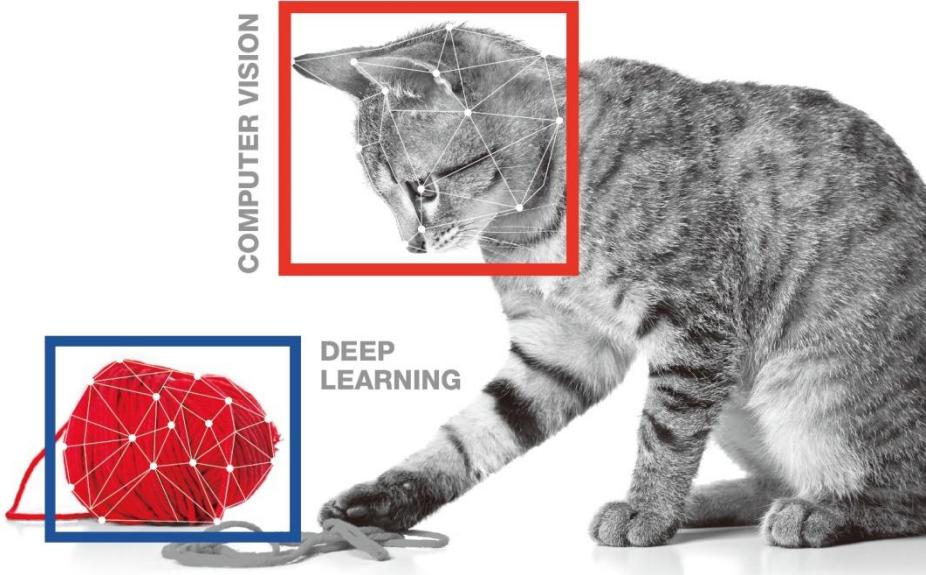


컴퓨터 비전과 딥러닝

[강의교안 이용 안내]

- 본 강의교안의 저작권은 한빛아카데미(주)에 있습니다.
- 이 자료를 무단으로 전제하거나 배포할 경우 저작권법 136조에 의거하여 처벌을 받을 수 있습니다.

COMPUTER VISION



컴퓨터 비전과 딥러닝

Chapter 05 지역 특징

차례

- 5.1 발상
- 5.2 이동과 회전 불변한 지역 특징
- 5.3 스케일 불변한 지역 특징
- 5.4 SIFT
- 5.5 매칭
- 5.6 호모그래피 추정

Preview

■ 대응점 문제 correspondence problem

- 이웃한 영상에 나타난 같은 물체의 같은 곳을 쌍으로 맺는 일
- 파노라마, 물체 인식, 물체 추적, 스테레오 비전, 카메라 갤리브레이션 등에 필수
 - 예) 파노라마 영상 제작에서 봉합 점을 결정하는데 필수



그림 5-1 지역 특징으로 대응점 문제를 해결해서 제작한 파노라마 영상

■ 에지 특징이나 영역 특징은 여러모로 부족

■ 2000년대 초에 지역 특징으로 해결책 모색

- 여러 실용 시스템 등장하여 컴퓨터 비전을 혁신함

5.1 발상

■ 물체 추적에서 대응점 찾기

- 예) 다중 물체 추적 챌린지 MOT-17-14-SDP 동영상 데이터셋 (<https://motchallenge.net/data/MOT17>)
- 반복성 *repeatability*이 뛰어난 특징 필요



[그림 5-2] 대응점 찾기(MOT-17-14-SDP 동영상의 70번째와 83번째 영상)

■ 1980년대에는 에지 경계선에서 모퉁이 *corner* 찾는 연구 왕성

- 특징점이 물체의 실제 모퉁이에 해당해야 한다는 생각이 지배적
- 1990년대 시들하다가 2000년대 초에 자취를 감춤. 지역 특징이라는 대안이 떠오름

5.1 발상

■ 지역 특징의 발상

- 좁은 지역을 보고 특징점 여부 결정 ([그림 5-2] 녹색 박스)
- 물체의 실제 모퉁이에 위치해야 한다는 완고한 생각을 버림
- 반복성을 더 중요하게 취급하는 발상의 전환

■ 지역 특징의 표현

- (위치, 스케일, 방향, 특징 기술자)로 표현
- 위치와 스케일은 검출 단계에서 알아냄 (5.2~5.4.1절)
- 방향과 특징 기술자는 기술 단계에서 알아냄 (5.4.2절)

5.1 발상

■ 지역 특징의 조건

- 반복성 repeatability
- 불변성 invariance
- 분별력 discriminating power
- 지역성 locality
- 적당한 양
- 계산 효율

■ 이들 조건은 상충 관계

- 응용에 따라 적절히 조절할 필요

5.2 이동과 회전 불변한 지역 특징

■ 간단한 인지 실험

- 왼쪽 영상의 a, b, c 중에 어느 것이 오른쪽 영상에서 찾기 쉬울까?
- a가 가장 쉽고 c가 가장 어려움. 왜?

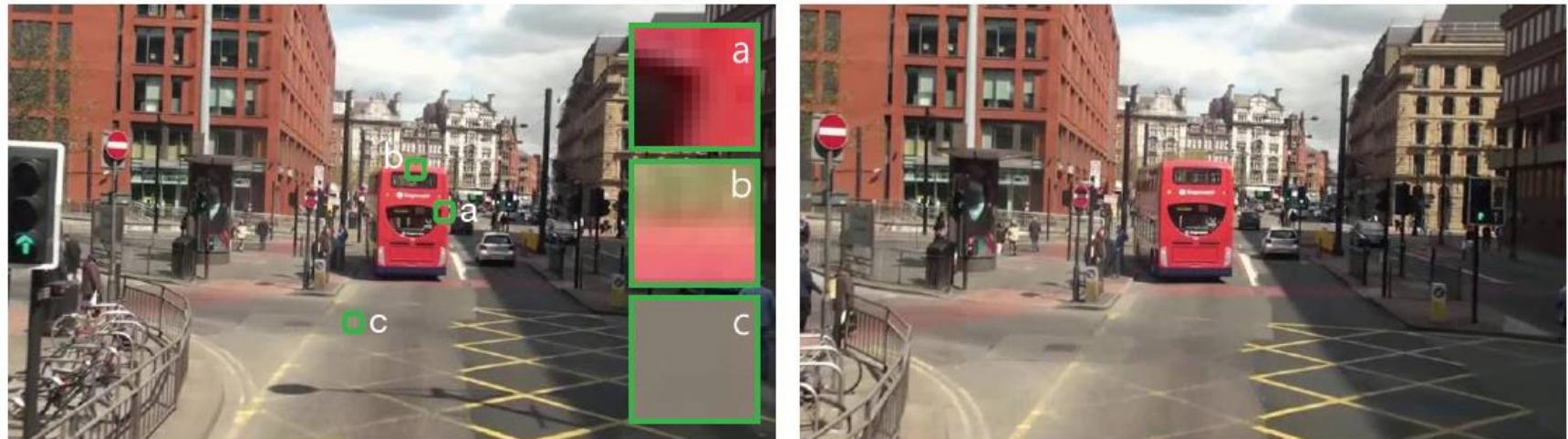


그림 5-3 대응점 찾기 인지 실험(MOT-17-14-SDP 동영상의 70번째와 83번째 영상)

5.2.1 모라벡 알고리즘

■ 모라벡의 설명

- a 는 여러 방향으로 색상 변화가 있어 찾기 쉬운데 c 는 어느 방향으로도 미세한 변화만 있어 찾기 어려움
- 여러 방향에 대해 색상 변화를 측정하는 제곱차의 합이라는 식 (5.1) 제안

$$S(v, u) = \sum_y \sum_x (f(y+v, x+u) - f(y, x))^2 \quad (5.1)$$

- 예) 화소 위치 (y, x) 가 $(4, 3)$ 인 경우

$$S(v, u) = \sum_{3 \leq y \leq 5} \sum_{2 \leq x \leq 4} (f(y+v, x+u) - f(y, x))^2$$

5.2.1 모라벡 알고리즘

[예시 5-1] 제곱차의 합 계산

[그림 5-4(a)]에서 b로 표시된 점 (4,3)에 계산을 적용한다고 가정한다.

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0
3	0	0	0	1	1	0	0	0	0	0
4	0	0	0	b ₁	1	1	0	0	0	0
5	0	0	0	1	1	1	1	0	0	0
6	0	0	0	1	1	1	1	a ₁	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

(a) 원래 영상

	-1	0	1		-1	0	1		-1	0	1
-1	3	4	4	v 0	3	1	6	v 0	0	0	0
v 0	2	0	2	v 0	3	0	4	v 0	0	0	0
1	4	3	2	1	3	0	3	1	0	0	0

(b) 세 점에서 추출한 S 맵

그림 5-4 제곱차의 합 계산

아래 식은 v 는 0, u 는 1인 경우인 $S(0,1)$ 을 계산하는 과정이다.

$$S(0,1) = \sum_{3 \leq y \leq 5} \sum_{2 \leq x \leq 4} (f(y, x+1) - f(y, x))^2 = 4$$

(v,u) 의 나머지 칸을 계산해 3×3 맵에 채우면 점 b의 맵은 [그림 5-4(b)]의 가운데가 된다.

