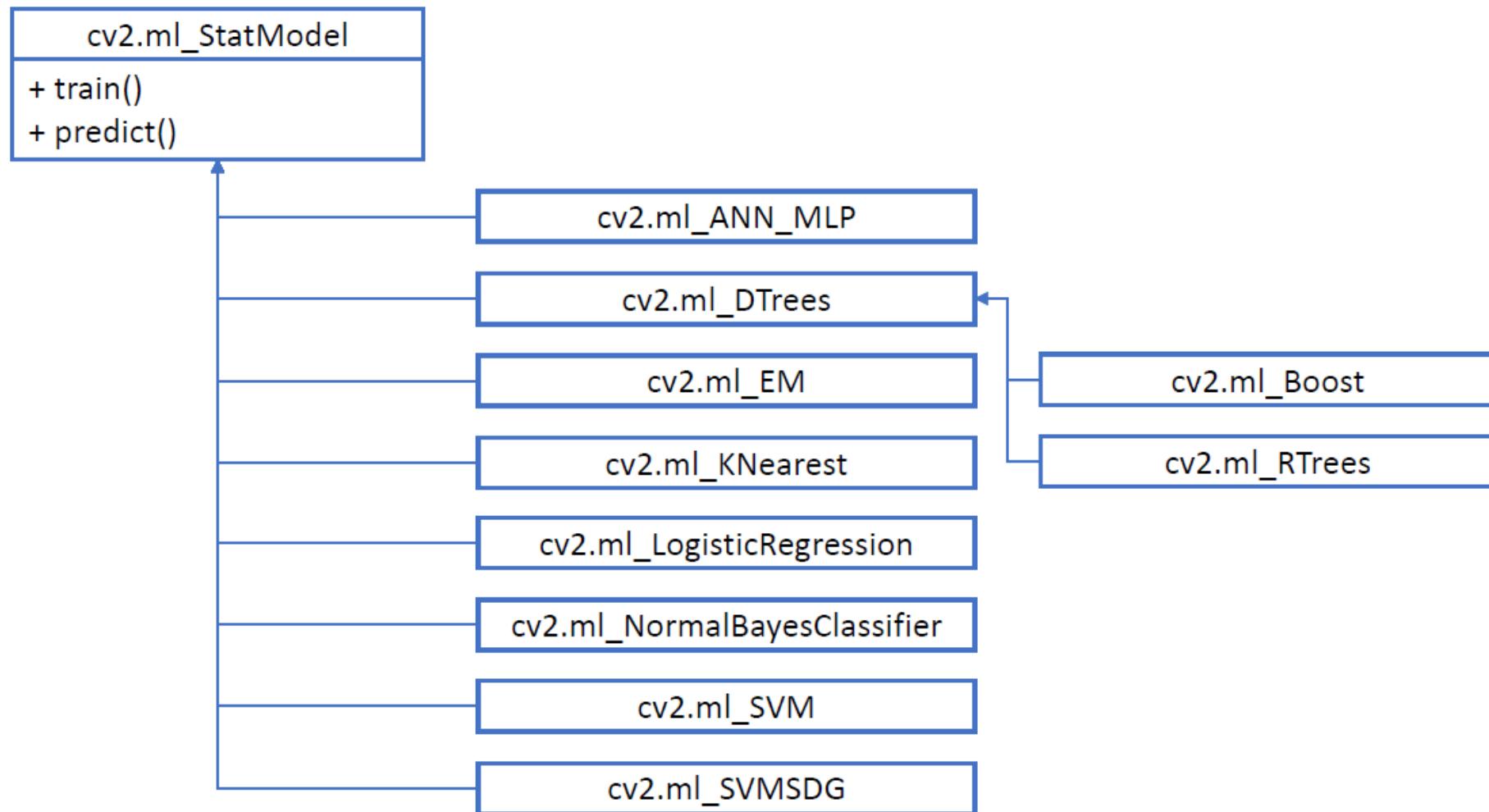
## 머신러닝 학습

- k-fold cross validation
  - 학습 데이터를 k개로 분할하여 여러 번 검증 수행



# Machine learning

## ML class in OpenCV



# Machine learning

## ML class in OpenCV

클래스 이름	설명
ANN_MLP	인공 신경망(artificial neural network) 다층 퍼셉트론(multi-layer perceptron). 여러 개의 은닉층을 포함한 신경망을 학습시킬 수 있고, 입력 데이터에 대한 결과를 예측할 수 있습니다.
DTrees	이진 의사 결정 트리(decision trees) 알고리즘. DTrees 클래스는 다시 부스팅 알고리즘을 구현한 ml::Boost 클래스와 랜덤 트리(random tree) 알고리즘을 구현한 ml::RTree 클래스의 부모 클래스 역할을 합니다.
Boost	부스팅(boosting) 알고리즘. 다수의 약한 분류기(weak classifier)에 적절한 가중치를 부여하여 성능이 좋은 분류기를 만드는 방법입니다.
RTrees	랜덤 트리(random tree) 또는 랜덤 포레스트(random forest) 알고리즘. 입력 특징 벡터를 다수의 트리로 예측하고, 그 결과를 취합하여 분류 또는 회귀를 수행합니다.
EM	기댓값 최대화(Expectation Maximization). 가우시안 혼합 모델(Gaussian mixture model)을 이용한 군집화 알고리즘입니다.

# Machine learning

## ML class in OpenCV

클래스 이름	설명
KNearest	K 최근접 이웃(K-Nearest Neighbors) 알고리즘. K 최근접 이웃 알고리즘은 샘플 데이터와 인접한 K개의 학습 데이터를 찾고, 이 중 가장 많은 개수에 해당하는 클래스를 샘플 데이터 클래스로 지정합니다.
LogisticRegression	로지스틱 회귀(logistic regression). 이진 분류 알고리즘의 일종입니다.
NormalBayesClassifier	정규 베이즈 분류기. 정규 베이즈 분류기는 각 클래스의 특징 벡터가 정규 분포를 따른다고 가정합니다. 따라서 전체 데이터 분포는 가우시안 혼합 모델로 표현 가능합니다. 정규 베이즈 분류기는 학습 데이터로부터 각 클래스의 평균 벡터와 공분산 행렬을 계산하고, 이를 예측에 사용합니다.
SVM	서포트 벡터 머신(support vector machine) 알고리즘. 두 클래스의 데이터를 가장 여유 있게 분리하는 초평면을 구합니다. 커널 기법을 이용하여 비선형 데이터 분류에도 사용할 수 있으며, 다중 클래스 분류 및 회귀에도 적용할 수 있습니다.
SVMSDG	통계적 그래디언트 하향(stochastic gradient descent) SVM. 통계적 그래디언트 하향 방법을 SVM에 적용함으로써 대용량 데이터에 대해서도 빠른 학습이 가능합니다.

# Machine learning

## ML class in OpenCV

- ML 객체 생성

```
cv2.ml.ANN_MLP_create() -> retval  
cv2.ml.KNearest_create() -> retval  
cv2.ml.SVM_create() -> retval  
...
```

# Machine learning

## ML class in OpenCV

- 학습

```
cv2.ml_StatModel.train(samples, layout, responses) -> retval
```

- samples: 학습 데이터 행렬. `numpy.ndarray`. `shape=(N, d)`, `dtype=numpy.float32`.
- layout: 학습 데이터 배치 방법

<code>cv2.ROW_SAMPLE</code>	하나의 데이터가 한 행으로 구성됨
<code>cv2.COL_SAMPLE</code>	하나의 데이터가 한 열로 구성됨

- responses: 각 학습 데이터에 대응되는 응답(레이블) 행렬.  
`numpy.ndarray`. `shape=(N, 1)`, `dtype=numpy.int32` 또는 `numpy.float32`.
- retval: 학습이 성공하면 True.

# Machine learning

## ML class in OpenCV

- 예측(추론)

```
cv2.ml_StatModel.predict(samples, results=None, flags=None) -> retval, results
```

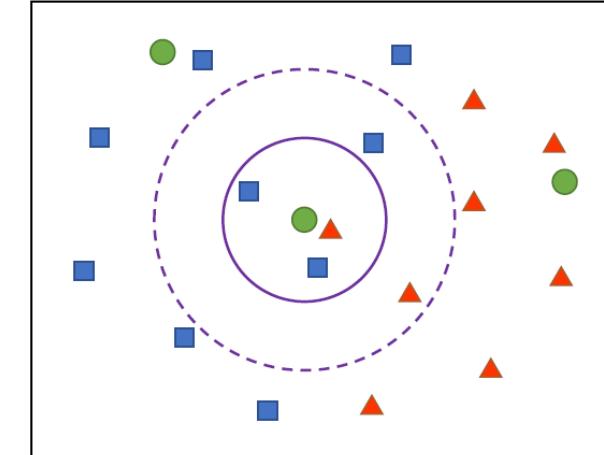
- samples: 입력 벡터가 행 단위로 저장된 행렬.  
`numpy.ndarray`. `shape=(N, d)`, `dtype=numpy.float32`.
- results: 각 입력 샘플에 대한 예측(분류 또는 회귀) 결과를 저장한 행렬.  
`numpy.ndarray`. `shape=(N, )` 또는 `(N, 1)`.  
`dtype=numpy.int32` 또는 `numpy.float32`.
- flags: 추가적인 플래그. 기본값은 0.  
`cv2.ml.STAT_MODEL_RAW_OUTPUT`을 지정하면 클래스 레이블이 아닌 실제 계산 결과 값을 출력.
- retval: 알고리즘에 따라 다름

# Machine learning

## Example) KNN algorithm

- 특징 공간에서 테스트 데이터와 가장 가까이 있는 K개의 학습 데이터를 찾아 분류 또는 회귀를 수행하는 알고리즘
- KNN algorithm 객체 생성

```
cv2.ml.KNearest_create() -> retval
```



- retval: cv2.ml\_KNearest 객체

# Machine learning

## Example) KNN algorithm

- KNN algorithm 입력 데이터의 클래스 예측

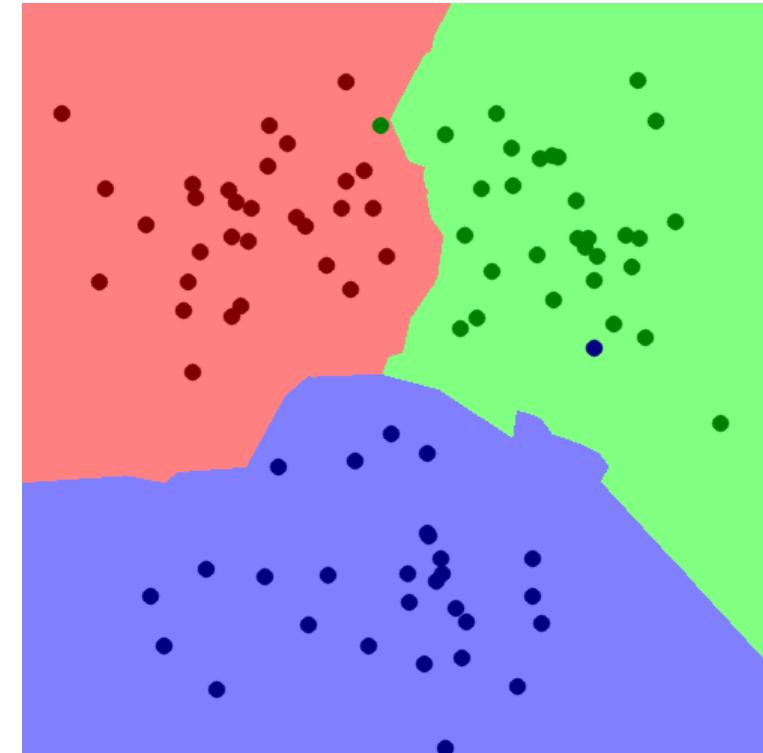
```
cv.ml_KNearest.findNearest(samples, k, results=None,  
                           neighborResponses=None, dist=None, flags=None)  
-> retval, results, neighborResponses, dist
```

- samples: 입력 벡터가 행 단위로 저장된 입력 샘플 행렬.  
`numpy.ndarray`. shape=(N, d), dtype=`numpy.float32`.
- k: 사용할 최근접 이웃 개수
- results: 각 입력 샘플에 대한 예측(분류 또는 회귀) 결과를 저장한 행렬.  
`numpy.ndarray`. shape=(N, 1), dtype=`numpy.float32`.
- neighborResponses: 예측에 사용된 k개의 최근접 이웃 클래스 정보 행렬.  
`numpy.ndarray`. shape=(N, k), dtype=`numpy.float32`.
- dist: 입력 벡터와 예측에 사용된 k개의 최근접 이웃과의 거리를 저장한 행렬.  
`numpy.ndarray`. shape=(N, k), dtype=`numpy.float32`.
- retval: 입력 벡터가 하나인 경우에 대한 응답

