

컴퓨터 비전과 딥러닝

Chapter 05 지역 특징

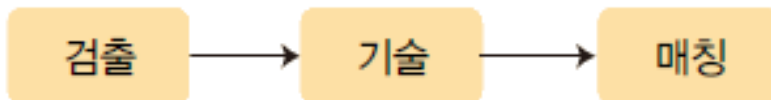
PREVIEW

■ 대응점 문제 correspondence problem

- 같은 장면을 다른 시점에서 찍은 두 영상에서 대응하는 점의 쌍을 찾는 문제
- 파노라마 영상, 물체 인식/추적, 스테레오 비전, 카메라 캘리브레이션 등에 필수



- 대응점 문제는 세 단계로 해결



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

5.1 발상

■ 물체 추적에서 대응점 찾기

- 예) 다중 물체 추적 챌린지 MOT-17-14-SDP 동영상 데이터셋
- 반복성 repeatability 이 뛰어난 특징 필요 (두 영상에서 모두에서 추출되어야 함)



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

■ 무엇을 대응점으로 쓸 것인가?

- 에지: 에지 강도와 방향 정보만 가지므로, 매칭에 참여하기에 부족

■ 1980년대에는 에지 경계선에서 모퉁이_{corner} 찾는 연구 왕성

- 특징점이 물체의 실제 모퉁이에 해당해야한다는 생각이 지배적
- 2000년대 초에 자취를 감춤.
- 지역 특징 (local features) 이라는 대안이 떠오름

5.1 발상

■ 지역 특징의 발상

- 좁은 지역을 보고 특징점 여부 결정 ([그림 5-2] 녹색 박스)
- 물체의 실제 모퉁이에 위치해야 한다는 생각을 버림
- 반복성을 더 중요하게 취급하는 발상의 전환
 - 두 영상에서 모두 같은 위치에서 추출되어야 함

■ 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 지역 특징(local features)

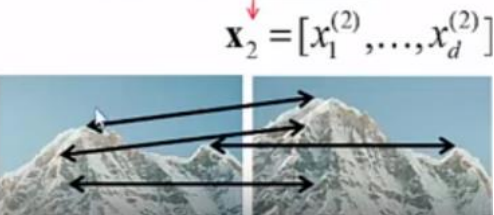
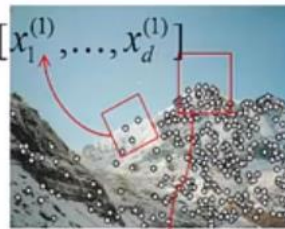
- <위치, 스케일, 방향, 특징 벡터> = $((y, x), s, \theta, \mathbf{x})$ 로 표현

Local features: main components

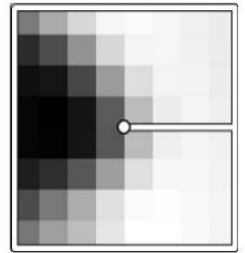
- 1) Detection: Identify the interest points



- 2) Description :Extract feature vector descriptor surrounding each interest point.



- 3) Matching: Determine correspondence between descriptors in two views

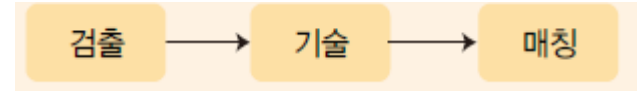


Once a local scale and orientation estimate has been determined using an 8×8 sampling of bias and gain normalized averaging of five pixels relative to the detection scale (Bro

5.1 지역 특징의 조건

■ 지역 특징이 만족해야 할 특성

- 반복성(repetability)
- 분별력(distinctiveness)
- 지역성(locality)
- 정확성(accuracy)
- 적당한 양
- 계산 효율



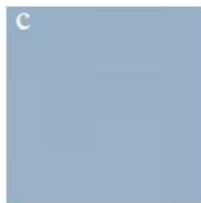
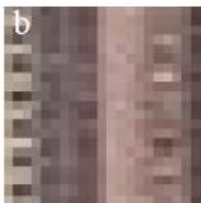
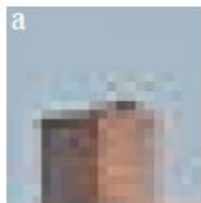
■ 이들 특성은 길항 관계(trade-off)

- 응용에 따라 적절한 특징을 선택해야 함
 - 실시간 작업: 반복성을 희생하더라도 계산 효율을 높여야 함
 - 지역성과 분별력: 분별력을 높이면 지역성이 떨어짐
 - off-line작업인 경우 정확성을 높이고 계산 효율을 낮춤

5.2 지역 특징 검출 원리

■ 원리

- 인지 실험
 - 대응점을 찾기가 쉬운(좋은) 점은? → 사람에게 쉬운 곳이 컴퓨터에게도 쉽다.
- 좋은 정도를 어떻게 수량화할까?
 - 여러 방향으로 밝기 변화가 나타나는 곳일수록 높은 점수
- 아래 그림의 a, b, c중 에서 어느 곳이 지역특징으로 유리?