5.2.1 모라벡 알고리즘

■ 식 (5.1)의 의도

- c는 모든 방향에서 변화가 없어 S 맵의 모든 요소가 0. 지역 특징으로 자격이 없음
- b는 수평 방향만 변화가 있어 S 맵의 좌우 이웃만 큰 값. 지역 특징으로 부족
- a는 모든 방향으로 변화가 있어 S 맵에서 8이웃 모두 큰 값. 지역 특징으로 훌륭함

$$C = \min(S(0,1), S(0,-1), S(1,0), S(-1,0)) \quad (5.2)$$

■ 특징 가능성 값 측정

- S 맵에서 상하좌우 네 이웃 화소의 최소값

5.2.2 해리스 특징점

■ 모라벡은 지역 특징의 길을 열었지만 현실적이지 않음

- 실제 세계의 영상은 식 (5.2)의 상하좌우 이웃만 보는 것으로 부족
- 해리스의 확장

5.2.2 해리스 특징점

■ 2차 모멘트 행렬을 통한 특징점 검출

- 잡음에 대처하기 위해 가우시안 적용하여 가중치 제곱차의 합을 식 (5.3)으로 정의

$$S(v, u) = \sum_y \sum_x G(y, x) (f(y+v, x+u) - f(y, x))^2 \quad (5.3)$$

- 테일러 확장으로 식 (5.4) 성립

$$f(y+v, x+u) \approx f(y, x) + vd_y(y, x) + ud_x(y, x) \quad (5.4)$$

- 식 (5.4)를 식 (5.3)에 대입하면 식 (5.5)

$$S(v, u) \approx \sum_y \sum_x G(y, x) (vd_y(y, x) + ud_x(y, x))^2 \quad (5.5)$$

5.2.2 해리스 특징점

- 식 (5.5)를 정리하면

$$\begin{aligned} S(v, u) &\cong \sum_y \sum_x G(y, x) (vd_y + ud_x)^2 = \sum_y \sum_x G(y, x) (v^2 d_y^2 + 2vud_y d_x + u^2 d_x^2) \\ &= \sum_y \sum_x G(y, x) (v - u) \begin{pmatrix} d_y^2 & d_y d_x \\ d_y d_x & d_x^2 \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix} \\ &= (v - u) \sum_y \sum_x G(y, x) \begin{pmatrix} d_y^2 & d_y d_x \\ d_y d_x & d_x^2 \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix} \\ &= (v - u) \begin{pmatrix} \sum_y \sum_x G(y, x) d_y^2 & \sum_y \sum_x G(y, x) d_y d_x \\ \sum_y \sum_x G(y, x) d_y d_x & \sum_y \sum_x G(y, x) d_x^2 \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix} \end{aligned}$$

- 컨볼루션 연산자 \circledast 를 이용하여 쓰면

$$S(v, u) \cong (v - u) \begin{pmatrix} G \circledast d_y^2 & G \circledast d_y d_x \\ G \circledast d_y d_x & G \circledast d_x^2 \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix} = \mathbf{u} \mathbf{A} \mathbf{u}^T \quad (5.6)$$

- 2차 모멘트 행렬 \mathbf{A}

$$\mathbf{A} = \begin{pmatrix} G \circledast d_y^2 & G \circledast d_y d_x \\ G \circledast d_y d_x & G \circledast d_x^2 \end{pmatrix} \quad (5.7)$$

5.2.2 해리스 특징점

■ 2차 모멘트 행렬 A의 특성

- (v, u) 는 실수 가능
- A는 화소 주위의 영상 구조를 표현하기 때문에 A만 분석하면 지역 특징 여부 알 수 있음((v, u) 를 변화시키면서 맵을 생성할 필요 없음)

■ 해리스의 특징 가능성 계산

- A의 고유값 λ_1 과 λ_2

$$C = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \quad (5.8)$$

- 식 (5.9)를 이용한 빠른 계산

$$C = (pq - r^2) - k(p + q)^2 \quad (5.9)$$

$$\mathbf{A} = \begin{pmatrix} p & r \\ r & q \end{pmatrix}$$

5.2.2 해리스 특징점

표 5-1 [그림 5-4(a)]에서 세 점의 특징 가능성 측정

	a	b	c
2차 모멘트 행렬	$\mathbf{A} = \begin{pmatrix} 0.52 & -0.2 \\ -0.2 & 0.53 \end{pmatrix}$	$\mathbf{A} = \begin{pmatrix} 0.08 & -0.08 \\ -0.08 & 0.8 \end{pmatrix}$	$\mathbf{A} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
고윳값	$\lambda_1=0.72, \lambda_2=0.33$	$\lambda_1=0.81, \lambda_2=0.07$	$\lambda_1=0.0, \lambda_2=0.0$
특징 가능성 값	$C=0.1925$	$C=0.0237$	$C=0.0$

5.2.2 해리스 특징점

■ C 계산 예제와 비최대 억제 적용([프로그램 5-1]의 실행 결과)

```
[[ 0.   0.   -0.   -0.   -0.    0.    0.    0.    0.    0. ]
 [ 0.   -0.   0.02  0.04  0.02  -0.    0.    0.    0.    0. ]
 [ 0.   -0.   0.07  0.19  0.08  -0.02  -0.    0.    0.    0. ]
 [ 0.   -0.   0.03  0.17  0.09  -0.06  -0.02  -0.    0.    0. ]
 [ 0.   -0.   -0.02  0.02  0.05  -0.06  -0.06  -0.02  -0.    0. ]
 [ 0.   -0.   0.02  0.09  0.08  0.05  0.09  0.08  0.02  -0. ]
 [ 0.   -0.   0.07  0.19  0.09  0.02  0.17  0.19  0.04  -0. ]
 [ 0.   -0.   0.03  0.07  0.02  -0.02  0.03  0.07  0.02  -0. ]
 [ 0.   0.   -0.   -0.   -0.    -0.    -0.    -0.    -0.    0. ]
 [ 0.   0.   0.   0.    0.    0.    0.    0.    0.    0. ]]
```

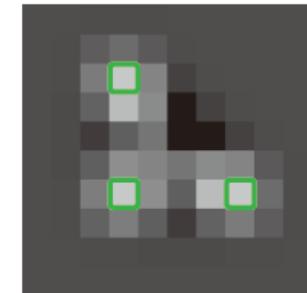


그림 5-5 특징 가능성 맵 C 와 비최대 억제로 찾은 지역 특징점

5.2.2 해리스 특징점

프로그램 5-1

해리스 특징점 검출 구현하기

```
01 import cv2 as cv
02 import numpy as np
03
04 img=np.array([[0,0,0,0,0,0,0,0,0],
05                 [0,0,0,0,0,0,0,0,0],
06                 [0,0,0,1,0,0,0,0,0],
07                 [0,0,0,1,1,0,0,0,0],
08                 [0,0,0,1,1,1,0,0,0],
09                 [0,0,0,1,1,1,1,0,0],
10                 [0,0,0,1,1,1,1,1,0],
11                 [0,0,0,0,0,0,0,0,0],
12                 [0,0,0,0,0,0,0,0,0],
13                 [0,0,0,0,0,0,0,0,0]],dtype=np.float32)
14
15 ux=np.array([[-1,0,1]])
16 uy=np.array([-1,0,1]).transpose()
17 k=cv.getGaussianKernel(3,1)
18 g=np.outer(k,k.transpose())
19
```

5.2.2 해리스 특징점

```
20 dy=cv.filter2D(img,cv.CV_32F,uy)
21 dx=cv.filter2D(img,cv.CV_32F,ux)
22 dyy=dy*dy
23 dxx=dx*dx
24 dyx=dy*dx
25 gdyy=cv.filter2D(dyy,cv.CV_32F,g)
26 gdxx=cv.filter2D(dxx,cv.CV_32F,g)
27 gdyx=cv.filter2D(dydx,cv.CV_32F,g)
28 C=(gdyy*gdxx-gdyx*gdyx)-0.04*(gdyy+gdxx)*(gdyy+gdxx)
29
30 for j in range(1,C.shape[0]-1):           # 비최대 억제
31     for i in range(1,C.shape[1]-1):
32         if C[j,i]>0.1 and sum(sum(C[j,i]>C[j-1:j+2,i-1:i+2]))==8:
33             img[j,i]=9                      # 특징점을 원본 영상에 9로 표시
34
```

5.2.2 해리스 특징점

```
35 np.set_printoptions(precision=2)
36 print(dy) ①
37 print(dx) ②
38 print(dyy) ③
39 print(dxx) ④
40 print(dyx) ⑤
41 print(gdyy) ⑥
42 print(gdxx) ⑦
43 print(gdyx) ⑧
44 print(C) ⑨          # 특징 가능성 맵
45 print(img) ⑩         # 특징점을 9로 표시한 원본 영상
46
47 popping=np.zeros([160,160],np.uint8)      # 화소 확인 가능하게 16배로 확대
48 for j in range(0,160):
49     for i in range(0,160):
50         popping[j,i]=np.uint8((C[j//16,i//16]+0.06)*700)
51
52 cv.imshow('Image Display2',popping) ⑪
53 cv.waitKey()
54 cv.destroyAllWindows()
```

5.2.2 해리스 특징점

①

```
[[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  1.  1.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  1.  1.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  1.  1.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  1.  1.  0.  0. ]
 [ 0.  0.  0. -1. -1. -1.  0.  0.  0.  0. ]
 [ 0.  0.  0. -1. -1. -1. -1.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  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.  0.  0.  0.  0.  0. ]
 [ 0.  0.  1.  0. -1.  0.  0.  0.  0.  0. ]
 [ 0.  0.  1.  1. -1. -1.  0.  0.  0.  0. ]
 [ 0.  0.  1.  1.  0. -1. -1.  0.  0.  0. ]
 [ 0.  0.  1.  1.  0.  0. -1. -1.  0.  0. ]
 [ 0.  0.  1.  1.  0.  0.  0. -1. -1.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]]]
```

③

```
[[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  1.  1.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  1.  1.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  1.  1.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  1.  1.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  1.  1.  1.  0.  0. ]
 [ 0.  0.  0.  1.  1.  1.  1.  0.  0.  0. ]
 [ 0.  0.  0.  1.  1.  1.  1.  1.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]]]
```

④

```
[[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  1.  0.  1.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  1.  1.  1.  1.  0.  0.  0.  0. ]
 [ 0.  0.  1.  1.  0.  1.  1.  0.  0.  0. ]
 [ 0.  0.  1.  1.  0.  1.  1.  1.  0.  0. ]
 [ 0.  0.  1.  1.  0.  0.  1.  1.  0.  0. ]
 [ 0.  0.  1.  1.  0.  0.  0.  1.  1.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]]]
```

⑤

```
[[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0. -1.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0. -1. -1.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0. -1. -1.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0. -1. -1.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0. -1. -1.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -1. -1. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. -1. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]]]
```

⑥

```
[[[ 0.   0.   0.15 0.25 0.15 0.   0.   0.   0.   0. ]
 [ 0.   0.   0.2  0.4  0.32 0.08 0.   0.   0.   0.   0. ]
 [ 0.   0.   0.2  0.53 0.6  0.32 0.08 0.   0.   0.   ]
 [ 0.   0.   0.08 0.32 0.6  0.6  0.32 0.08 0.   0.   ]
 [ 0.   0.   0.   0.08 0.32 0.6  0.6  0.32 0.08 0.   ]
 [ 0.   0.   0.08 0.2  0.35 0.6  0.73 0.48 0.12 0.   ]
 [ 0.   0.   0.2  0.53 0.73 0.8  0.8  0.52 0.15 0.   ]
 [ 0.   0.   0.2  0.53 0.73 0.73 0.65 0.4  0.12 0.   ]
 [ 0.   0.   0.08 0.2  0.27 0.27 0.27 0.2  0.08 0.   ]
 [ 0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]]]
```