# Machine learning

Example) KNN algorithm

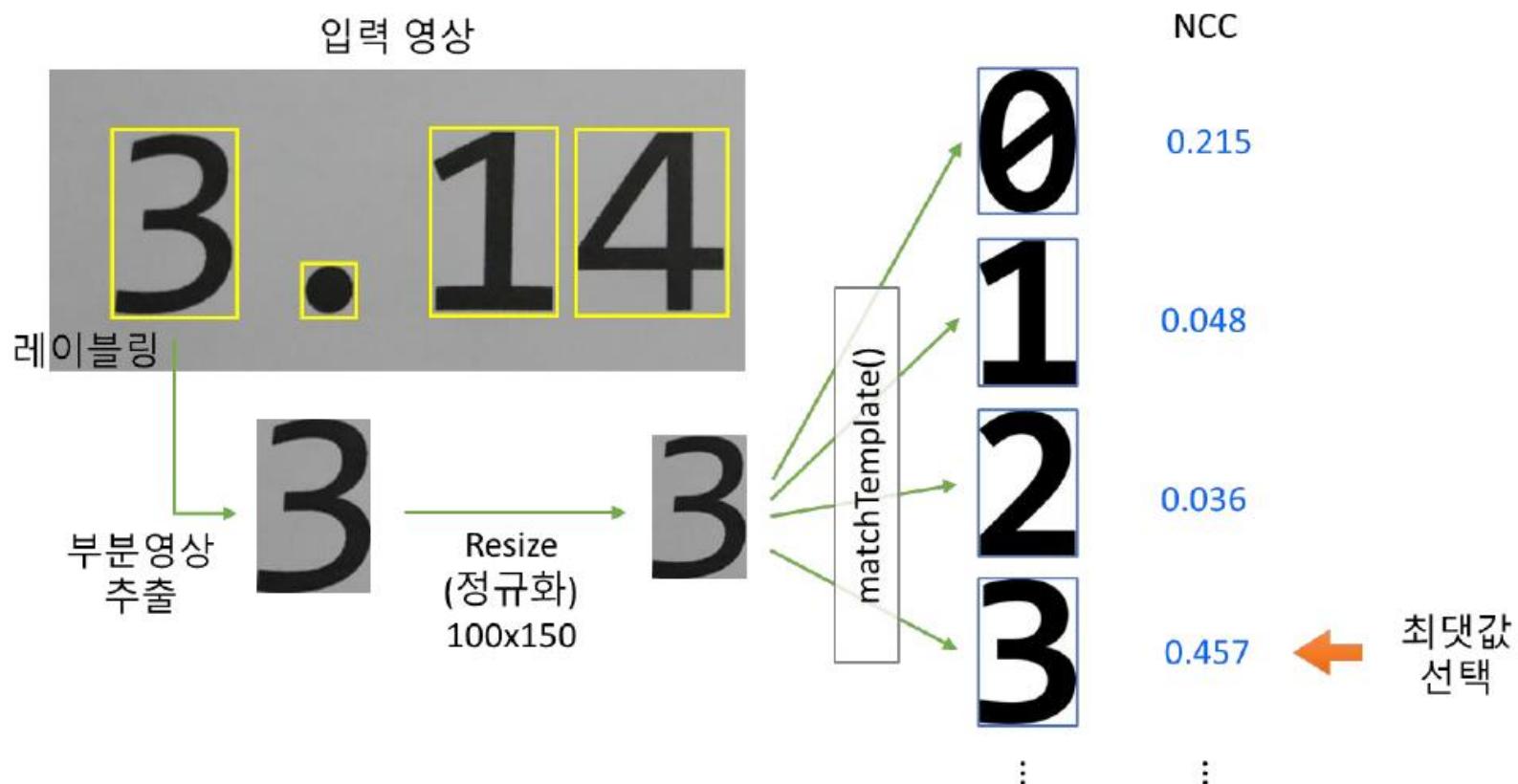
- 5\_1\_KNN\_Basic.py
- 5\_2\_KNN\_Vis.py



# Machine learning

Example) 필기체 숫자 인식

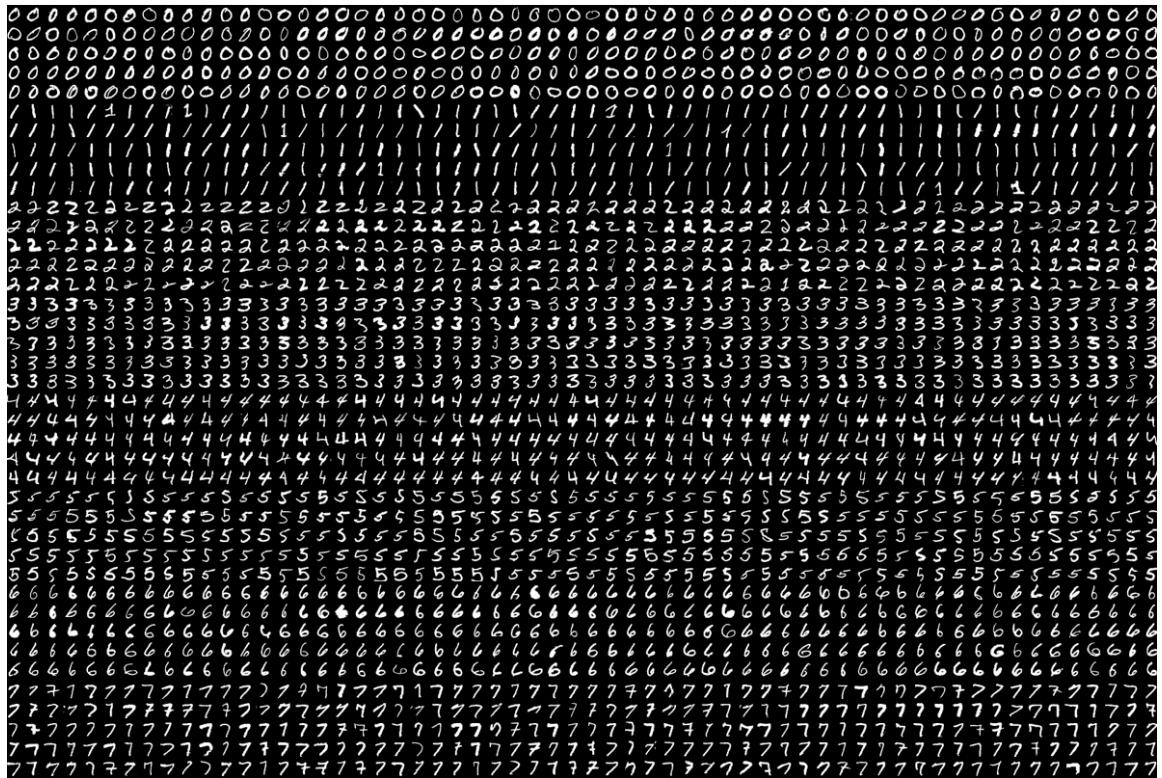
- 전략



# Machine learning

## Example) 필기체 숫자 인식

- 학습 데이터셋 확보 → OpenCV가 일부 제공
- <https://github.com/opencv/opencv/blob/master/samples/data/digits.png>

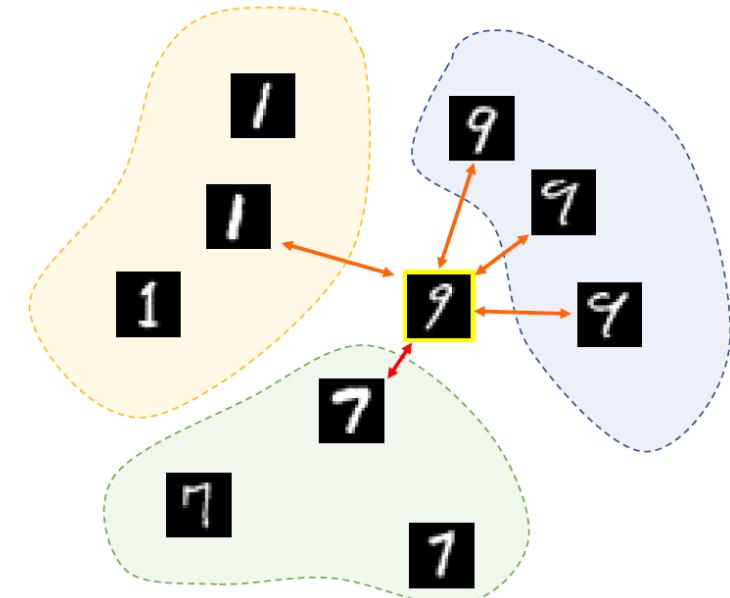
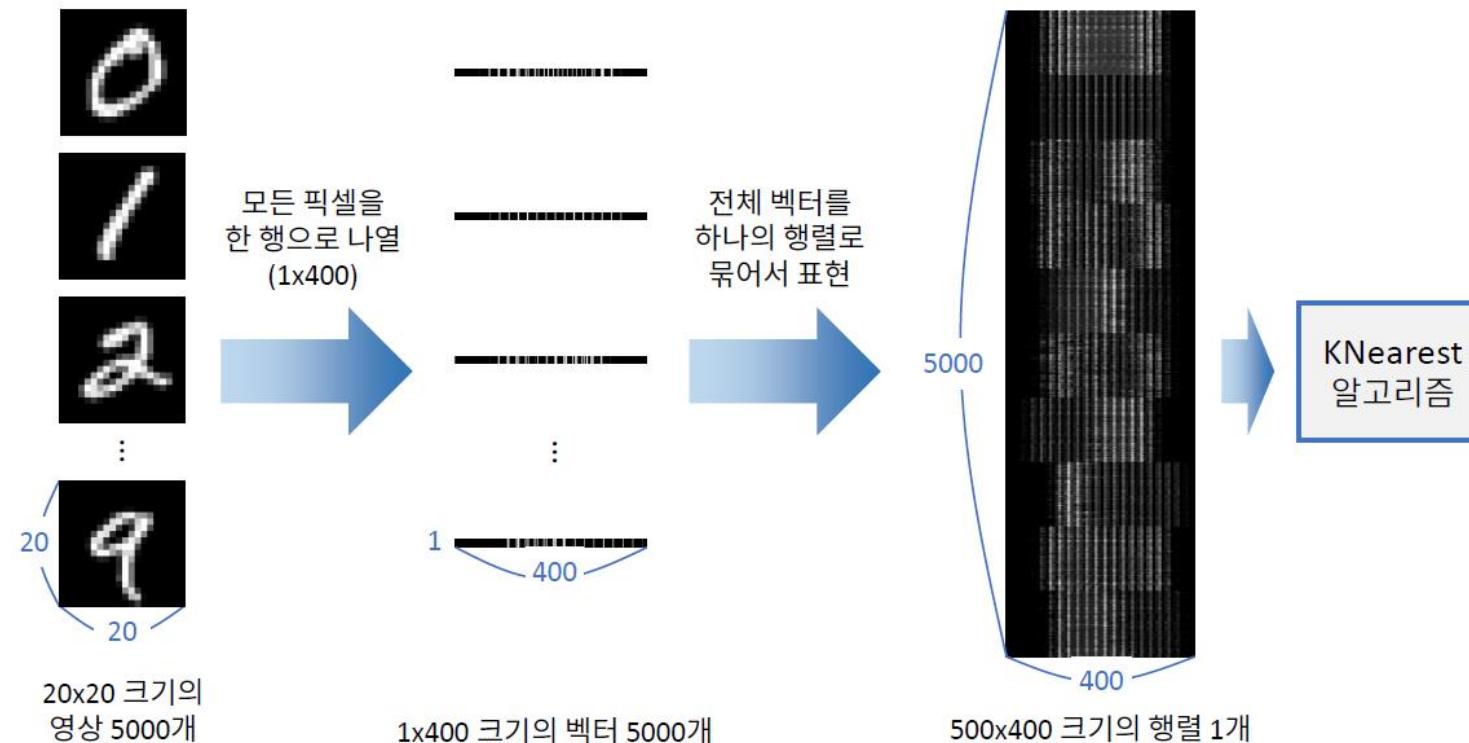


master		opencv / samples / data /	Go to file
This branch is 44 commits behind 4.x.			
	alalek	Merge remote-tracking branch 'upstream/3.4' into merge-3.4	✓ 5d14cc6 on Nov 17, 2022 History
..			
	dnn	ADD weights yolov4 in models.yml	7 months ago
	Blender_Suzanne1.jpg	Add Demo 5: Basic panorama stitching from a rotating camera in the ho...	6 years ago
	Blender_Suzanne2.jpg	Add Demo 5: Basic panorama stitching from a rotating camera in the ho...	6 years ago
	H1to3p.xml	fix files permissions	3 years ago
	HappyFish.jpg	fixed display_image tutorial sample	9 years ago
	LinuxLogo.jpg	fix files permissions	3 years ago
	Megamind.avi	Tutorials: moved Megamind.avi files to samples/data folder	8 years ago
	Megamind_buggy.avi	Tutorials: moved Megamind.avi files to samples/data folder	8 years ago
	WindowsLogo.jpg	fix files permissions	3 years ago
	aero1.jpg	initial commit	9 years ago
	aero3.jpg	initial commit	9 years ago
	aloeGT.png	Added ground-truth disparity map for 'aloe'	8 years ago
	aloeL.jpg	initial commit	9 years ago
	aloeR.jpg	initial commit	9 years ago
	alphabet_36.txt	Merge pull request #17570 from HannibalAPE:text_det_recog_demo	3 years ago
	alphabet_94.txt	Merge pull request #17570 from HannibalAPE:text_det_recog_demo	3 years ago

# Machine learning

Example) 필기체 숫자 인식

- 20x20 크기 영상의 픽셀값을 이용해서 400차원 공간에서 한 점의 좌표를 생성
- 400차원 공간에서 KNN algorithm 점 분류



# Machine learning

Example) 필기체 숫자 인식

- 5\_3\_KNN\_DigitRecog.py

```
train_images = cells.reshape(-1, 400).astype(np.float32)
train_labels = np.repeat(np.arange(10), len(train_images)/10)
```

