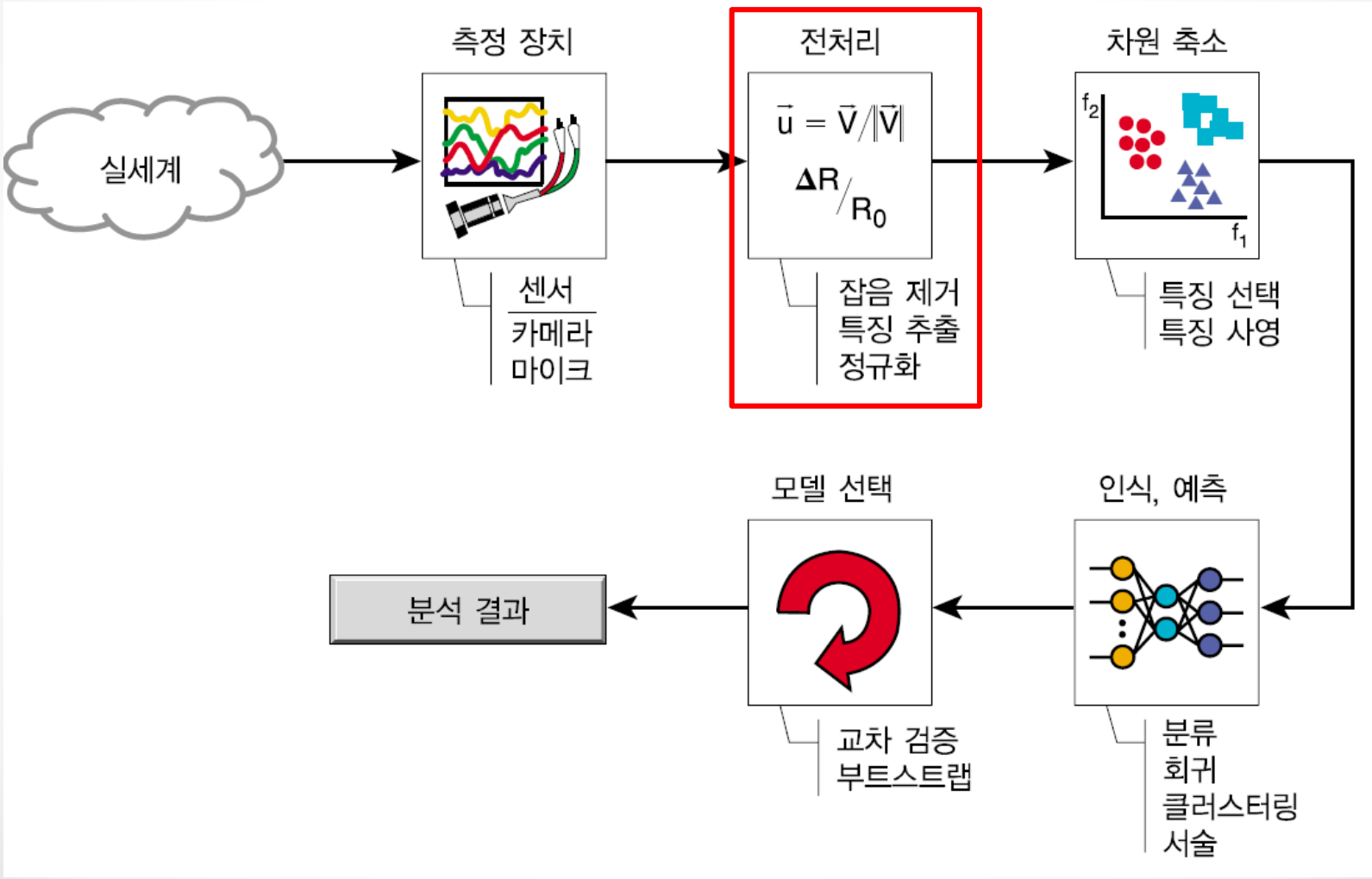
Python

Application3

Application



Application

- 템플릿 매칭 방법
  - 제곱차 매칭

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

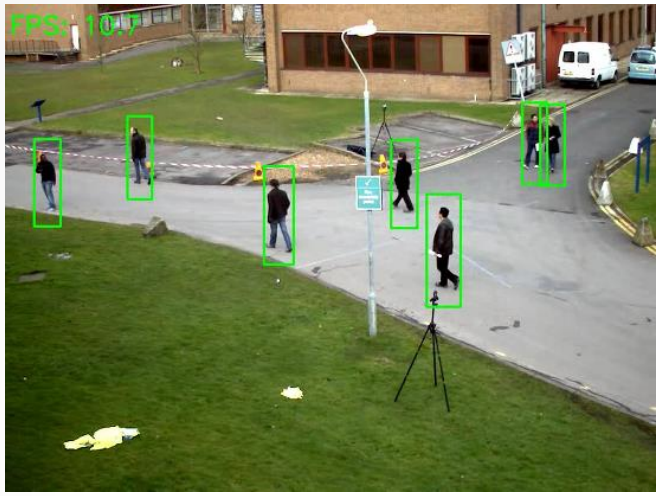
  - 상관관계 매칭

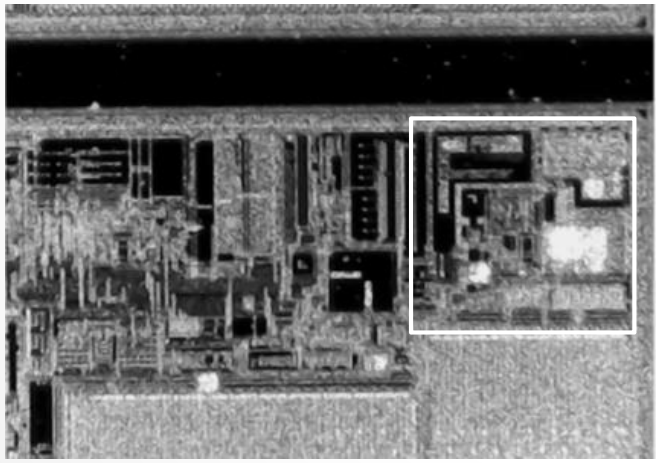
$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

  - 상관계수 매칭

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$
$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

Application





Application

Application

- 원본 이미지

- 템플릿 이미지

Application

- Exercise 1
  - 템플릿 매칭 방법을 이용하여 위치를 검출해보자!!
    1. 슬라이딩 윈도우 방식으로 이미지의 모든 영역을 탐색
    2. 탐색 대상 윈도우 내에서 템플릿 매칭 방법을 이용하여 계산
    3. 가장 상관 관계가 높은 윈도우 위치를 선택

Application

```
import numpy as np
import cv2
import copy

srcImage = cv2.imread("lenna.png", 0)  # 원본 이미지 로드
Template = cv2.imread( " template.PNG " , 0)  # 템플릿 이미지 로드

Rows_srcImage, cols_srcImage = srcImage.shape[:2] # 원본 이미지 크기
Rows_template, cols_template = template.shape[:2] # 템플릿 이미지 크기
```

Application

```
for i in range(rows_srcImage - rows_template):
    for j in range(cols_srcImage - cols_template):

        new_image = copy.copy(srcImage)
        rect = (j, i, rows_template, cols_template) # 템플릿 이미지 크기의 Rect 생성

        comp_data = srcImage[i:i+rows_template, j:j+cols_template] # 템플릿과 비교 영역

        """
        # 템플릿 매칭 및 비교 프로세스 구현
        """
        cv2.rectangle(new_image, rect, (0, 255, 0), 3) # Rect 그리기
        cv2.imshow('srcImage', new_image) # 이미지 출력
        cv2.waitKey(1)

"""
# 템플릿 매칭 결과 Rect 그리기
"""
cv2.imshow('srcImage', srcImage)
cv2.waitKey(0)
```