⑦

```
[[[ 0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]
 [ 0.   0.08 0.12 0.15 0.12 0.08 0.   0.   0.   0. ]
 [ 0.   0.2  0.4  0.52 0.48 0.32 0.08 0.   0.   0. ]
 [ 0.   0.27 0.65 0.8  0.73 0.6  0.32 0.08 0.   0. ]
 [ 0.   0.27 0.73 0.8  0.6  0.6  0.32 0.08 0.   0. ]
 [ 0.   0.27 0.73 0.73 0.35 0.32 0.6  0.6  0.32 0.15]
 [ 0.   0.2  0.53 0.53 0.2  0.08 0.32 0.53 0.4  0.25]
 [ 0.   0.08 0.2  0.2  0.08 0.   0.08 0.2  0.2  0.15]
 [ 0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]
 [ 0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]]]
```

5.2.2 해리스 특징점

⑧

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0. -0.08 -0.12 -0.08 0.  0.  0.  0. ]
 [ 0.  0.  0. -0.2 -0.4 -0.32 -0.08 0.  0.  0. ]
 [ 0.  0.  0. -0.2 -0.53 -0.6 -0.32 -0.08 0.  0. ]
 [ 0.  0.  0. -0.08 -0.32 -0.6 -0.6 -0.32 -0.08 0. ]
 [ 0.  0. -0.08 -0.12 -0.15 -0.32 -0.53 -0.4 -0.12 0. ]
 [ 0.  0. -0.12 -0.2 -0.12 -0.08 -0.2 -0.2 -0.08 0. ]
 [ 0.  0. -0.08 -0.12 -0.08 0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]]
```

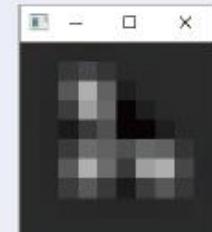
⑨

```
[[ 0.  0. -0. -0. -0.  0.  0.  0.  0.  0. ]
 [ 0. -0.  0.02 0.04 0.02 -0.  0.  0.  0.  0. ]
 [ 0. -0.  0.07 0.19 0.08 -0.02 -0.  0.  0.  0. ]
 [ 0. -0.  0.03 0.17 0.09 -0.06 -0.02 -0.  0.  0. ]
 [ 0. -0. -0.02 0.02 0.05 -0.06 -0.06 -0.02 -0.  0. ]
 [ 0. -0.  0.02 0.09 0.08 0.05 0.09 0.08 0.02 -0. ]
 [ 0. -0.  0.07 0.19 0.09 0.02 0.17 0.19 0.04 -0. ]
 [ 0. -0.  0.03 0.07 0.02 -0.02 0.03 0.07 0.02 -0. ]
 [ 0.  0. -0. -0. -0. -0. -0. -0. -0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]]
```

⑩