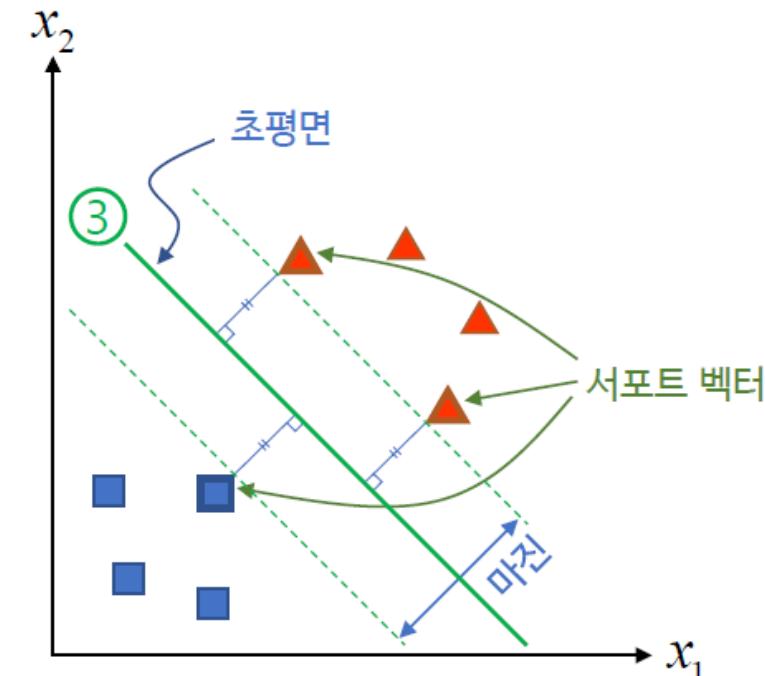
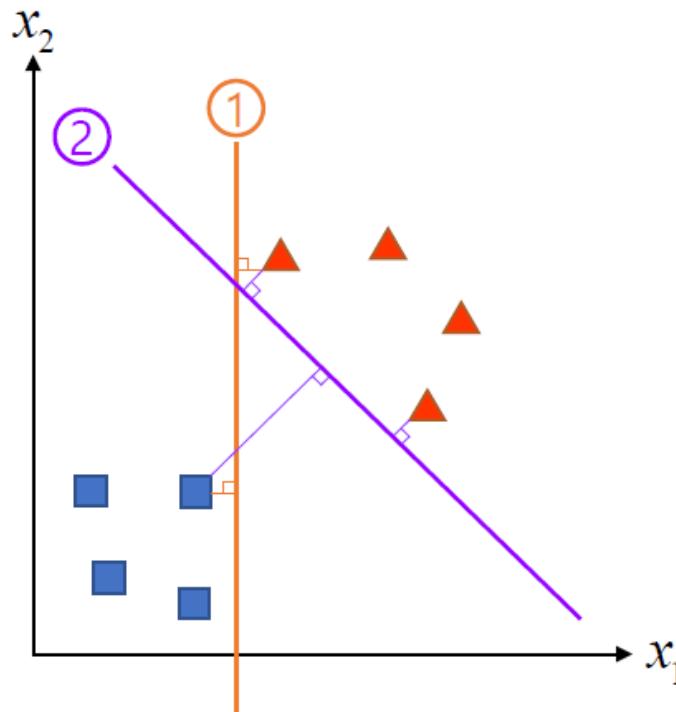
```
test_image = cv2.resize(img, (20, 20), interpolation=cv2.INTER_AREA)
test_image = test_image.reshape(-1, 400).astype(np.float32)
```



# Machine learning

## Example) Support Vector Machine (SVM)

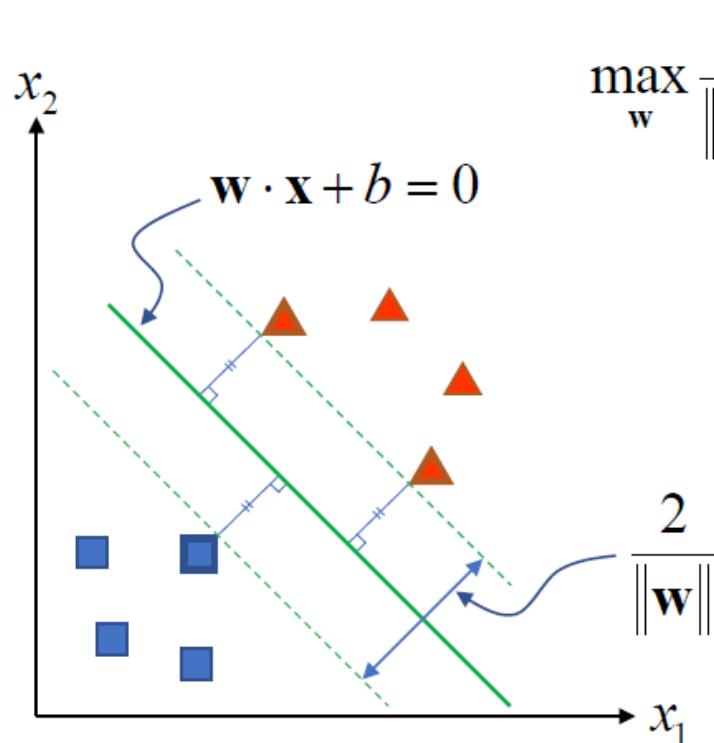
- 두 개의 그룹(데이터)을 분리하는 기계학습 방법 중 하나
- 데이터들과 거리가 가장 먼 초평면(hyperplane)을 선택하는 방법
  - Maximum margin classifier



# Machine learning

## Example) Support Vector Machine (SVM)

- How to obtain maximum marginal hyperplane?
  - $\mathbf{w} \cdot \mathbf{x} + b = 0 \leftarrow \text{hyperplane}$



$$\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|} \rightarrow \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2$$

조건부 최적화 문제  
Constraint definition  
Lagrange multiplier  
KKT condition, etc.

Kernel trick  
Polynomial kernel  
Radial basis function

# Machine learning

## Example) Support Vector Machine (SVM)

- SVM 객체 생성

```
cv2.ml.SVM_create() -> retval
```

- retval: cv2.ml\_SVM 객체

# Machine learning

## Example) Support Vector Machine (SVM)

- SVM 타입 지정

```
cv.ml_SVM.setType(type) -> None
```

- type: SVM 종류 지정. cv2.ml.SVM\_ 으로 시작하는 상수.

type	설명	파라미터
cv2.ml.SVM_C_SVC	C-서포트 벡터 분류. 일반적인 n-클래스 분류 문제에서 사용됩니다.	C
cv2.ml.SVM_NU_SVC	v-서포트 벡터 분류. C_SVC와 비슷하지만 Nu 값 범위가 0~1 사이로 정규화되어 있습니다.	Nu
cv2.ml.SVM_ONE_CLASS	1-분류 서포트 벡터 머신. 데이터 분포 측정에 사용됩니다.	C, Nu
cv2.ml.SVM_EPS_SVR	$\epsilon$ -서포트 벡터 회귀	P, C
cv2.ml.SVM_NU_SVR	v-서포트 벡터 회귀	Nu, C

# Machine learning

## Example) Support Vector Machine (SVM)

- SVM 커널 지정

```
cv.ml_SVM.setKernel(kernelType) -> None
```

- kernelType: 커널 함수 종류 지정. cv2.ml.SVM\_ 으로 시작하는 상수.

kernelType	설명	파라미터
cv2.ml.SVM_LINEAR	선형 커널	
cv2.ml.SVM_POLY	다항식 커널	Degree, Gamma, Coef0
cv2.ml.SVM_RBF	방사 기저 함수 커널	Gamma
cv2.ml.SVM_SIGMOID	시그모이드 커널	Gamma, Coef0
cv2.ml.SVM_CHI2	지수 카이 제곱 커널	Gamma
cv2.ml.SVM_INTER	히스토그램 교차 커널	

# Machine learning

## Example) Support Vector Machine (SVM)

- SVM 모델 학습

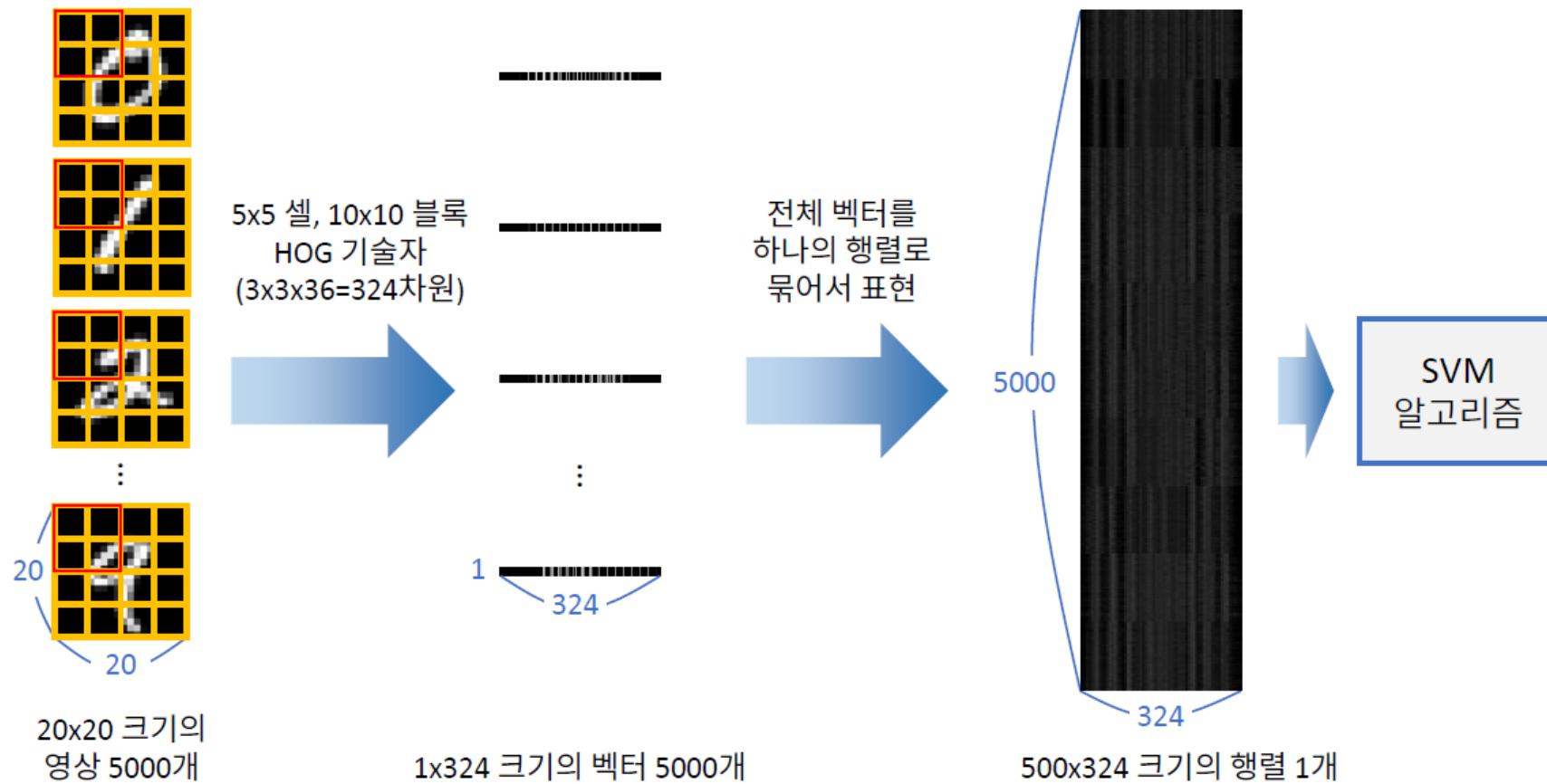
```
cv.ml_SVM.trainAuto(samples, layout, responses, kFold=None, ...) -> retval
```

- samples: 학습 데이터 행렬. `numpy.ndarray`. `shape=(N, d)`, `dtype=numpy.float32`.
- layout: 학습 데이터 배치 방법. `cv2.ROW_SAMPLE` 또는 `cv2.COL_SAMPLE`.
- responses: 각 학습 데이터에 대응되는 응답(레이블) 벡터. `numpy.ndarray`.  
`shape=(N, )` 또는 `(N, 1)`. `dtype=numpy.int32` 또는 `numpy.float32`.
- kFold: 교차 검증을 위한 부분 집합 개수
- retval: 학습이 정상적으로 완료되면 True

# Machine learning

Example) Support Vector Machine (SVM) 을 이용한 필기체 숫자 인식

- HOG 특징 벡터 활용



# Machine learning

Example) Support Vector Machine (SVM) 을 이용한 필기체 숫자 인식

- 5\_4\_SVM.py
- 5\_5\_SVM\_HOG\_DigitRecog.py

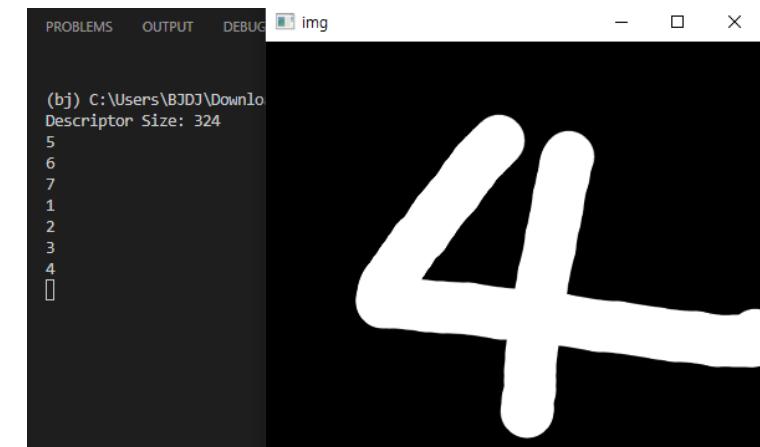
```
hog = cv2.HOGDescriptor((20, 20), (10, 10), (5, 5), (5, 5), 9)
```

```
...
```

```
desc = []
for img in cells:
    desc.append(hog.compute(img))
```

```
...
```

```
train_desc = np.array(desc)
train_desc = train_desc.squeeze().astype(np.float32)
train_labels = np.repeat(np.arange(10), len(train_desc)/10)
```