```
[[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  9.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  1.  1.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  1.  1.  1.  0.  0.  0.  0. ]
 [ 0.  0.  0.  1.  1.  1.  1.  0.  0.  0. ]
 [ 0.  0.  0.  9.  1.  1.  1.  9.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]]
```

⑪



5.2.2 해리스 특징점

■ 해리스 특징점의 분석

- 물체의 실제 모퉁이 뿐 아니라 블롭에서도 검출
- 이후에는 모퉁이 대신 특징점 feature point 또는 관심점 interest point 으로 부름
- 이동과 회전에 불변
- 스케일에는 불변 아님

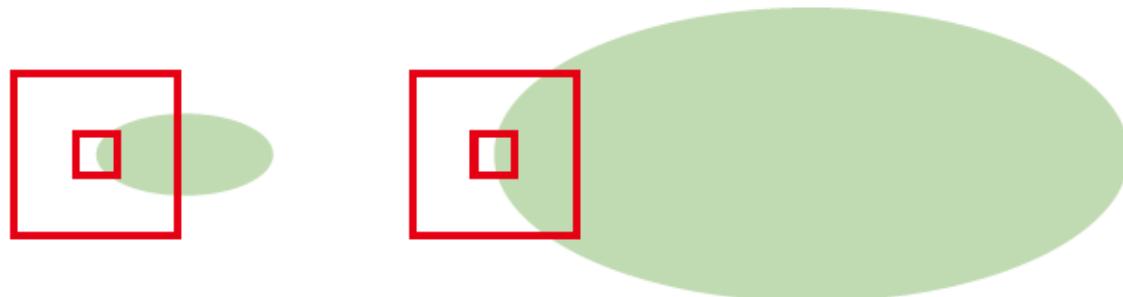


그림 5-6 물체의 크기에 따라 마스크의 크기를 적절하게 정해야 하는 상황

5.3 스케일 불변한 지역 특징

■ 사람은 스케일 불변인 특징 사용

- 거리에 상관없이 같은 물체를 같다고 인식
- 거리에 따라 세세한 내용에만 차이가 있음

■ 스케일 공간 scale space 이론은 컴퓨터 비전에 스케일 불변 가능성 열어줌

[알고리즘 5-1] 스케일 공간에서 특징점 검출

입력: 명암 영상 f

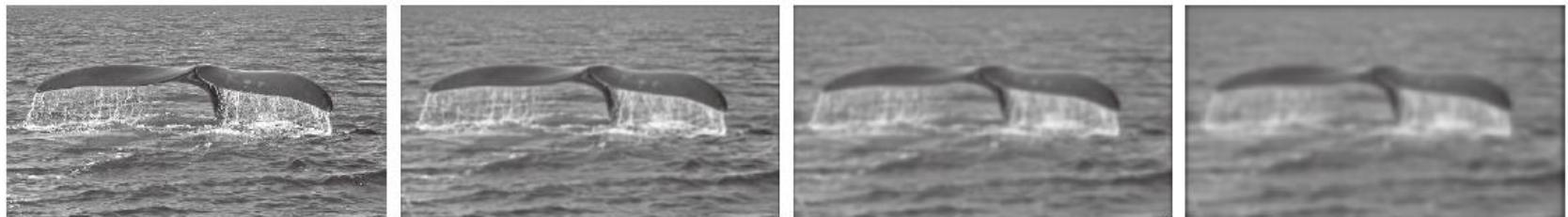
출력: 스케일에 불변한 특징점 집합

1. 입력 영상 f 로부터 다중 스케일 영상 \tilde{f} 를 구성한다.
2. \tilde{f} 에 적절한 미분 연산을 적용하여 다중 스케일 미분 영상 \tilde{f}' 를 구한다.
3. \tilde{f}' 에서 극점을 찾아 특징점으로 취한다.

5.3 스케일 불변한 지역 특징

■ 다중 스케일 영상 구현하는 방법

- 입력은 영상 한 장. 여러 거리에서 보았을 때 나타나는 현상을 흉내 내는 수밖에 없음
- 가우시안 스무딩과 피라미드 방법



(a) 가우시안 스무딩 방법



(b) 피라미드 방법

그림 5-7 다중 스케일 영상을 구성하는 두 가지 방법[오일석2014]

5.3 스케일 불변한 지역 특징

- 가우시안 스무딩은 스케일을 연속값으로 조절할 수 있는 장점

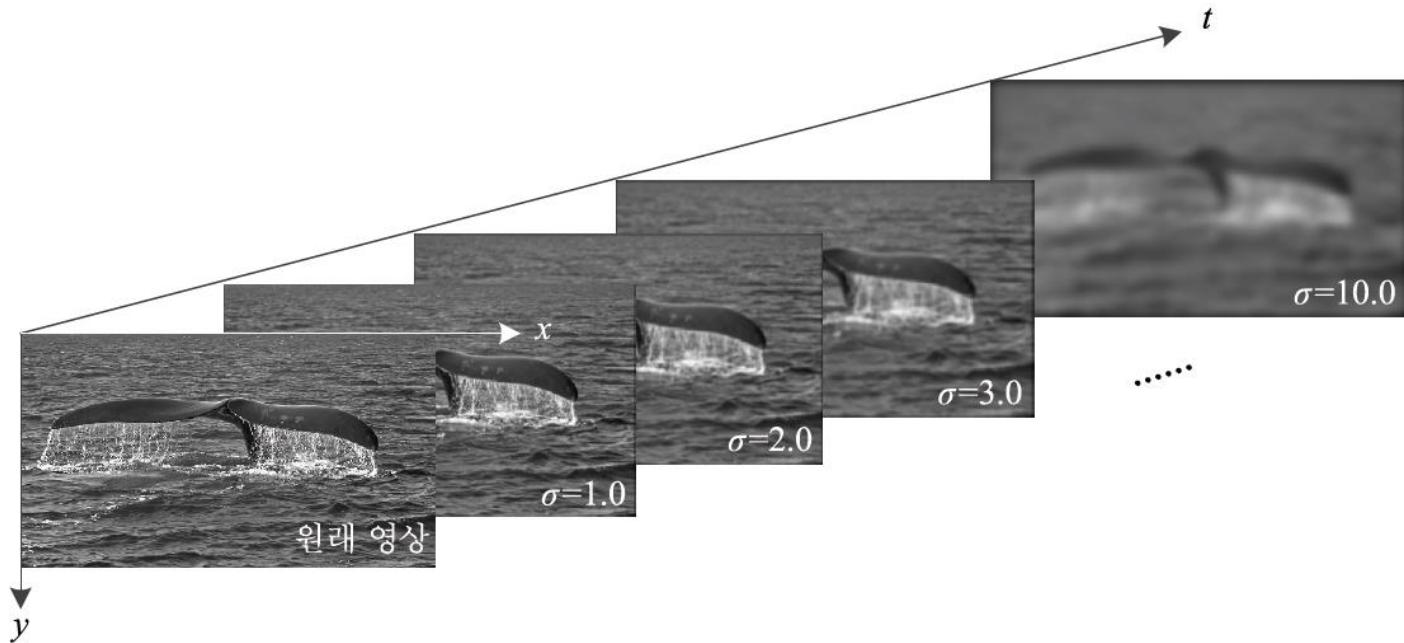


그림 5-8 스케일 공간 (y, x, t) 에 정의된 다중 스케일 영상[오일석2014]

- 스케일 공간의 미분 ([알고리즘 5-1]의 2행)은 정규 라플라시안 사용

$$\text{라플라시안: } \nabla^2 f = d_{yy} + d_{xx} \quad (5.10)$$

$$\text{정규 라플라시안: } \nabla^2_{normal} f = \sigma^2 |d_{yy} + d_{xx}| \quad (5.11)$$

5.4 SIFT

■ SIFT는 [알고리즘 5-1]을 구현하는 가장 성공적인 방법

- 5.4.1절은 [알고리즘 5-1]로 SIFT 검출하는 과정
- 5.4.2절은 특징점에서 기술자를 추출하는 과정
- 5.5절은 특징점을 빠르게 매칭하는 방법

5.4.1 검출

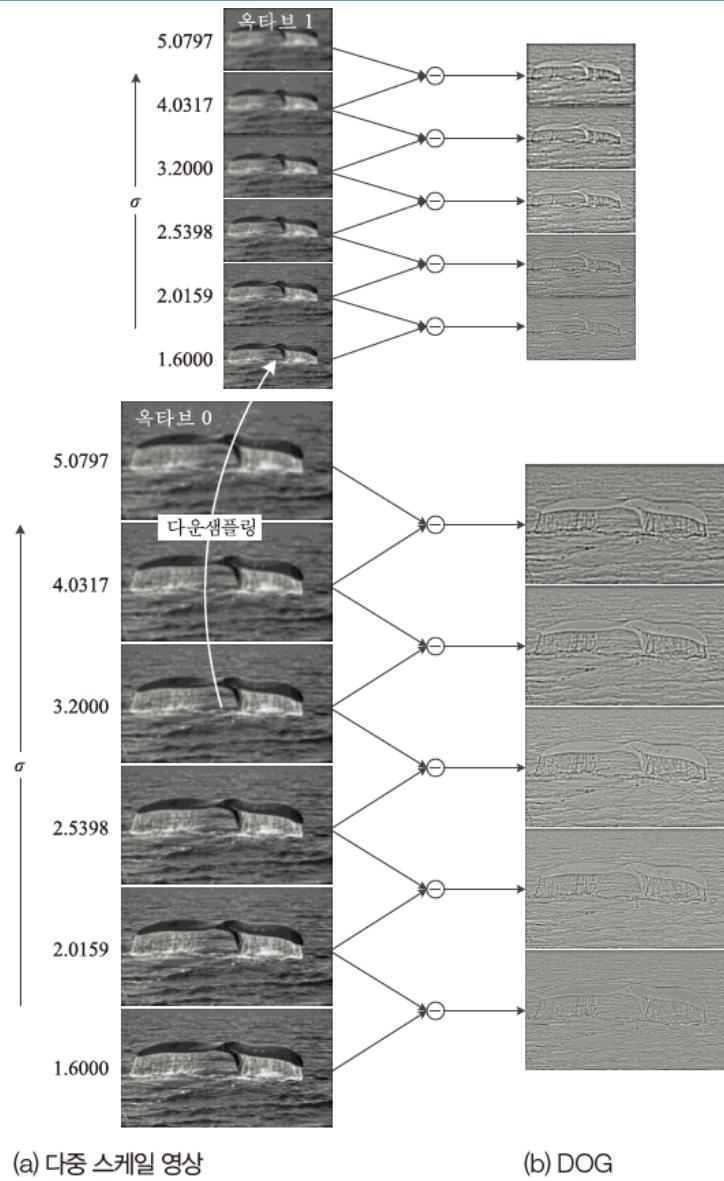
■ 1단계: 다중 스케일 영상 구축

- 가우시안 스무딩과 피라미드 방법을 결합해 사용 ([그림 5-9])

■ 2단계: 다중 스케일 영상에 미분 적용

- 식 (5.11)의 정규 라플라시안 사용. 시간이 많이 걸려 유사한 DOG_{difference of Gaussian} 사용
- DOG는 가우시안 영상은 이미 있고 이웃한 가우시안을 빼면 되기 때문에 획기적으로 빠름

5.4.1 검출



(a) 다중 스케일 영상

(b) DOG

그림 5-9 SIFT의 다중 스케일 영상[오일석2014]

5.4.1 검출

■ 3단계: 극점 검출

- 3차원에서 비최대 억제 적용

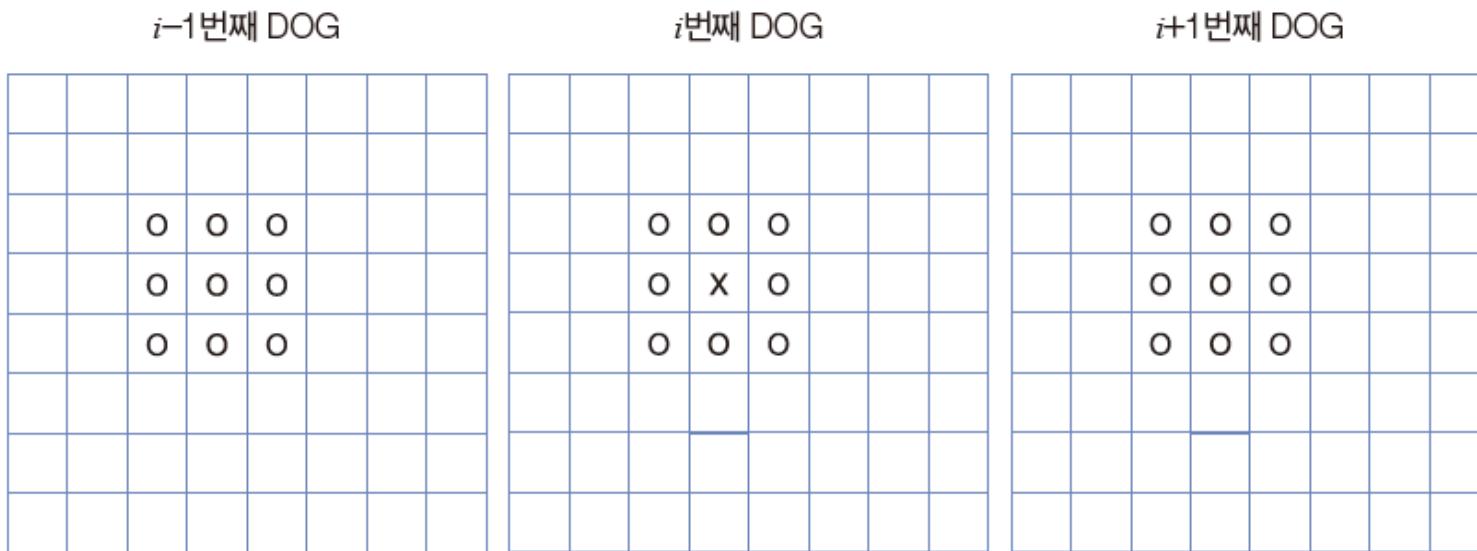


그림 5-10 3차원 구조의 DOG 영상에서 특징점(키포인트) 검출

■ 특징점은 (y, x, o, i) 로 표현

- (y, x) 는 위치 표현. 옥타브 o 와 DOG 번호 i 는 스케일 표현

5.4.2 SIFT 기술자

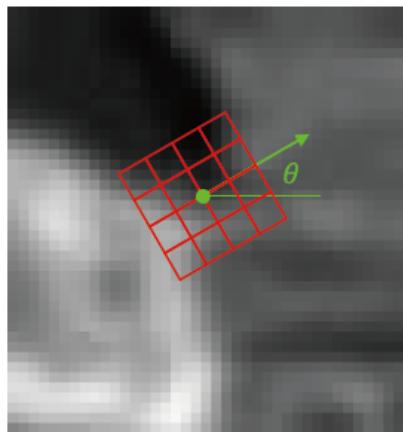
■ 특징점 주위를 살펴 풍부한 정보를 가진 기술자 추출

- 불변성 달성이 매우 중요

■ 기술자 추출 알고리즘(그림은 다음 슬라이드)

- o 와 i 로 가장 가까운 가우시안을 결정하고 거기서 기술자 추출하여 스케일 불변성 달성
- 기준 방향을 정하고 기준 방향을 중심으로 특징을 추출하여 회전 불변성 달성
- 기술자 \mathbf{x} 를 단위 벡터로 바꿈으로써 조명 불변성 달성

5.4.2 SIFT 기술자



(a) 지배적인 방향의 윈도우



(b) 16×16 부분 영역 샘플링과 기술자 추출

그림 5-11 SIFT 특징점에서 기술자 추출

5.4.2 SIFT 기술자

프로그램 5-2

SIFT 검출