# Machine learning

Example) Support Vector Machine (SVM) 을 이용한 필기체 숫자 인식

- 5\_4\_SVM.py
- 5\_5\_SVM\_HOG\_DigitRecog.py

```
svm = cv2.ml.SVM_create()
svm.setType(cv2.ml.SVM_C_SVC)
svm.setKernel(cv2.ml.SVM_RBF)
svm.setC(2.5)
svm.setGamma(0.50625)

...
test_image = cv2.resize(img, (20, 20), interpolation=cv2.INTER_AREA)
test_desc = hog.compute(test_image).reshape(-1, 1).T

_, res = svm.predict(test_desc)
```

# Machine learning

Example) Support Vector Machine (SVM) 을 이용한 필기체 숫자 인식 2

- 정중앙에 숫자를 쓰지 않는 경우 → 오인식

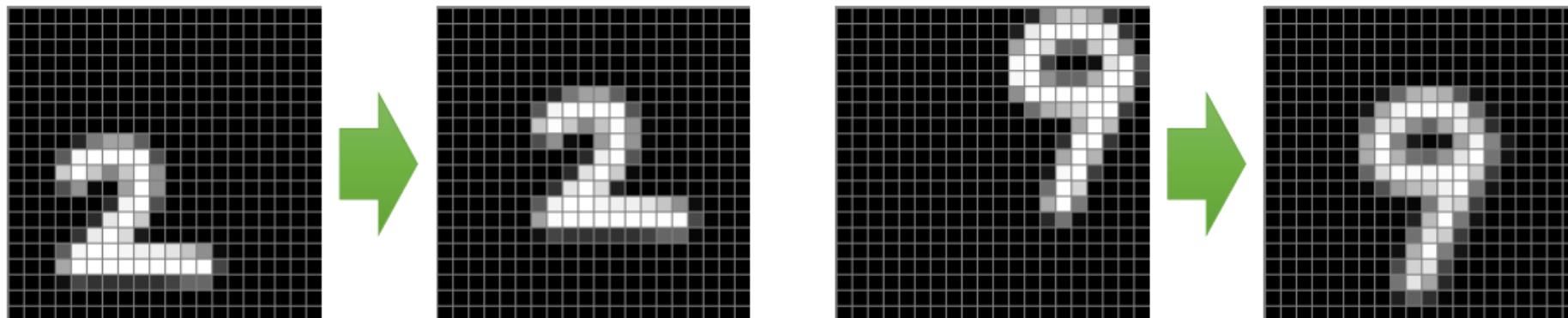


- 학습 데이터 영상 정규화 (Normalization)
  - 학습 데이터 영상과 테스트 데이터 영상의 위치, 크기, 회전 요소를 정규화
  - 어디에 숫자가 있든지 인식 가능 (성능 향상)

# Machine learning

Example) Support Vector Machine (SVM) 을 이용한 필기체 숫자 인식 2

- How?
  - 숫자 영상 무게 중심이 전체 영상 중앙이 되도록 위치 정규화
  - 영상의 무게 중심  $(\bar{x}, \bar{y}) = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$
  - 무게 중심: 흰색 픽셀의 x, y좌표의 합 / 전체 픽셀 수



# Machine learning

Example) Support Vector Machine (SVM) 을 이용한 필기체 숫자 인식 2

- 5\_6\_SVM\_HOG\_DigitRecog\_Norm.py

```
def norm_digit(img):
    m = cv2.moments(img)
    cx = m['m10'] / m['m00']
    cy = m['m01'] / m['m00']
    h, w = img.shape[:2]
    aff = np.array([[1, 0, w/2 - cx], [0, 1, h/2 - cy]], dtype=np.float32)
    dst = cv2.warpAffine(img, aff, (0, 0))
    return dst

...
test_image = cv2.resize(img, (20, 20), interpolation=cv2.INTER_AREA)
test_image = norm_digit(test_image)
test_desc = hog.compute(test_image).reshape(-1, 1).T
_, res = svm.predict(test_desc)
```

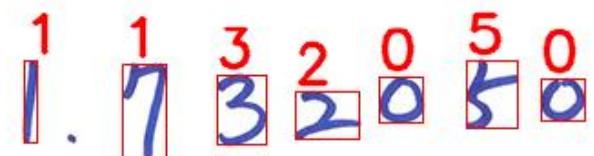
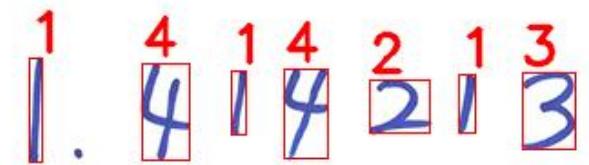
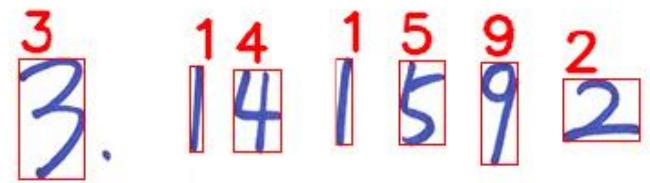
# Machine learning

Example) Support Vector Machine (SVM) 을 이용한 필기체 숫자 인식 3

- 미리 학습된 SVM 모델 loading 하여 활용하기
- 5\_7\_SVM\_YAML\_DigitalRecog.py

```
opencv_ml_svm:  
    format: 3  
    svmType: C_SVC  
    kernel:  
        type: RBF  
        gamma: 5.062499999999998e-01  
    C: 2.500000000000000e+00  
    term_criteria: { epsilon:1.1920928955078125e-07, iterations:1000 }  
    var_count: 324  
    class_count: 10  
    class_labels: !!opencv-matrix  
        rows: 10  
        cols: 1  
        dt: i  
        data: [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ]  
    sv_total: 2099  
    support_vectors:  
        - [ 0., 0., 0., 0., 0., 0., 0., 8.58905539e-02, 0., 0., 0.,  
            0., 0., 0., 6.15069084e-02, 1.92065626e-01, 1.91699609e-01,  
            3.08994293e-01, 3.87710929e-02, 0., 0., 0., 0., 3.92445689e-03,  
            4.86440510e-01, 4.86440510e-01, 4.86440510e-01, 1.55084327e-01,  
            6.33909088e-03, 2.20010877e-02, 0., 0., 2.80889213e-01,  
            7.07933605e-02, 5.13778962e-02, 6.54745335e-03, 0., 0., 0., 0.,
```

```
# 미리 학습된 SVM 데이터 불러오기  
svm = cv2.ml.SVM_load('..../data/svmdigits.yml')
```

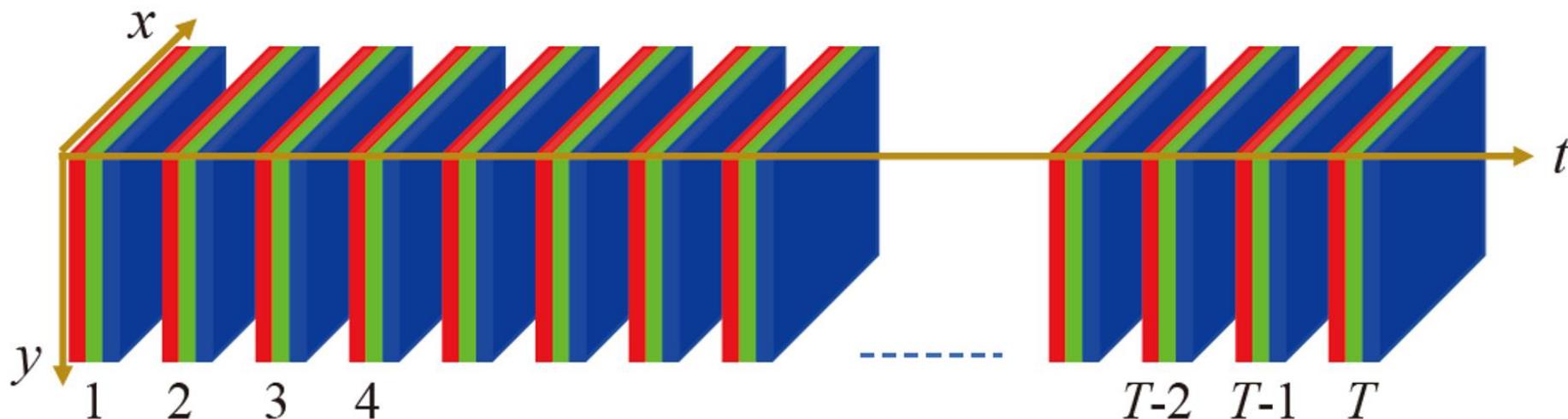


## 2. Tracking and Motion Vector

# Preview

Video (with motioned-scene)

- 동적 비전이 처리하는 연속 영상
- 보통 30 frames/sec (30 FPS)
- 3차원 시공간(Spatio-temporal) 형성  $f(y, x, t); 1 \leq t \leq T$
- 비디오:  $f(y, x, c, t)$ 의 4차원(c=RGB) 텐서 형성



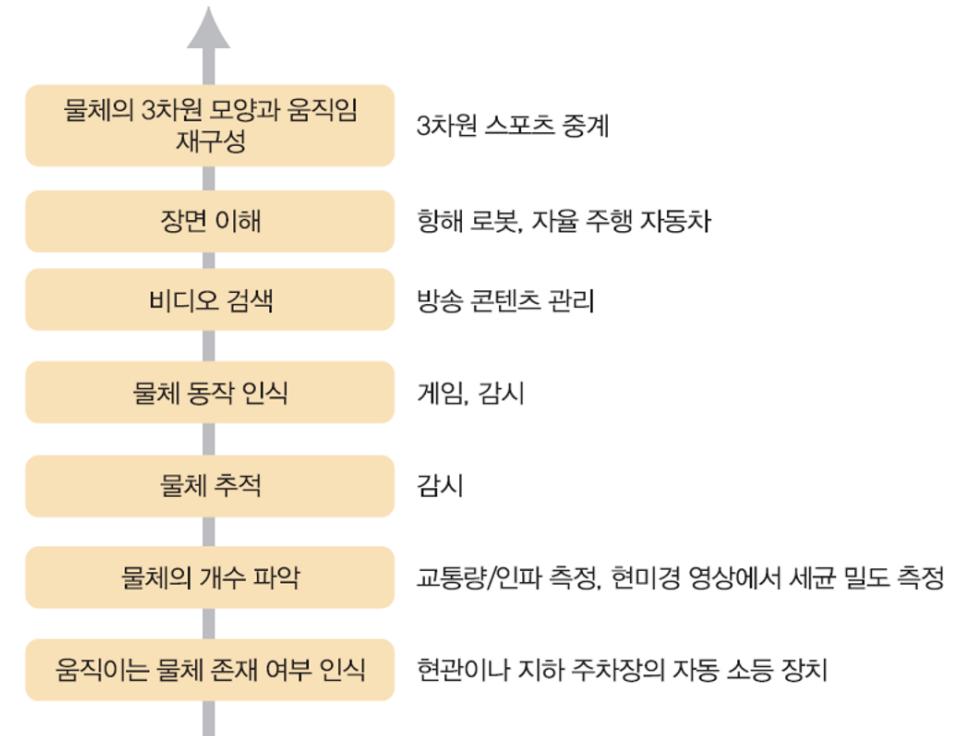
# Preview

여러 움직임 상황

- 정지 카메라 + 정지 장면 → No motion
- 정지 카메라 + 동적 장면
  - 과속 단속 카메라, 감시용 카메라 등
- 동적 카메라 + 정적 장면
  - 불법 주차 촬영 차량
- 동적 카메라 + 동적 장면
  - 자율주행 차량, 스포츠 중계, 항해로봇 등

알아 내야할 정보

- 물체가 움직이는 방향과 속도
- 물체에 대한 행위 인식 (최종 목표)



## In this chapter...

배경 차분 방법 → 초기 연구

- 정적 배경 (Static background) 차분
- 이동 평균 (Moving average) 배경
- Mixture of Gaussian (MOG), Gaussian Mixture Model (GMM)

추적 방법

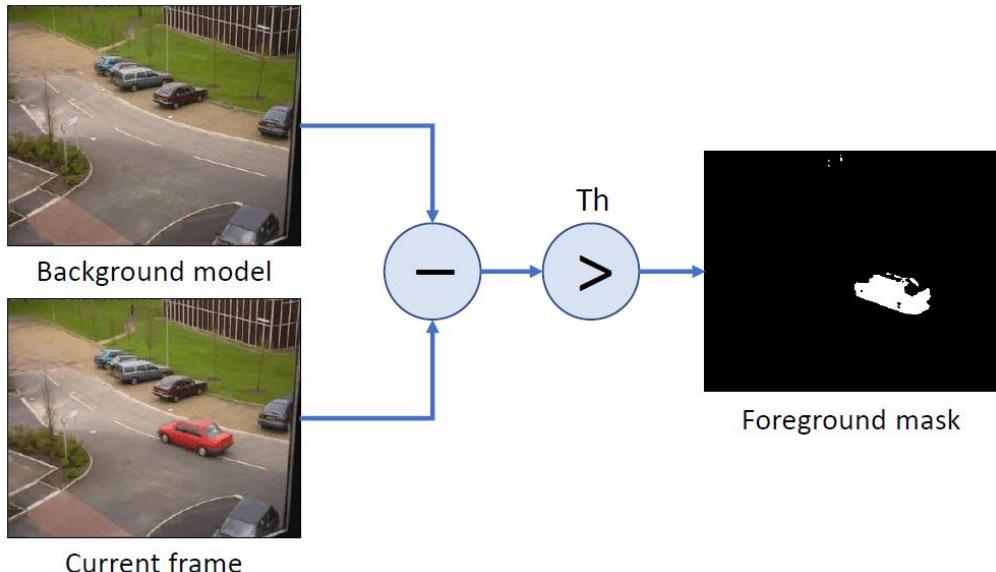
- 평균 이동 (Mean shift) 알고리즘
- Continuously Adaptive Mean (CAM) shift 알고리즘
- Optical flow → 최근까지...
- OpenCV tracker

Nowadays → deep learning..

# Static background subtraction

## 정적 배경 차분

- 배경과 현재 입력 프레임과의 차영상(Frame difference)을 이용하여 전경 객체를 검출
- 움직이는 전경 객체 검출을 위한 기본적인 방법



$$d(y, x) = \begin{cases} 1, & |f(y, x, t) - f(y, x, r)| > \tau \\ 0, & \text{else} \end{cases}$$

# Static background subtraction

정적 배경 차분 예시

- 5\_8\_1\_Static\_BS.py



```
# cap = cv2.VideoCapture(0)
cap = cv2.VideoCapture(path)

# 배경 영상 등록
ret, back = cap.read()
back = cv2.cvtColor(back, cv2.COLOR_BGR2GRAY)

while True:

    ret, frame = cap.read()

    if ret:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Frame difference
        diff = cv2.absdiff(gray, back)

        _, diff = cv2.threshold(diff, 30, 255, cv2.THRESH_BINARY)

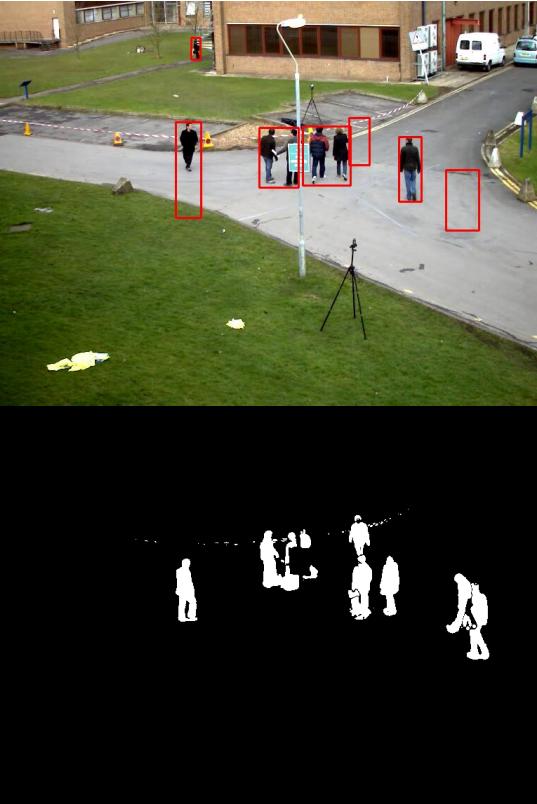
        # Exit if the 'q' key is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

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

# Static background subtraction

정적 배경을 이용한 전경 객체 검출 후 주요 객체 바운딩 박스 표시

- 레이블링 수행 후 픽셀 개수가 100 이상인 객체에 바운딩 박스 표시
- 5\_8\_2\_Static\_BS2.py



```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (0, 0), 1.0)

diff = cv2.absdiff(gray, back)
_, diff = cv2.threshold(diff, 30, 255, cv2.THRESH_BINARY)
cnt, _, stats, _ = cv2.connectedComponentsWithStats(diff)

for i in range(1, cnt):
    x, y, w, h, s = stats[i]

    if s < 100:
        continue

    cv2.rectangle(frame, (x, y, w, h), (0, 0, 255), 2)
```

# Moving average background

정적 배경 모델 사용 시 문제점

- 미리 등록된 기준 영상이 실제 배경과 크게 달라질 경우 오동작
  - 그림자 등으로 인한 조도 변경, 새로운 객체가 화면에 고정될 경우



고정 객체 발생



현재 프레임 (#840)



차영상

# Moving average background

## Idea

- 움직이는 객체가 존재하는 수백 장의 입력 영상으로부터 평균 영상을 구하면?
- 평균 연산에 의해 배경 영상이 생성됨



- 수백 장의 이전 프레임을 buffer에 저장하려면 대용량 메모리가 필요

# Moving average background

## Moving average

- 수백 장의 영상을 저장하는 대신 매 프레임이 들어올 때마다 평균 영상을 갱신
  - Dynamic programming

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

갱신된 배경 영상

현재 프레임

현재 프레임에 대한 가중치  
( $0 < \alpha < 1$ )

이전 프레임까지의 배경 영상

- 대용량 메모리 필요 없음
- In OpenCV...
  - `cv2.accumulatedWeighted(src, dst, alpha, mask) → dst`

# Moving average background

이동 평균에 의한 배경 차분 예제

- 5\_9\_BS\_MovAvg.py



```
ret, back = cap.read()

if not ret:
    print('Background image registration failed!')
    sys.exit()

back = cv2.cvtColor(back, cv2.COLOR_BGR2GRAY)
back = cv2.GaussianBlur(back, (0, 0), 1.0)
fback = back.astype(np.float32)
```

# Moving average background

이동 평균에 의한 배경 차분 예제

- 5\_9\_BS\_MovAvg.py

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

- src: 입력 영상. 1 또는 3채널. 8비트 또는 32비트 실수형
- dst: 축적 영상. 입력 영상과 동일 채널 개수.  
32비트 또는 64비트 실수형
- alpha: (입력 영상에 대한) 가중치
- mask: 마스크 영상

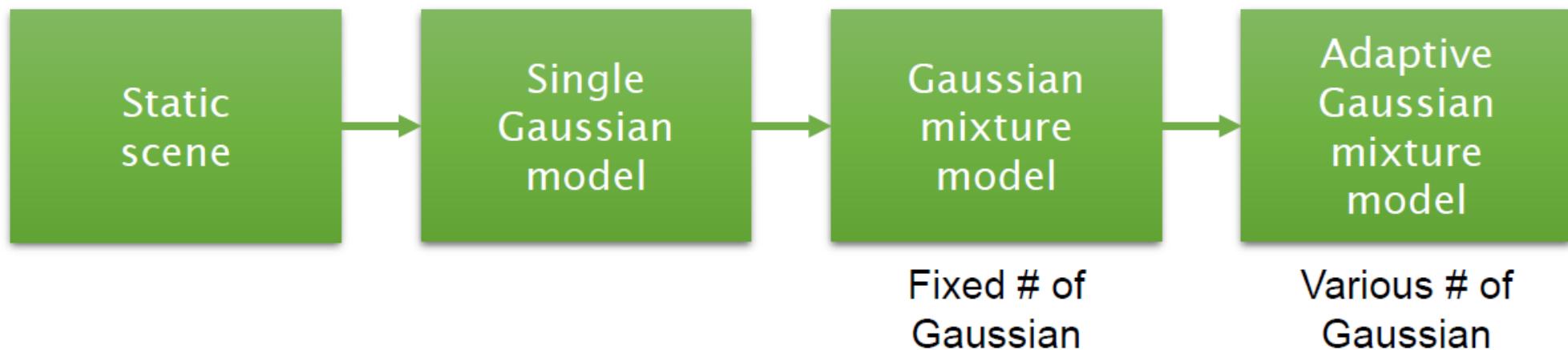
$$dst(x, y) = (1 - \alpha) \cdot dst(x, y) + \alpha \cdot src(x, y) \quad \text{if } mask(x, y) \neq 0$$

```
while True:  
    ret, frame = cap.read()  
  
    if not ret:  
        break  
  
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
    gray = cv2.GaussianBlur(gray, (0, 0), 1.0)  
  
    cv2.accumulateWeighted(gray, fback, 0.01)  
    back = fback.astype(np.uint8)  
  
    diff = cv2.absdiff(gray, back)  
    _, diff = cv2.threshold(diff, 30, 255, cv2.THRESH_BINARY)  
  
    cnt, _, stats, _ = cv2.connectedComponentsWithStats(diff)  
  
    for i in range(1, cnt):  
        x, y, w, h, s = stats[i]  
  
        if s < 100:  
            continue  
  
        cv2.rectangle(frame, (x, y, w, h), (0, 0, 255), 2)
```

# MOG background model

What is MOG?

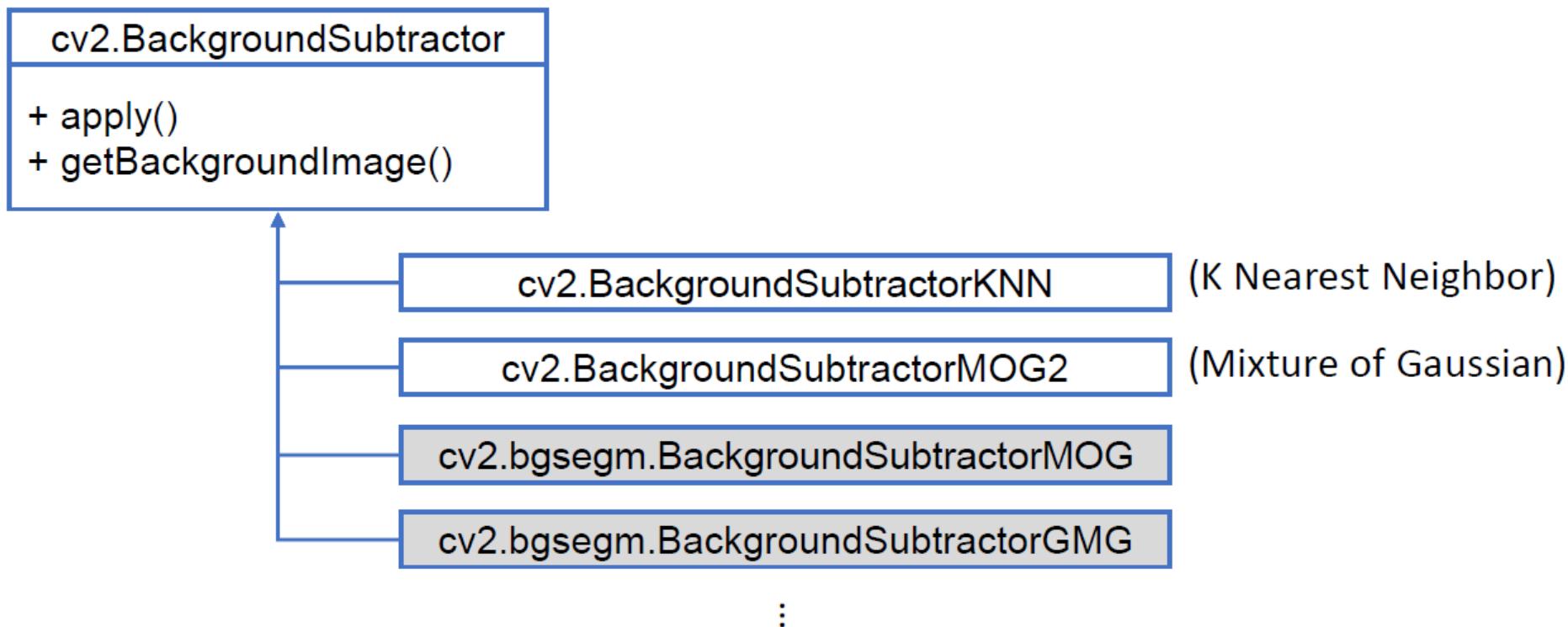
- Mixture of Gaussian, Gaussian Mixture Model (GMM)
- 각 픽셀에 대해 MOG 확률 모델을 설정하여 배경과 전경을 구분



# MOG background model

움직이는 전경 객체 마스크 영상

- OpenCV에서 background subtractor 제공



# MOG background model

MOG 예제

- 5\_10\_BS\_MOG.py

```
bs = cv2.createBackgroundSubtractorMOG2()

while True:
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    fgmask = bs.apply(gray)
    back = bs.getBackgroundImage()

    cnt, _, stats, _ = cv2.connectedComponentsWithStats(fgmask)

    for i in range(1, cnt):
        x, y, w, h, s = stats[i]