```
01 import cv2 as cv
02
03 img=cv.imread('mot_color70.jpg')          # 영상 읽기
04 gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
05
06 sift=cv.SIFT_create()
07 kp,des=sift.detectAndCompute(gray,None)
08
09 gray=cv.drawKeypoints(gray,kp,None,flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_
KEYPOINTS)
10 cv.imshow('sift', gray)
11
12 k=cv.waitKey()
13 cv.destroyAllWindows()
```

5.4.2 SIFT 기술자



5.4.2 SIFT 기술자

■ SIFT의 변종

- 특징점 검출에 SURF, FAST, AGAST 등
- 기술자 추출에 PCA-SIFT, GLOH, 모양 컨텍스트, 이진 기술자 등

5.5 매칭

■ 매칭은 컴퓨터 비전의 다양한 문제에서 핵심 역할

- 물체 인식
- 물체 추적
- 스테레오
- 카메라 캘리브레이션
-

5.5.1 매칭 전략

■ 매칭은 꽤 까다로운 문제

- 두 영상 모두 특징점이 많아 후보 쌍이 아주 많음
- 잡음이 섞인 기술자

5.5.1 매칭 전략

■ 문제의 이해

- 두 영상에서 추출한 기술자 집합 $A = \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m\}$ 과 $B = \{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$ 에서 같은 물체의 같은 곳에서 추출된 \mathbf{a}_i 와 \mathbf{b}_j 쌍을 모두 찾는 문제
- 매칭을 적용하는 다양한 상황
 - 물체 인식에서는 물체의 모델 영상이 A 고 장면 영상이 B
 - 물체 추적이나 스테레오오는 두 영상이 동등한 입장
- 가장 쉬운 매칭 방법
 - mn 개 쌍 각각에 대해 거리 계산하고 거리가 임계값보다 작은 쌍을 모두 취함
 - [그림 5-12]는 이런 순진한 방법의 한계를 보여줌

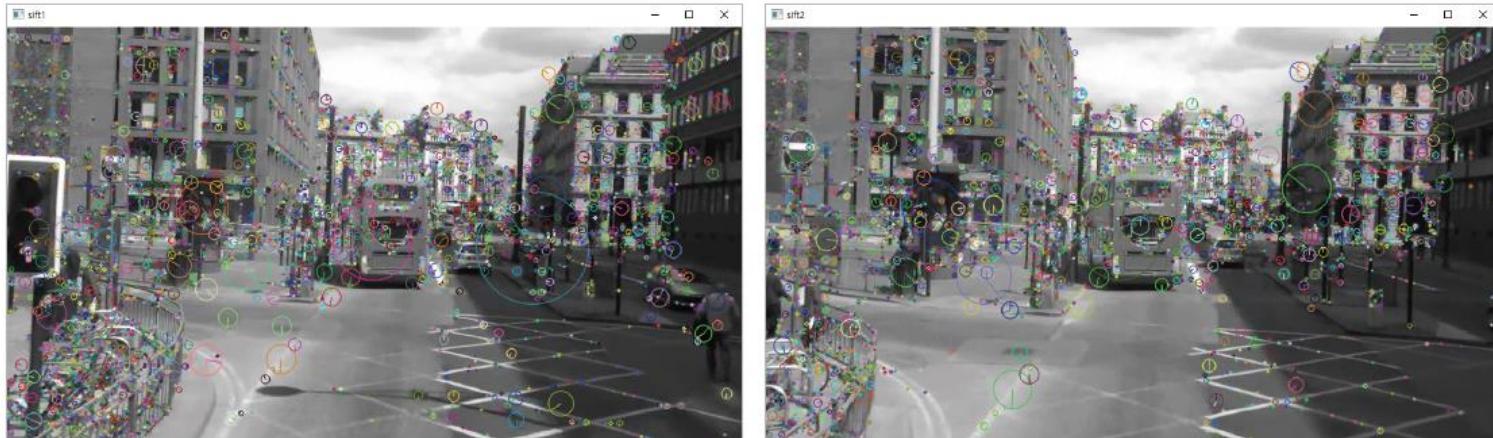


그림 5-12 두 장의 영상에서 추출한 SIFT 특징점을 어떻게 매칭할까?

5.5.1 매칭 전략

■ 두 기술자의 거리 계산

■ 매칭 전략

- 고정 임계값: 식 (5.13)을 만족하는 모든 쌍을 매칭으로 간주
- 최근접 이웃: \mathbf{a}_i 는 가장 가까운 \mathbf{b}_j 를 찾고, \mathbf{a}_i 와 \mathbf{b}_j 가 식 (5.13)을 만족하면 매칭 쌍

$$d(\mathbf{a}_i, \mathbf{b}_j) < T \quad (5.13)$$

- 최근접 이웃 거리 비율: \mathbf{a}_i 는 가장 가까운 \mathbf{b}_j 와 두 번째 가까운 \mathbf{b}_k 를 찾음. \mathbf{b}_j 와 \mathbf{b}_k 가 식 (5.14)를 만족하면 \mathbf{a}_i 와 \mathbf{b}_j 는 매칭 쌍. 세 가지 전략 중 가장 높은 성능

$$\frac{d(\mathbf{a}_i, \mathbf{b}_j)}{d(\mathbf{a}_i, \mathbf{b}_k)} < T \quad (5.14)$$

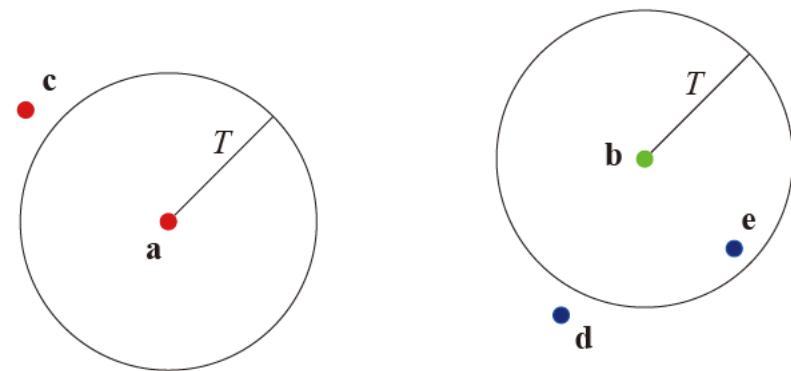


그림 5-13 세 가지 매칭 전략의 비교

5.5.2 매칭 성능 측정

■ 성능 측정은 컴퓨터 비전에서 아주 중요

- 알고리즘 개선이나 최선의 알고리즘을 선택하는 기준
- 현장 투입 여부 결정하는 기준

■ 정밀도와 재현률



(a) 참 긍정



(b) 거짓 부정



(c) 거짓 긍정



(d) 참 부정

그림 5-14 매칭의 네 가지 경우(같은 색이 진짜 매칭 쌍이고 ----은 매칭 알고리즘이 맺어준 쌍)

표 5-2 혼동 행렬

		정답(GT)	
		긍정	부정
예측	긍정	참 긍정(TP)	거짓 긍정(FP)
	부정	거짓 부정(FN)	참 부정(TN)

$$\left. \begin{aligned} \text{정밀도} &= \frac{TP}{TP + FP} \\ \text{재현율} &= \frac{TP}{TP + FN} \end{aligned} \right\} \quad (5.15)$$

$$F1 = \frac{2 \times \text{정밀도} \times \text{재현율}}{\text{정밀도} + \text{재현율}}$$

$$\text{정확율} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.16)$$

5.5.2 매칭 성능 측정

■ ROC Receiver Operating Characteristic 곡선과 AUC Area Under Curve

- 식 (5.14)에서 T 를 작게 하면 거짓 긍정률_{FPR, False Positive Rate}은 작아짐
- T 를 크게 하면 거짓 긍정률이 커지는데 참 긍정률_{TPR, True Positive Rate}도 따라 커지는 경향

$$\left. \begin{array}{l} \text{참 긍정률} = \frac{TP}{TP + FN} \\ \text{거짓 긍정률} = \frac{FP}{TN + FP} \end{array} \right\} \quad (5.17)$$

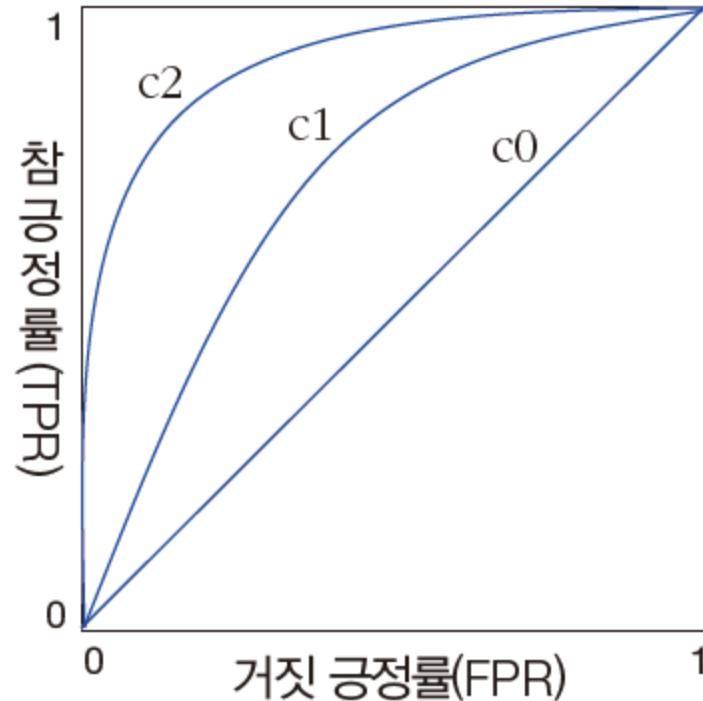
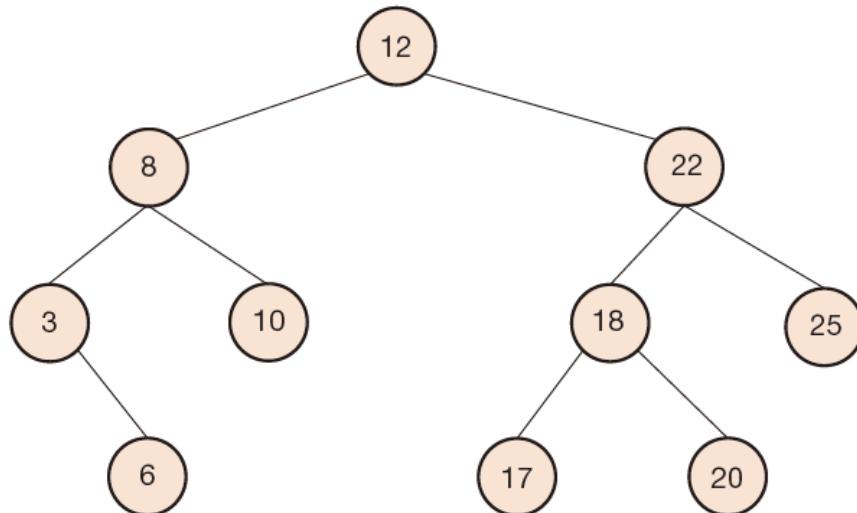


그림 5-15 ROC 곡선

5.5.3 빠른 매칭

■ 속도라는 성능 지표

- 실시간 요구되는 응용에서 양보할 수 없는 강한 조건
- 컴퓨터 과학에서 개발한 빠른 자료 구조 도입해 사용
 - 이진 탐색 트리 $\text{binary search tree}$ 와 해싱 hashing



(a) 이진 탐색 트리

해시 함수
 $h(x) = x \% 13$

0	
1	14
2	
3	
4	134
5	
6	
7	7
8	
9	
10	65023
11	
12	

(b) 해싱

그림 5-16 빠른 탐색을 위한 자료 구조

5.5.3 빠른 매칭

■ *kd* 트리

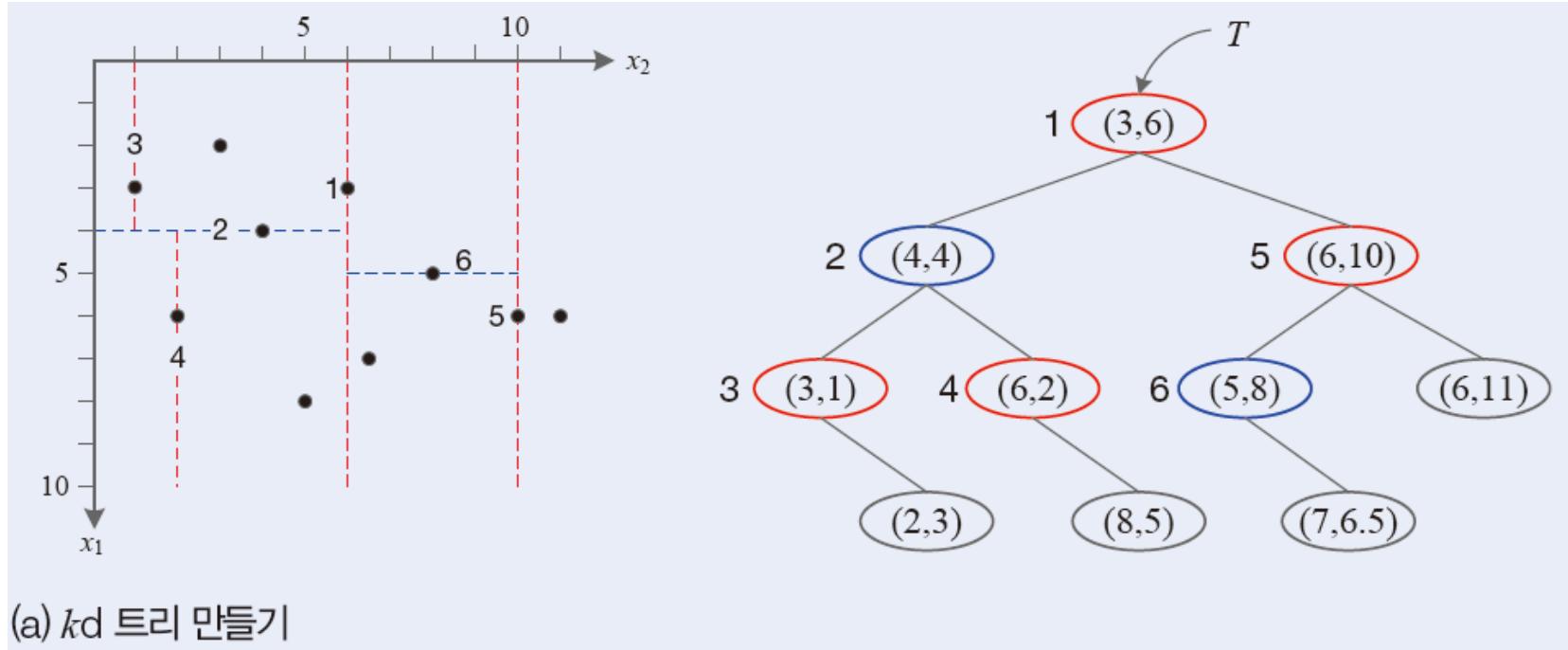
- 특징점 매칭의 독특한 성질 때문에 이진 탐색 트리 그대로 적용 불가능
 - 특징점은 여러 값으로 구성된 특징 벡터임
 - 같은 값이 아니라 최근접 이웃을 찾음
- *kd* 트리는 이런 경우에 적합한 자료 구조
 - *kd* 트리 만들기 과정 [그림 5-17(a)]
 - *kd* 트리 탐색 [그림 5-17(b)]

5.5.3 빠른 매칭

■ [예시 5-2] $d=2, m=10$ 인 상황

- $X = \{\mathbf{x}_1 = (3,1), \mathbf{x}_2 = (2,3), \mathbf{x}_3 = (6,2), \mathbf{x}_4 = (4,4), \mathbf{x}_5 = (3,6), \mathbf{x}_6 = (8,5), \mathbf{x}_7 = (7,6.5), \mathbf{x}_8 = (5,8), \mathbf{x}_9 = (6,10), \mathbf{x}_{10} = (6,11)\}$
- 두 축 값은 $(3,2,6,4,\dots,6)$ 과 $(1,3,2,4,\dots,11)$. 두 번째 축의 분산이 더 커서 두 번째 축 선택
- 두 번째 축 $(1,3,2,4,\dots,11)$ 정렬. 중앙값은 $j=5$, 즉 다섯 번째 기술자 \mathbf{x}_5 가 분할 기준
- \mathbf{x}_5 와 두 번째 축을 기준으로 나머지 9개 기술자를 분할하면
$$X_{left} = \{\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_6\} \text{과 } X_{right} = \{\mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9, \mathbf{x}_{10}\}$$
- 루트 노드에 \mathbf{x}_5 를 배치하고 2번 축을 기준으로 분할했다는 정보를 기록
- X_{left} 와 X_{right} 각각에 대해 같은 과정을 재귀적으로 반복

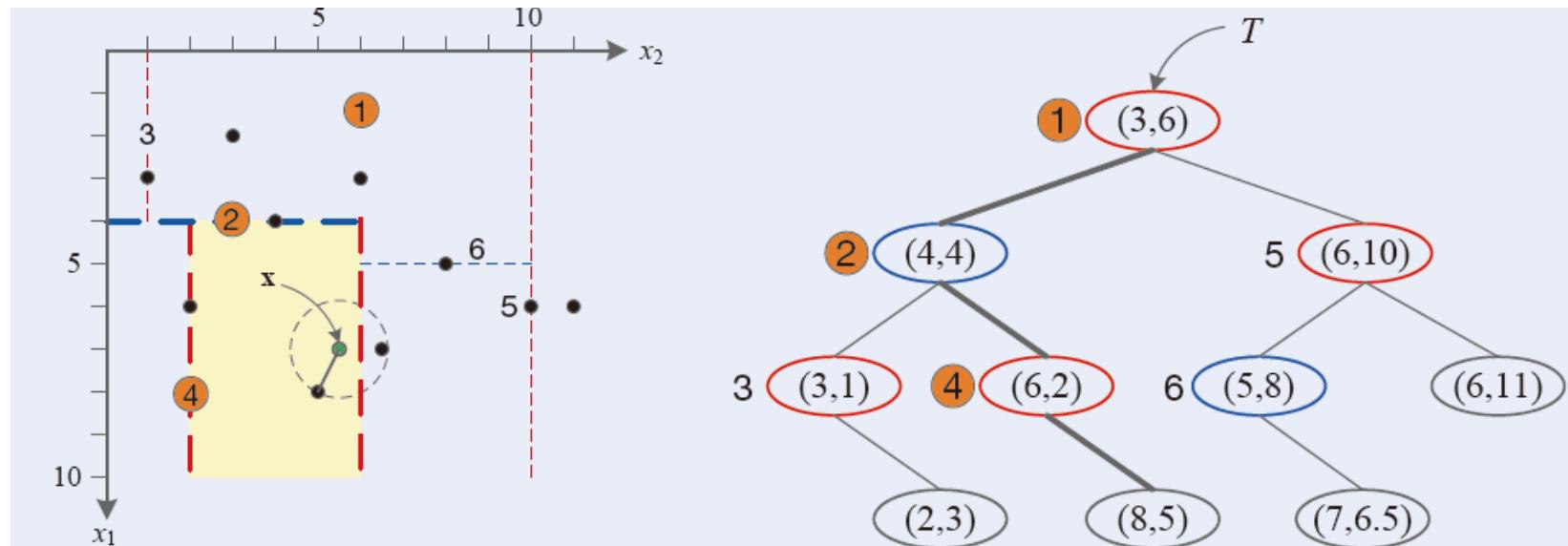
5.5.3 빠른 매칭



5.5.3 빠른 매칭

■ [그림 5-17(b)]

- 새로운 특징 벡터 $x = (7, 5.5)$ 가 입력되었다고 가정하고 최근접 이웃을 찾는 과정



(b) kd 트리 탐색

그림 5-17 kd 트리 만들기와 kd 트리 탐색[오일석2014]

5.5.3 빠른 매칭

■ 위치 의존 해싱

- 일반적인 해싱 [그림 5-16(b)]
 - 해시 함수로 $O(1)$ 시간에 탐색 달성
 - 효율적인 해시 함수와 충돌 해결 기법이 개발되어 있음
- 특징점 매칭에 사용하려면 개조 필요
 - 특징 벡터이고 최근접 이웃 찾아야 하기 때문
 - 충돌을 최소화하는 일반 해싱과 달리 비슷한 특징 벡터를 같은 칸에 담아야 함
 - 위치 의존 해싱은 이런 조건을 만족하는 방법

■ 빠른 매칭을 보장하는 라이브러리: FLANN과 FAISS

5.5.4 프로그래밍 실습: FLANN을 이용한 특징점 매칭

프로그램 5-3

FLANN 라이브러리를 이용한 SIFT 매칭