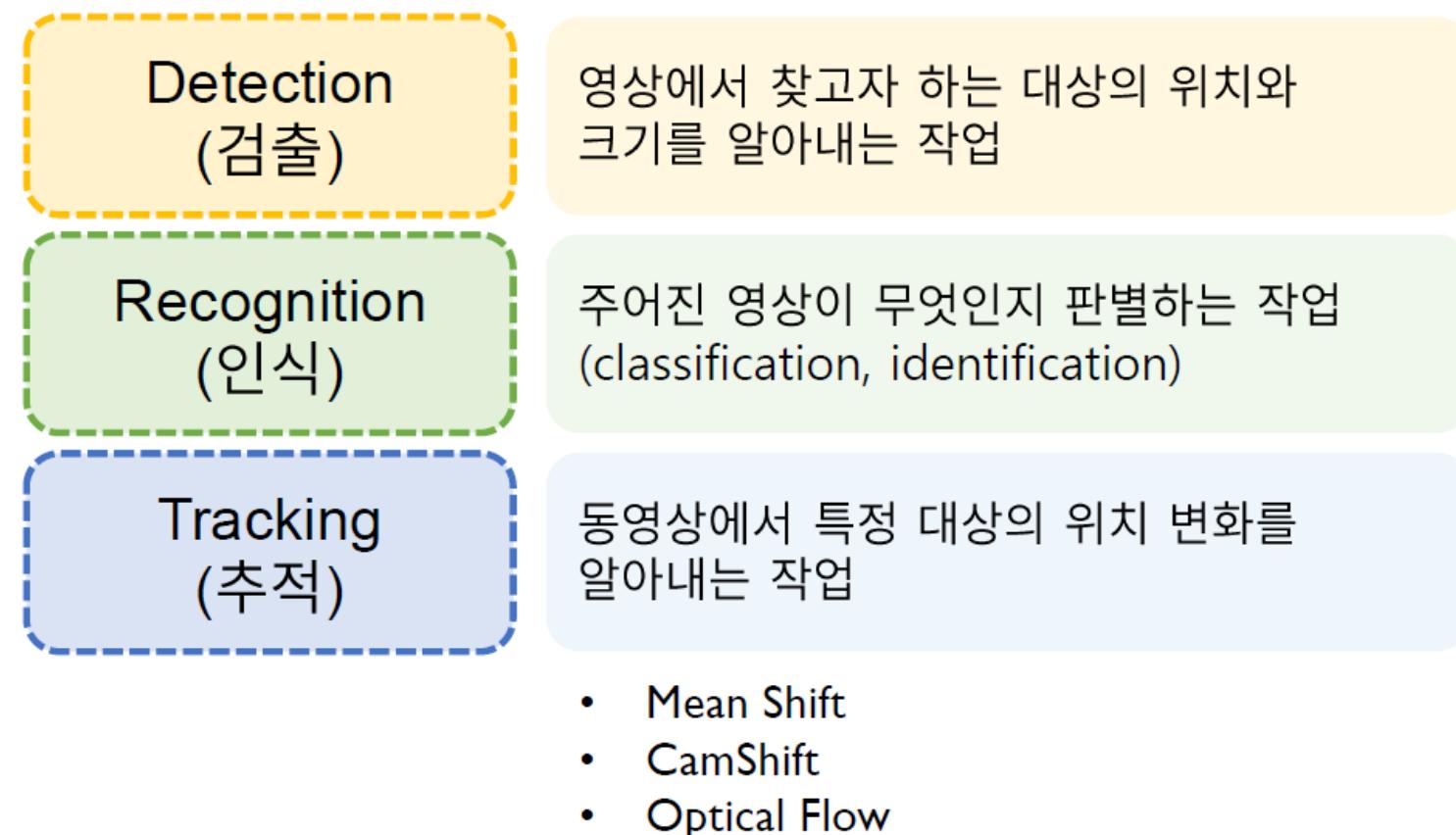
        if s < 80:
            continue

        cv2.rectangle(frame, (x, y, w, h), (0, 0, 255), 2)
```



# Tracking

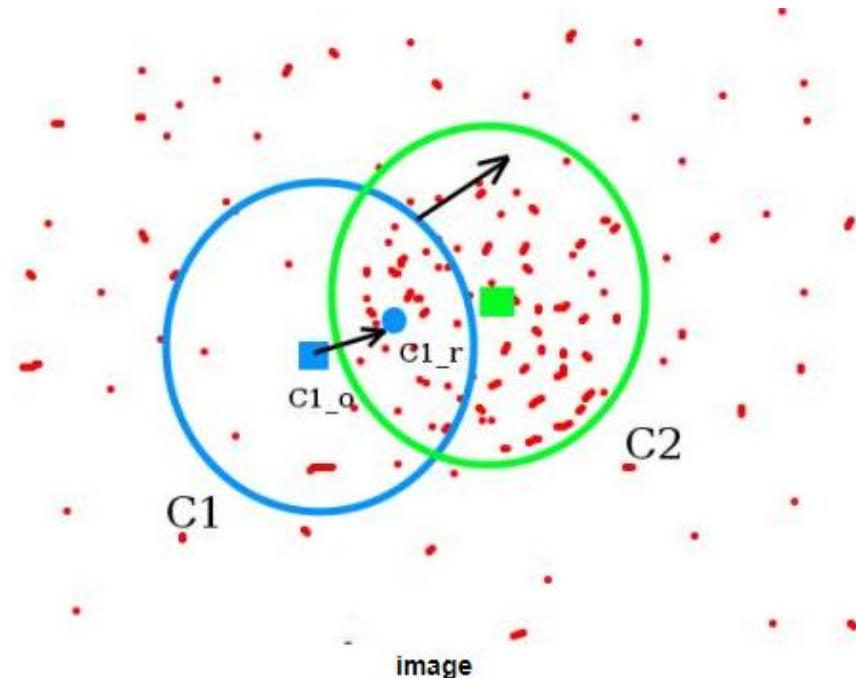
## Detection vs. Recognition vs. Tracking



# Mean shift algorithm

## Idea

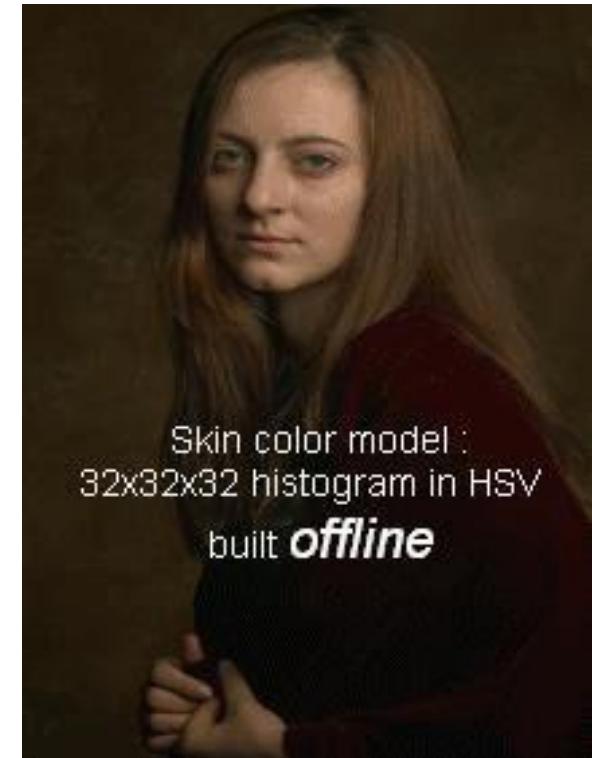
- 불빛 없는 야간에 산을 올라가야 할 때
  - 보이는 한도 내에서 가장 높은 쪽으로 이동 → 반복 (Hill Climb 탐색 기법)
- Mean shift → 어떤 데이터 분포의 peak 또는 무게중심을 찾는 방법
  - 현재 자신의 주변에서 가장 데이터가 밀집된 방향으로 이동



# Mean shift algorithm

평균 이동 알고리즘

- A non-parametric feature-space analysis technique for locating the maxima of a density function
- 국지적 평균을 탐색하면서 이동
- 모드 검출 (mode seeking) 알고리즘



# Mean shift algorithm

평균 이동 알고리즘

- 평균 이동 알고리즘을 이용한 관심 영역 추적
- 1) 추적 객체 인식
  - 첫 번째 프레임에서 추적할 객체의 HSV 색 공간에서 HS 히스토그램 계산
- 2) 평균 이동 추적
  - 매 프레임마다 히스토그램 역투영 수행
  - → 여기에 평균 이동 알고리즘 적용하여 객체 추적

# Mean shift algorithm

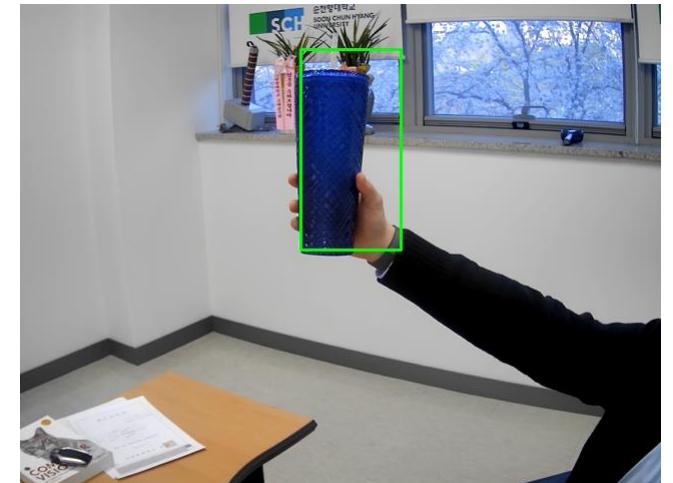
평균 이동 객체 추적 예제

- 히스토그램 역투영 & 평균 이동 추적
- [https://docs.opencv.org/4.x/d7/d00/tutorial\\_meanshift.html](https://docs.opencv.org/4.x/d7/d00/tutorial_meanshift.html)
- 5\_11\_MeanShift.py

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
dst = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)

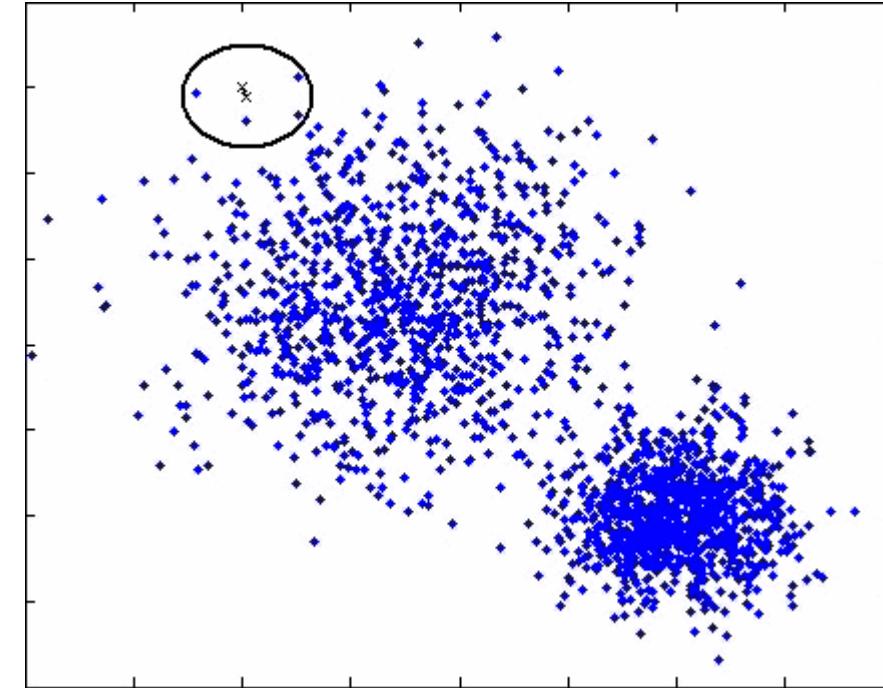
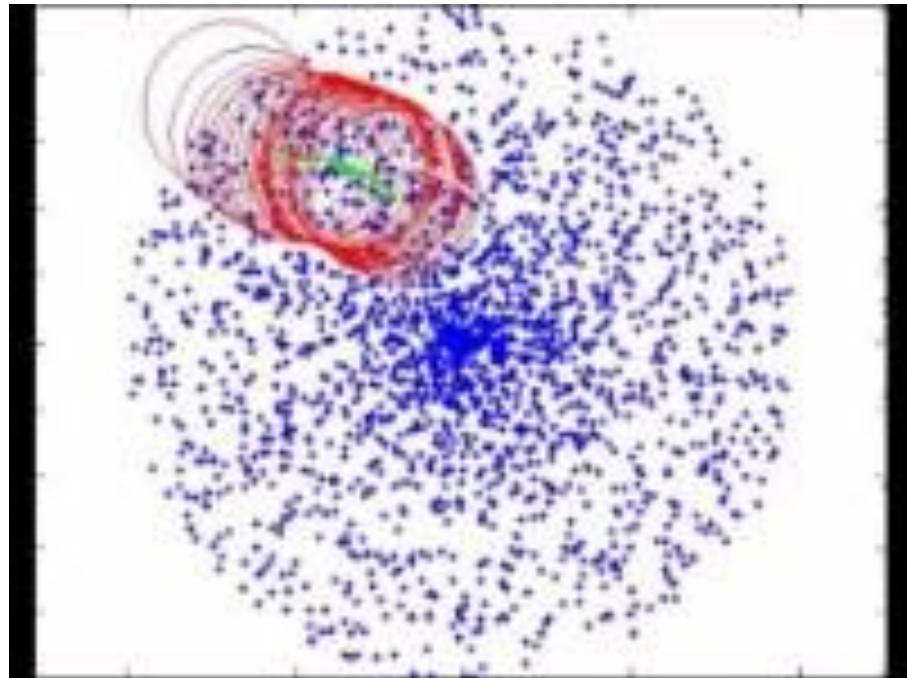
# Apply meanshift to get the new location
ret, track_window = cv2.meanShift(dst, track_window, term_crit)

# Draw it on image
x, y, w, h = track_window
img = cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
```