```
01 import cv2 as cv
02 import numpy as np
03 import time
04
05 img1=cv.imread('mot_color70.jpg')[190:350,440:560] # 버스를 크롭하여 모델 영상으로 사용
06 gray1=cv.cvtColor(img1,cv.COLOR_BGR2GRAY)
07 img2=cv.imread('mot_color83.jpg') # 장면 영상
08 gray2=cv.cvtColor(img2,cv.COLOR_BGR2GRAY)
09
10 sift=cv.SIFT_create()
11 kp1,des1=sift.detectAndCompute(gray1,None)
12 kp2,des2=sift.detectAndCompute(gray2,None)
13 print('특징점 개수:',len(kp1),len(kp2)) ①
14
```

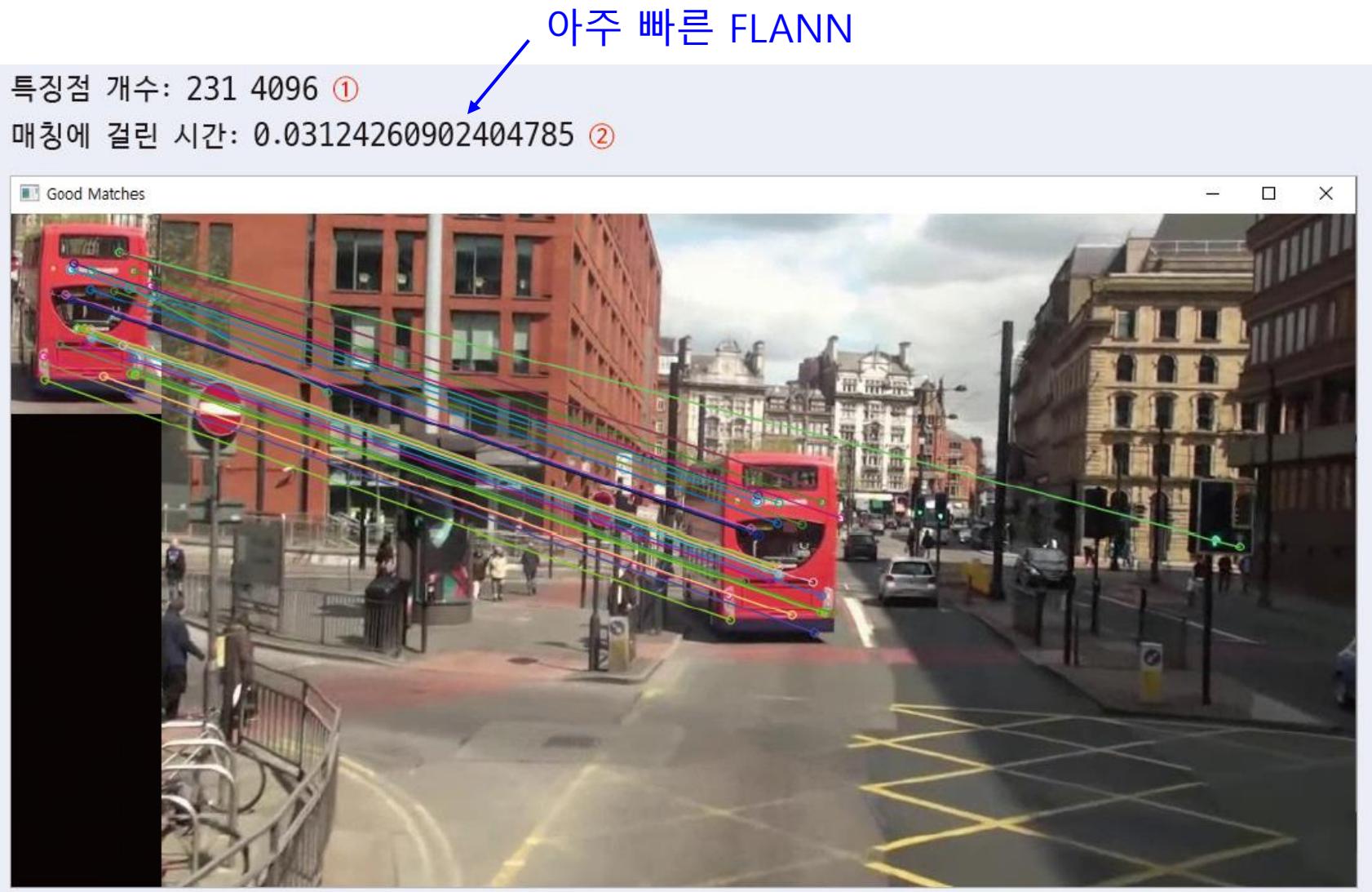
5.5.4 프로그래밍 실습: FLANN을 이용한 특징점 매칭