# Mean shift algorithm

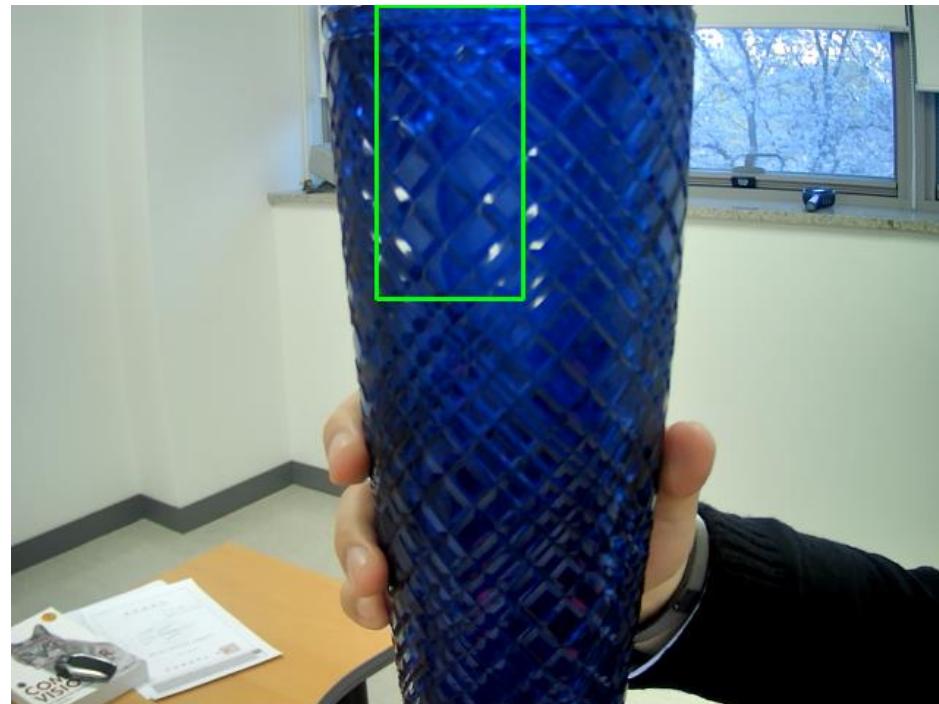
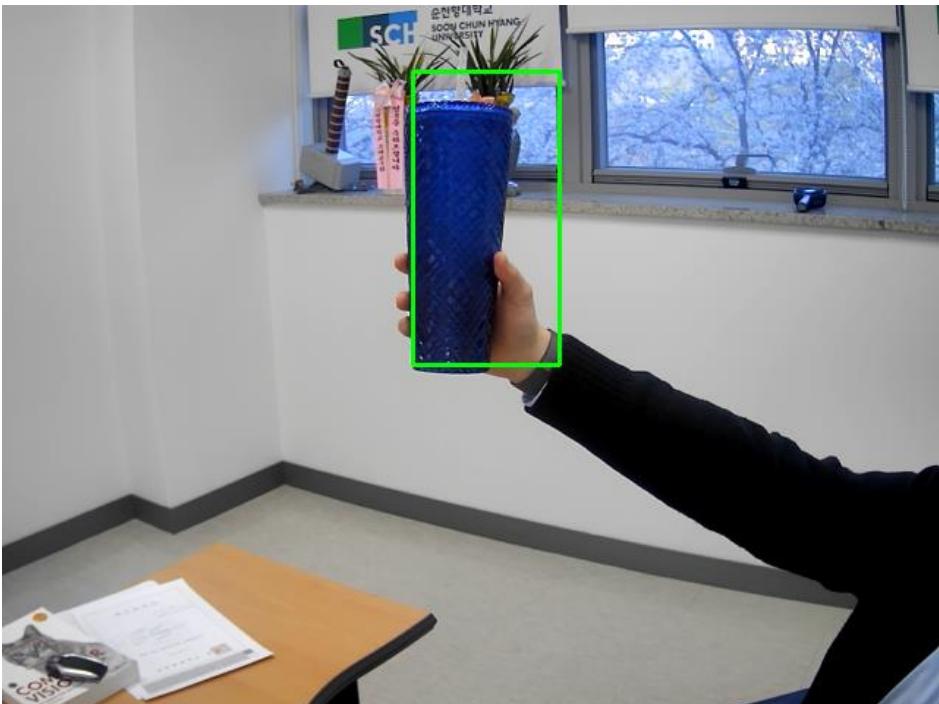
평균 이동 알고리즘의 한계



# Mean shift algorithm

평균 이동 알고리즘의 한계

- Window 사이즈의 변화 X



# CAM shift algorithm

What CAM shift?

- Continuously Adaptive Mean Shift
- 추적하는 객체의 크기가 변하더라도 검색 윈도우의 크기가 고정되어 있는 평균 이동 알고리즘의 단점을 보완

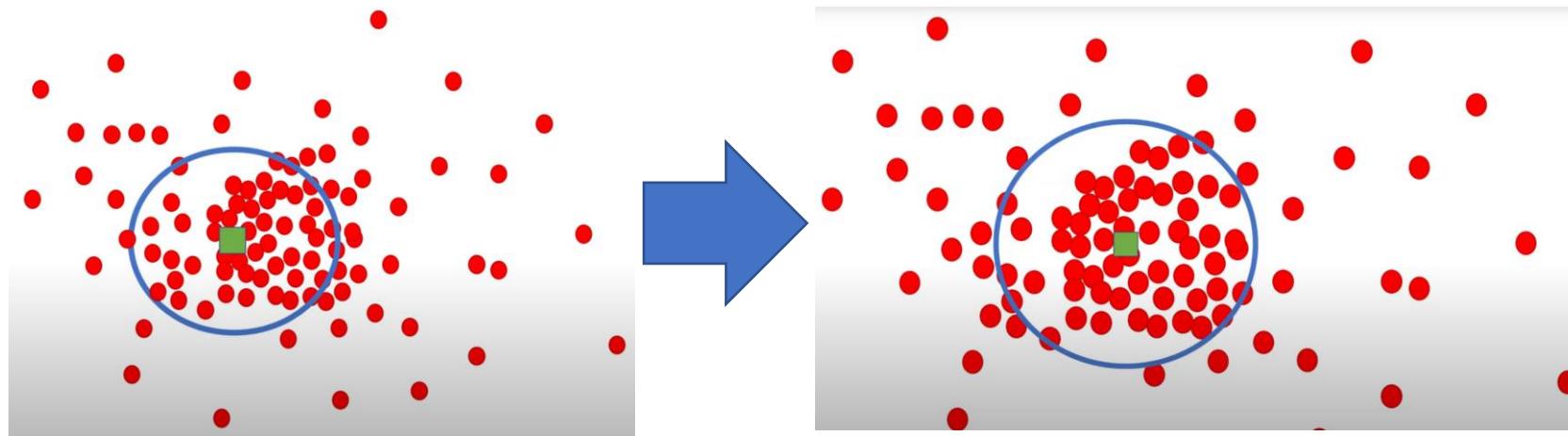
동작 방법

- 평균 이동 알고리즘으로 이동 위치 계산
- Window 크기를 조정
- 특징 공간을 가장 잘 표현하는 타원 검출
- 새로운 크기의 window를 이용하여 다시 평균 이동 수행

# CAM shift algorithm

CAM shift를 이용한 관심 영역 추정

- 관심 영역의 물체 크기가 변화하면 window 크기가 함께 변화함



# CAM shift algorithm

CAM shift를 이용한 관심 영역 추정 예시

- 5\_12\_CAMShift.py

```
# Convert the frame to the HSV color space
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

# Perform back projection of the histogram onto the frame
dst = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)

# Apply the CAMShift algorithm to find the new position of the ROI
ret, track_window = cv2.CamShift(dst, track_window, term_crit)

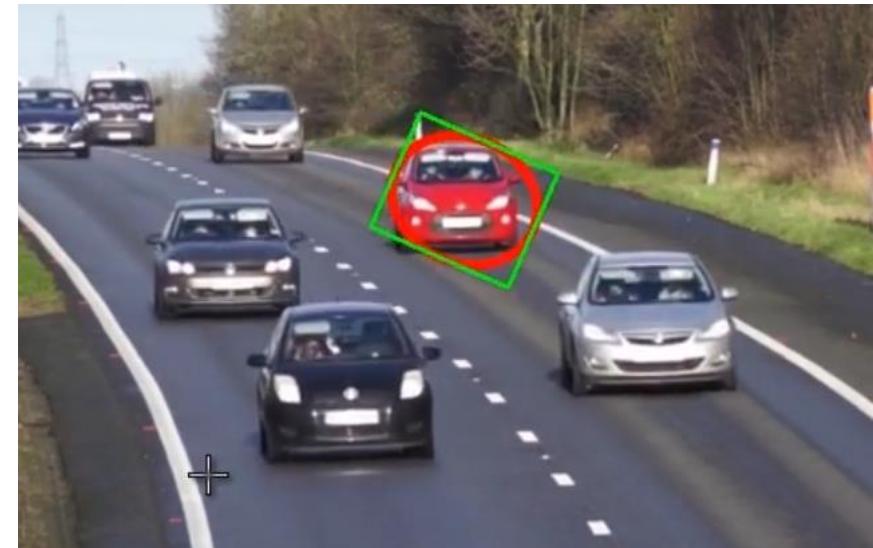
# Draw a rotated rectangle around the new position of the ROI
pts = cv2.boxPoints(ret)
pts = pts.astype(np.intp)

img2 = cv2.polylines(frame, [pts], True, 255, 2)
```

# Mean shift algorithm

평균 이동 알고리즘의 한계

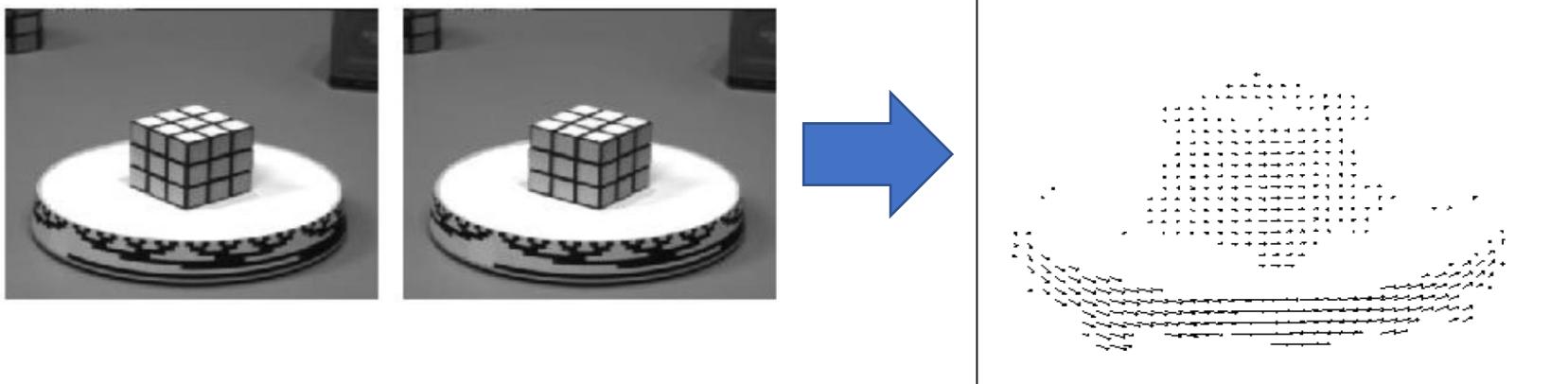
- 복잡한 배경 vs 단순한 배경
  - 배경이 복잡하면 CAM shift 성능이 저하됨 (Mean shift가 더 유리함)
  - CAM shift → 단순 배경인 경우 좋은 성능



# Optical flow

What is optical flow?

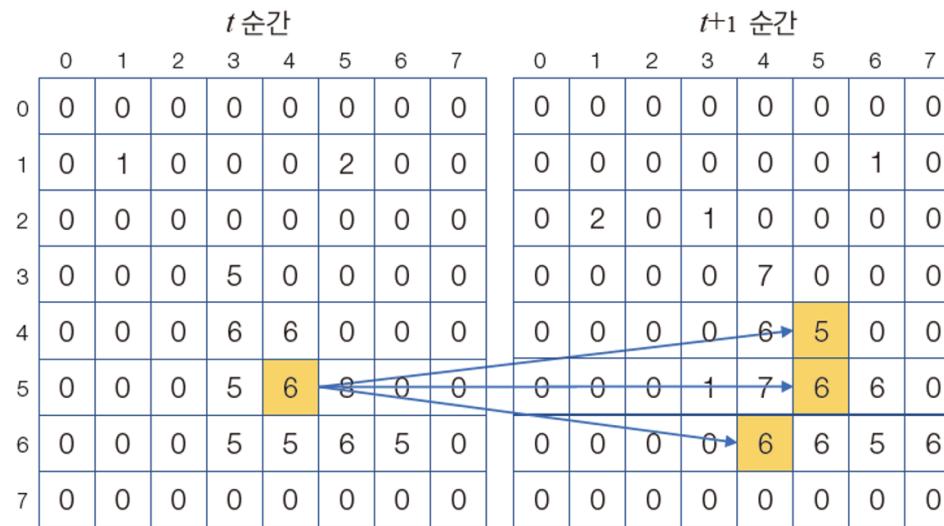
- 움직이는 물체는 명암 변화를 일으킴  
→ 명암 변화를 분석하면 역으로 물체의 motion을 알아 낼 수 있음
- 화소별로 motion vector를 추정해 기록한 맵 = optical flow
- 연속하는 두 프레임(영상)에서 카메라 또는 객체의 움직임에 의해 나타나는 객체의 이동 정보 패턴



# Optical flow

Motion vector 추정의 애매함

- Motion vector: 움직임을 추정하였을 때 이동된 거리를 화소 단위의 벡터량으로 표시한 것



- Optical flow 추정 → 모든 화소의 모션 벡터  $\mathbf{v} = (v, u)$ 를 추정해야 함

# Optical flow

## Optical flow 추정 방법

- 어려움...
  - 수백 × 수백 이상 크기의 256 명암 영상
  - 여러 기하 변환 및 광도 변환, 잡음 발생
  - 움직이는 물체와 정지한 물체가 혼재 + 가려짐 발생
- 현실을 적절이 표현하는 모델 필요
  - 실제 세계를 훼손하지 않는 범위 내에서 적절한 가정 필요
  - Assumption: 밝기 항상성
    - 물체의 같은 점은 다른 영상에서 같은 (유사한) 명암 값을 가진다

# Optical flow

## Optical flow 계산 (추정)

- Taylor 급수에 따르면,

$$f(y + dy, x + dx, t + dt) = f(y, x, t) + \frac{\partial f}{\partial y} dy + \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial t} dt + C$$

- → video에서  $dt \leq 1/30$  로 충분히 작음
- 밝기 항상성 (다음 프레임에서 명암 값이 같음) 적용/대입

$$\frac{\partial f}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial f}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial f}{\partial t} = 0$$

모션 벡터  $\mathbf{v} = (v, u)$ 로 대치

$$\frac{\partial f}{\partial y} v + \frac{\partial f}{\partial x} u + \frac{\partial f}{\partial t} = 0$$

$\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}$  → Edge 검출에 활용한 식

# Optical flow

## Optical flow 계산 (추정)

- Optical flow constraint equation

$$\frac{\partial f}{\partial y} v + \frac{\partial f}{\partial x} u + \frac{\partial f}{\partial t} = 0$$