```
15 start=time.time()
16 flann_matcher=cv.DescriptorMatcher_create(cv.DescriptorMatcher_FLANNBASED)
17 knn_match=flann_matcher.knnMatch(des1,des2,2)
18
19 T=0.7
20 good_match=[]
21 for nearest1,nearest2 in knn_match:
22     if (nearest1.distance/nearest2.distance)<T:
23         good_match.append(nearest1)
24 print('매칭에 걸린 시간:',time.time()-start) ②
25
26 img_match=np.empty((max(img1.shape[0],img2.shape[0]),img1.shape[1]+img2.
shape[1],3),dtype=np.uint8)
27 cv.drawMatches(img1,kp1,img2,kp2,good_match,img_match,flags=cv.
DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
28
29 cv.imshow('Good Matches', img_match)
30
31 k=cv.waitKey()
32 cv.destroyAllWindows()
```

식 (5.14)의 최근접 이웃 거리 비율 적용



5.5.4 프로그래밍 실습: FLANN을 이용한 특징점 매칭



5.6 호모그래피 추정

■ SIFT 검출과 기술자 추출 이후에 해야 할 일

- 물체 위치 찾기(아웃라이어 걸러내기)
- 호모그래피 homography는 이런 일을 해줌

5.6.1 문제의 이해

■ 3차원 투영

- 3차원 공간에 있는 평면 P의 두 점 p_1 과 p_2 그리고 카메라 A와 B가 있는 상황
- p_1 과 p_2 가 카메라의 2차원 영상 평면에 투영 변환됨

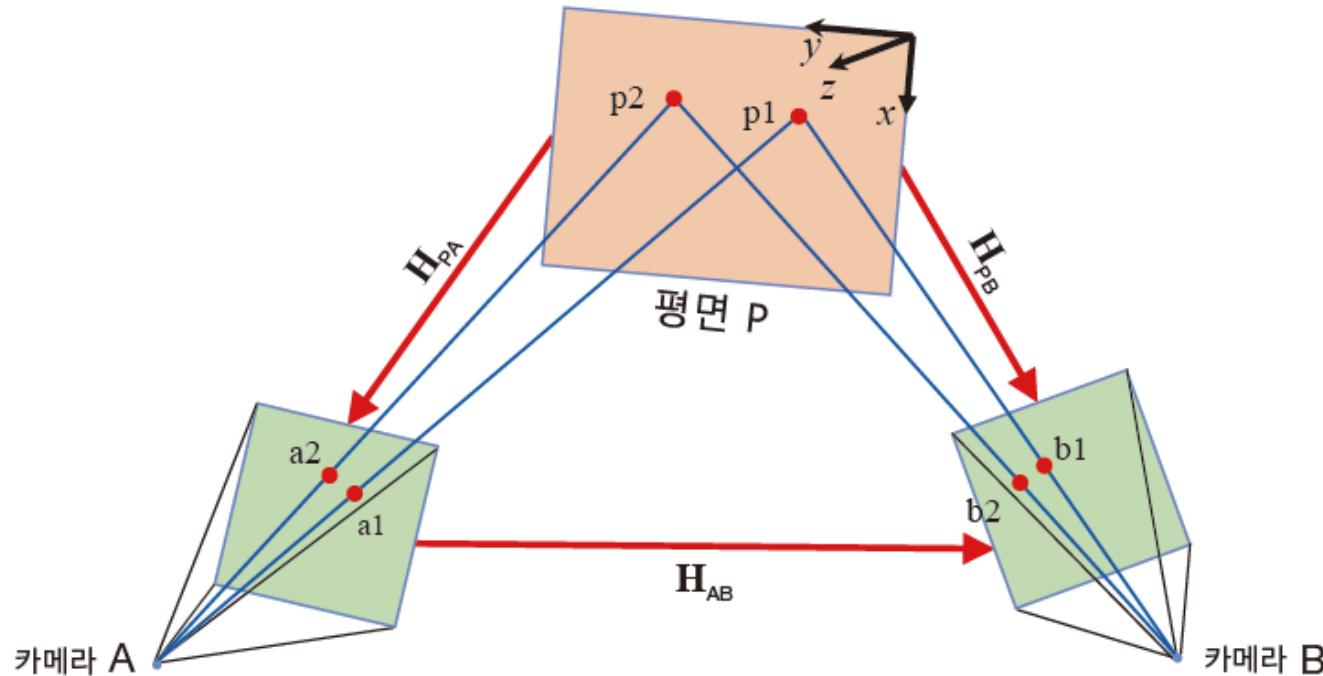


그림 5-18 호모그래피

5.6.1 문제의 이해

■ 투영 변환

- 3차원 점 p 를 동차 좌표로 표현하면 $(x,y,z,1)$. 투영 변환 행렬은 4×4
- p_1 과 p_2 가 같은 평면 상에 있다고 가정하고 $z=0$ 으로 간주하면 3×3 행렬로 표현 가능
- 이런 제한된 상황에서 이루어지는 투영 변환을 평면 호모그래피(줄여 호모그래피)라 부름

■ [그림 5-18]에는 세 개의 평면이 있음

- 물체가 놓인 평면 P , 카메라 A와 B의 영상 평면
- 어떤 평면의 점 \mathbf{a} 를 다른 평면의 점 \mathbf{b} 로 투영하는 변환 행렬을 \mathbf{H} 라 하면,

$$\mathbf{b}^T = \begin{pmatrix} b_x \\ b_y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ 1 \end{pmatrix} = \mathbf{H}\mathbf{a}^T \quad (5.18)$$

풀어쓰면, $b_x = \frac{h_{00}a_x + h_{01}a_y + h_{02}}{h_{20}a_x + h_{21}a_y + 1}$, $b_y = \frac{h_{10}a_x + h_{11}a_y + h_{12}}{h_{20}a_x + h_{21}a_y + 1}$ (5.19)

5.6.1 문제의 이해

■ 방정식을 풀어 \mathbf{H} 구하기

- 매칭 쌍 $(\mathbf{a}_1, \mathbf{b}_1), (\mathbf{a}_2, \mathbf{b}_2), (\mathbf{a}_3, \mathbf{b}_3), \dots$ 을 가지고 품
- 알아내야 할 값은 8개. 매칭 쌍 하나가 두 방정식을 제공하므로 최소 4개 매칭 쌍이면 됨
- 실제에서는 많은 매칭 쌍을 이용하여 최적의 \mathbf{H} 를 계산

■ 매칭 쌍을 가지고 \mathbf{H} 추정

- 매칭 쌍 n 개를 $(\mathbf{a}_1, \mathbf{b}_1), (\mathbf{a}_2, \mathbf{b}_2), (\mathbf{a}_3, \mathbf{b}_3), \dots, (\mathbf{a}_n, \mathbf{b}_n)$ 으로 표기하면 식 (5.20)이 성립
 - \mathbf{B} 는 \mathbf{b}_i 를 i 번째 열에 배치한 $3 * n$ 행렬이고 \mathbf{A} 는 \mathbf{a}_i 를 i 번째 열에 배치한 $3 * n$ 행렬

$$\mathbf{B} = \mathbf{HA} \quad (5.20)$$

5.6.2 강인한 호모그래피 추정

■ 최소평균제곱오차 방법

- 식 (5.21)의 E 가 최소인 \mathbf{H} 를 찾음
 - numpy의 lstsq 또는 scipy의 leastsq 함수 사용하여 풀 수 있음
 - 모든 매칭 쌍이 같은 자격으로 참여하므로 강인함 없음

$$E = \frac{1}{n} \sum_{i=1,n} \| \mathbf{H} \mathbf{a}_i^T - \mathbf{b}_i \|_2^2 \quad (5.21)$$

■ 식 (5.21)의 평균 대신 중앙값 사용하면 강인함 확보 가능

- 아웃라이어는 중앙값 계산까지만 영향을 미치고 수렴 여부 결정에서는 빠짐

5.6.2 강인한 호모그래피 추정

■ 더욱 강인한 RANSAC

[알고리즘 5-2] 호모그래피 추정을 위한 RANSAC

입력: 매칭 쌍 집합 $X = \{(\mathbf{a}_1, \mathbf{b}_1), (\mathbf{a}_2, \mathbf{b}_2), \dots, (\mathbf{a}_n, \mathbf{b}_n)\}$, 반복 횟수 m , 임곗값 t, d, e

출력: 최적 호모그래피 $\hat{\mathbf{H}}$

1. $h = []$
2. for $j=1$ to m
 3. X 에서 네 쌍을 랜덤하게 선택하고 식 (5.21)을 풀어 호모그래피 행렬 \mathbf{H} 를 추정한다.
 4. 이들 네 쌍으로 *inlier* 집합을 초기화한다.
 5. for (3행에서 선택한 네 쌍을 제외한 모든 쌍 p 에 대해)
 6. if (p 가 허용 오차 t 이내로 \mathbf{H} 에 적합하면) p 를 *inlier*에 삽입한다.
 7. if (*inlier*가 d 개 이상의 요소를 가지면)
 8. *inlier*의 모든 요소를 가지고 호모그래피 행렬 \mathbf{H} 를 다시 추정한다.
 9. if (8행에서 적합 오차가 e 보다 작으면) \mathbf{H} 를 h 에 삽입한다.
 10. h 에 있는 호모그래피 중에서 가장 좋은 것을 $\hat{\mathbf{H}}$ 로 취한다.

5.6.3 프로그래밍 실습: 호모그래피 추정

프로그램 5-4

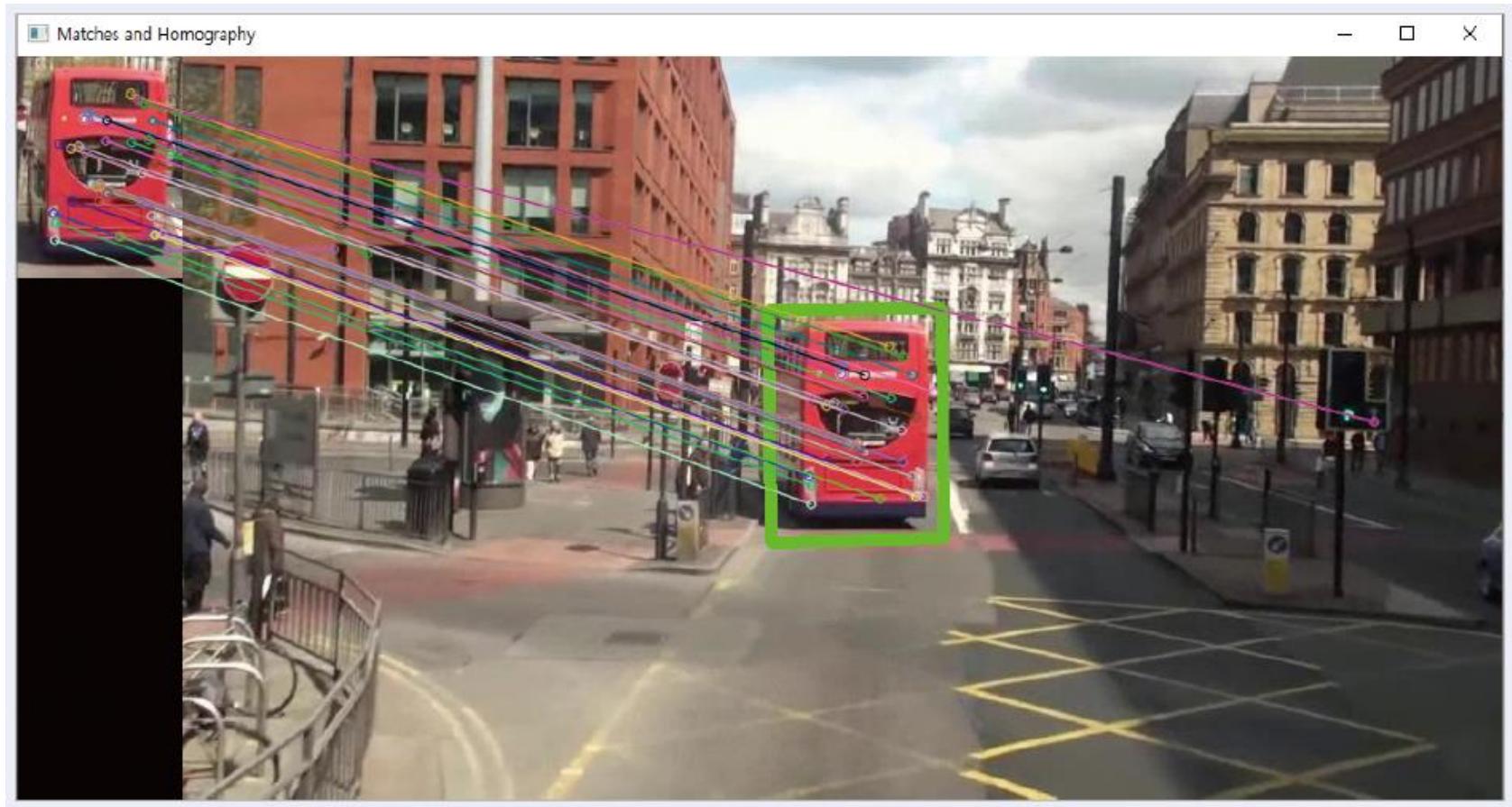
RANSAC을 이용해 호모그래피 추정하기