- Gradient를 구성하는 세 항의 값을 알아도  $v, u$ 를 유일한 값으로 결정 불가능
  - → 부정 방정식

예시

t 순간								t+1 순간							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	6	6	0	0	0	0	0	0	0	0	0	0
5	0	0	0	5	6	0	0	0	0	1	7	6	6	0	0
6	0	0	0	5	5	6	5	0	0	0	6	6	5	6	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$$\frac{\partial f}{\partial y} = f(6, 4, t) - f(5, 4, t) = 5 - 6 = -1$$

$$\frac{\partial f}{\partial x} = f(5, 5, t) - f(5, 4, t) = 8 - 6 = 2$$

$$\frac{\partial f}{\partial t} = f(5, 4, t+1) - f(5, 4, t) = 7 - 6 = 1$$



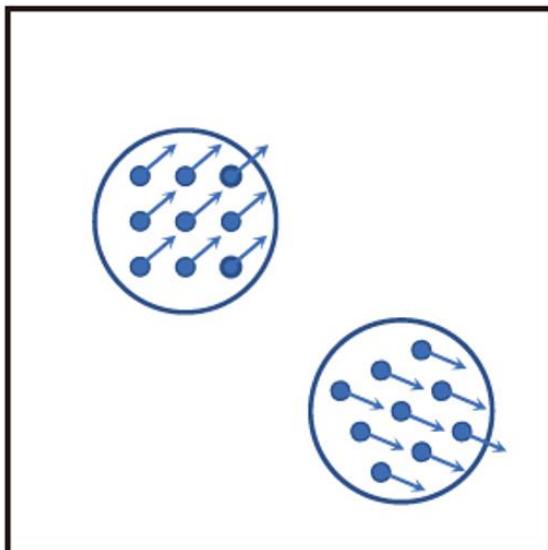
$$-v + 2u + 1 = 0$$

추가 가정 필요

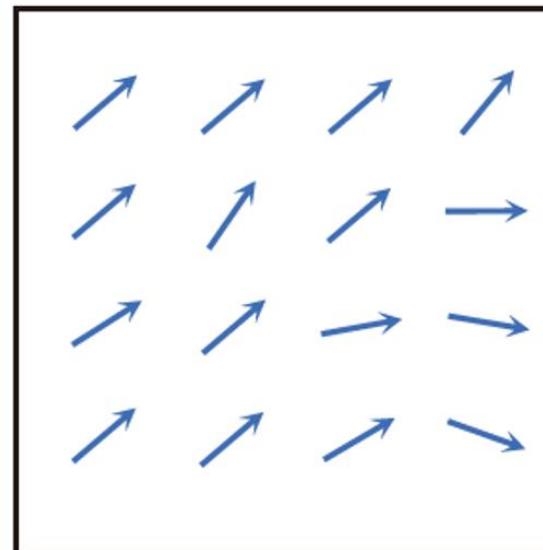
# Optical flow

여러가지 가정들

- Lucas-Kanade optical flow estimation (cv2.calOpticalFlowPyrLK(...))
  - 가정: 이웃 화소는 유사한 모션 벡터를 가진다 → 지역 조건 활용
- Horn-Schunck optical flow estimation
  - 가정: 영상 전체에 걸쳐 모션 벡터는 서서히 변해야 한다 → 광역 조건 활용



(a) Lucas-Kanade 알고리즘의 지역 조건

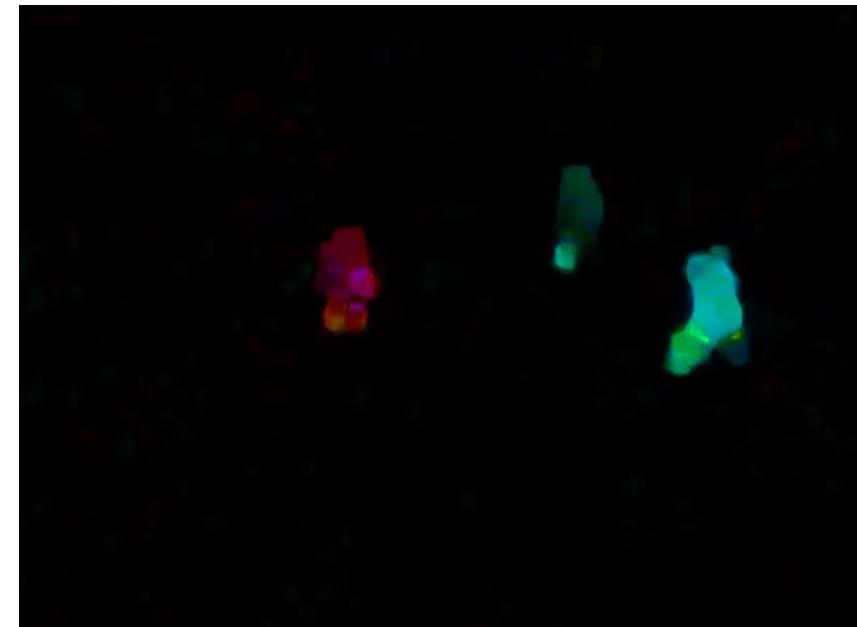


(b) Horn-Schunck 알고리즘의 광역 조건

# Optical flow

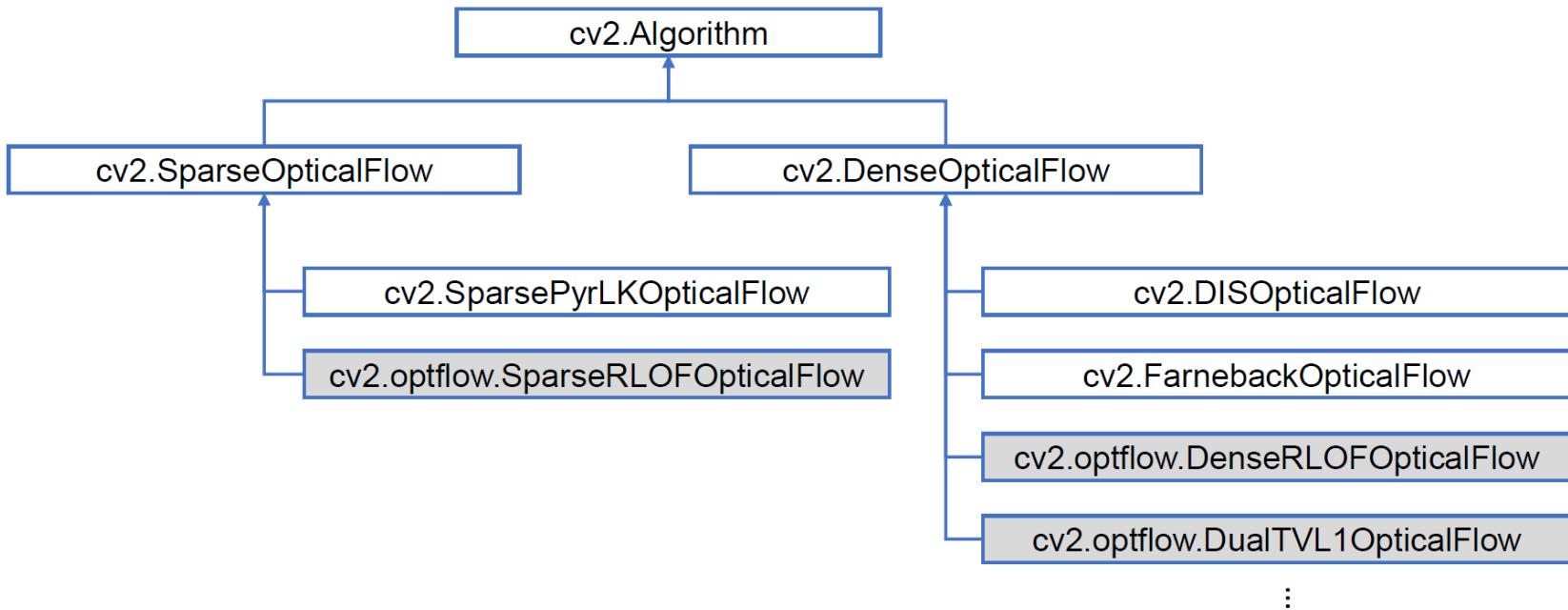
## Optical flow 예제

- 5\_13\_OpticalFlow.py
- 본 예제는 Farneback optical flow 추정 알고리즘을 활용함 (밀집도 기반)
  - Lucas-Kanade와 Horn-Schunck 알고리즘을 합성하여 활용
  - cv2.calOpticalFlowFarneback(...)



# Optical flow

참고: optical flow class in OpenCV



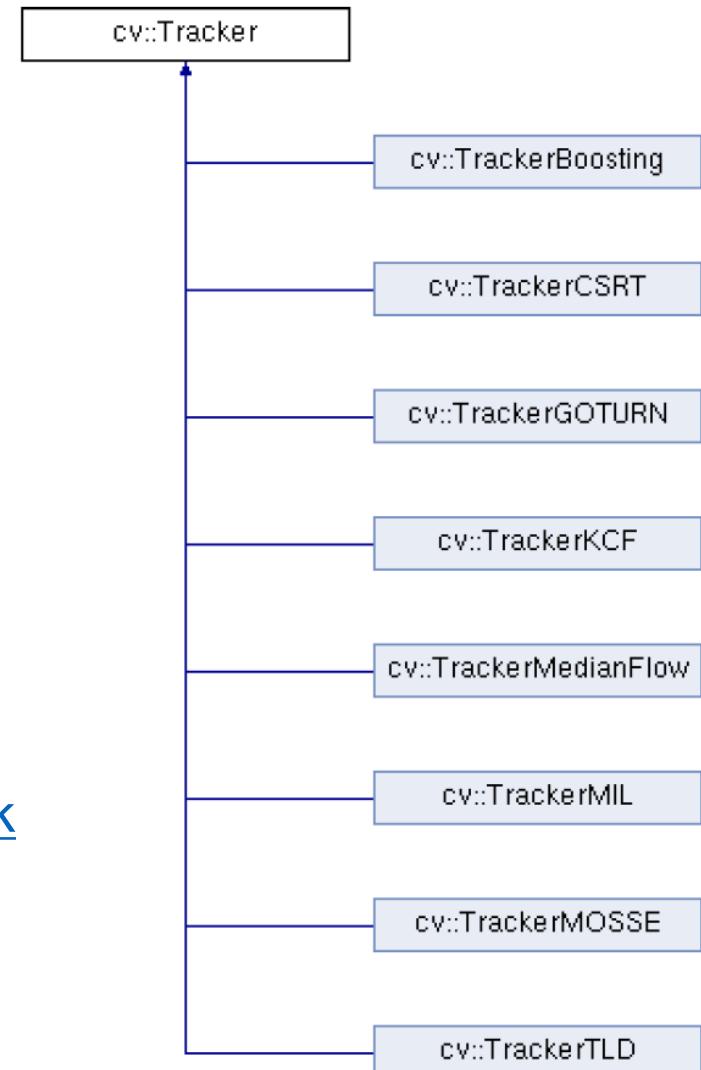
# OpenCV Tracker

## OpenCV Tracker

- OpenCV 3.0 버전부터 tracker 클래스 제공
- opencv-contrib-python 패키지 설치 후 활용
- OpenCV 4.1 기준으로 8가지 트래킹 알고리즘 지원

참고 사이트:

- <https://www.learnopencv.com/object-tracking-using-opencv-cpp-python>
- [https://docs.opencv.org/master/d0/d0a/classcv\\_1\\_1Tracker.html](https://docs.opencv.org/master/d0/d0a/classcv_1_1Tracker.html)



# OpenCV Tracker

## Tracker 클래스 사용 방법

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



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



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



XXX = Boosting, CSRT, GOTURN,  
KCF, MedianFlow, MIL,  
MOSSE, TLD

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

retval: 추적에 성공하면 True,  
실패하면 False.

# OpenCV Tracker

## Tracker 클래스 사용 예제

- 5\_14\_Tracker.py

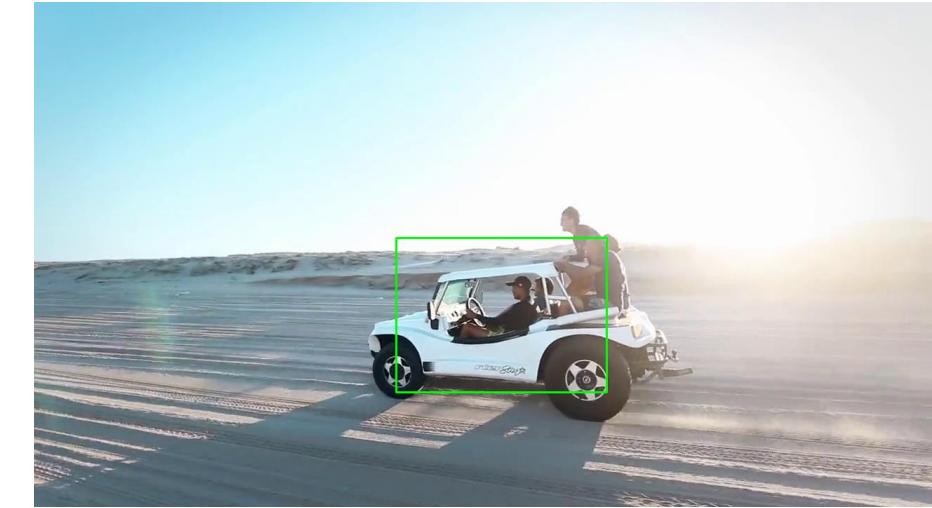
```
# Create the KCF tracker
tracker = cv2.TrackerKCF_create()

# Initialize the tracker with the first frame and the bounding box
ok = tracker.init(frame, bbox)

while True:
    ok, frame = video.read()
    if not ok:
        break

    # Update the tracker
    ok, bbox = tracker.update(frame)

    # Draw the bounding box if the tracking was successful
    if ok:
        (x, y, w, h) = [int(v) for v in bbox]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2, 1)
    else:
        cv2.putText(frame, "Tracking failed", (100, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 255), 2)
```



End of slide