```
01 import cv2 as cv
02 import numpy as np
03
04 img1=cv.imread('mot_color70.jpg')[190:350,440:560] # 버스를 크롭하여 모델 영상으로 사용
05 gray1=cv.cvtColor(img1,cv.COLOR_BGR2GRAY)
06 img2=cv.imread('mot_color83.jpg')                      # 장면 영상
07 gray2=cv.cvtColor(img2,cv.COLOR_BGR2GRAY)
08
09 sift=cv.SIFT_create()
10 kp1,des1=sift.detectAndCompute(gray1,None)
11 kp2,des2=sift.detectAndCompute(gray2,None)
12
13 flann_matcher=cv.DescriptorMatcher_create(cv.DescriptorMatcher_FLANNBASED)
14 knn_match=flann_matcher.knnMatch(des1,des2,2)          # 최근접 2개
15
16 T=0.7
17 good_match=[]
18 for nearest1,nearest2 in knn_match:
19     if (nearest1.distance/nearest2.distance)<T:
20         good_match.append(nearest1)
21
22 points1=np.float32([kp1[gm.queryIdx].pt for gm in good_match])
23 points2=np.float32([kp2[gm.trainIdx].pt for gm in good_match])
24
```

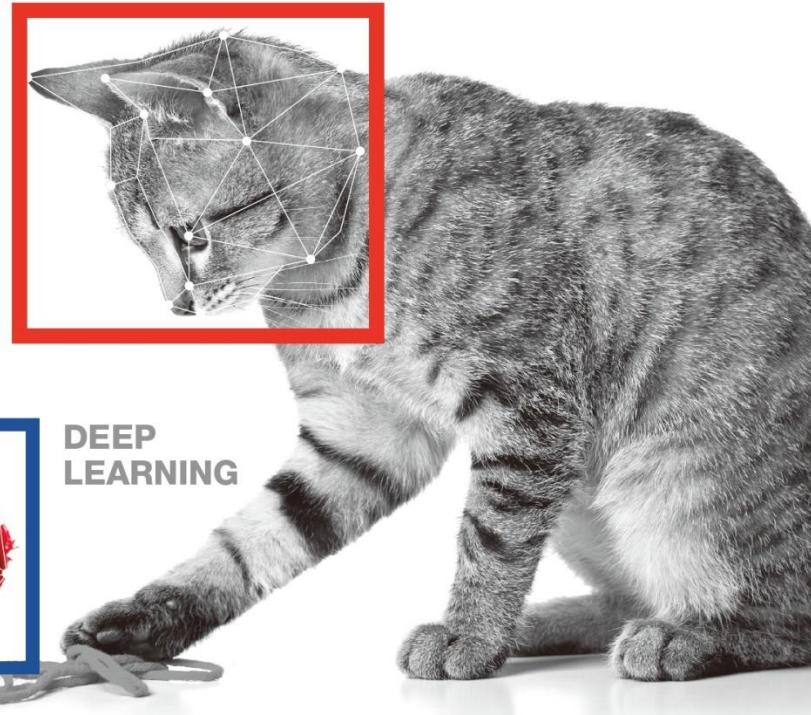
5.6.3 프로그래밍 실습: 호모그래피 추정

```
25 H,_=cv.findHomography(points1,points2,cv.RANSAC)
26
27 h1,w1=img1.shape[0],img1.shape[1]                      # 첫 번째 영상의 크기
28 h2,w2=img2.shape[0],img2.shape[1]                      # 두 번째 영상의 크기
29
30 box1=np.float32([[0,0],[0,h1-1],[w1-1,h1-1],[w1-1,0]]).reshape(4,1,2)
31 box2=cv.perspectiveTransform(box1,H)
32
33 img2=cv.polyline(img2,[np.int32(box2)],True,(0,255,0),8)
34
35 img_match=np.empty((max(h1,h2),w1+w2,3),dtype=np.uint8)
36 cv.drawMatches(img1,kp1,img2,kp2,good_match,img_match,flags=cv.
DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
37
38 cv.imshow('Matches and Homography',img_match)
39
40 k=cv.waitKey()
41 cv.destroyAllWindows()
```

5.6.3 프로그래밍 실습: 호모그래피 추정



COMPUTER VISION



컴퓨터 비전과 딥러닝

감사합니다.