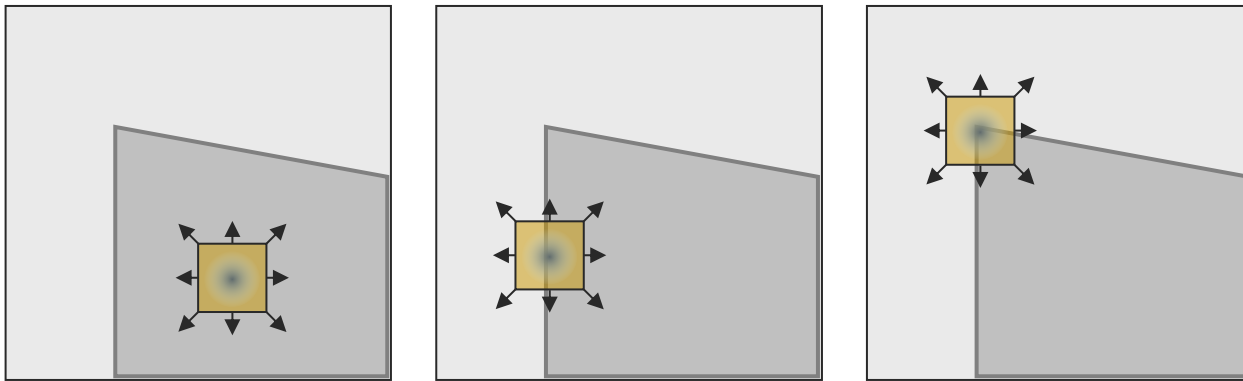


5.2 이동과 회전에 불변한 특징점 검출

- 이제 '어떻게' 찾을 것인지 공부해보자.

5.2.1 모라벡 알고리즘

5.2.2 해리스 코너



5.2.1 모라벡 알고리즘

■ 인지 실험에 주목한 모라벡 [Moravec80]

- 주어진 픽셀에서 제공차의 합으로 밝기 변화 측정

$$S(v, u) = \sum_y \sum_x w(y, x) (f(y + v, x + u) - f(y, x))^2 \quad (4.1)$$

v, u 는 마스크 내의 상대 위치를 나타내는 인덱스로 각 각 3x3인 경우 $\{-1, 0, 1\}$

예제 4-1 제공차 합 계산

[그림 4-3]은 삼각형을 가진 12×12 영상이다. 현재 조사하고 있는 점은 (5, 3)에 위치한 b이고, 마스크는 모든 값이 1인 3×3 크기의 박스형이라 하자. 이때 오른쪽으로 한 화소만큼 이동시킨 $S(0, 1)$ 을 계산해 보면 다음과 같이 4라는 값을 얻는다.

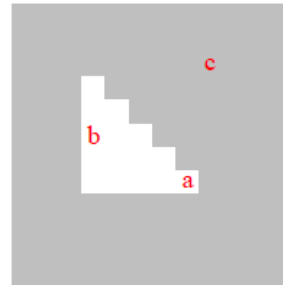
$$\begin{aligned} S(0, 1) &= \sum_y \sum_x w(y, x) (f(y, x + 1) - f(y, x))^2, \text{ 이때 } w(y, x) = \begin{cases} 1, & 4 \leq y \leq 6, 2 \leq x \leq 4 \\ 0, & \text{그 외} \end{cases} \\ &= \sum_{4 \leq y \leq 6} \sum_{2 \leq x \leq 4} (f(y, x + 1) - f(y, x))^2 = 4 \end{aligned}$$

		u		
		-1	0	1
v	-1	3	1	6
	0	3	0	4
	1	3	0	3

같은 방식으로 나머지 v 와 u 값에 대해 $S(v, u)$ 를 계산해 보면, 왼쪽과 같은 $S(v, u)$ 맵을 완성할 수 있다. 점 b를 기준으로 생성한 맵으로, 연필을 들고 다른 점에 대해서도 계산해 보기 바란다. 손으로 직접 해 보는 것의 힘은 생각보다 강하다!

5.2.1 모라벡 알고리즘

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	c	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	1	1	0	0	0	0	0	0	0
5	0	0	0	b	1	1	0	0	0	0	0	0
6	0	0	0	1	1	1	1	0	0	0	0	0
7	0	0	0	1	1	1	1	a	0	0	0	0
8	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
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0



(a) 합성 영상

		u		
		-1	0	1
v	-1	3	4	4
	0	2	0	2
	1	4	3	2
		a		

		u		
		-1	0	1
v	-1	3	1	6
	0	3	0	4
	1	3	0	3
		b		

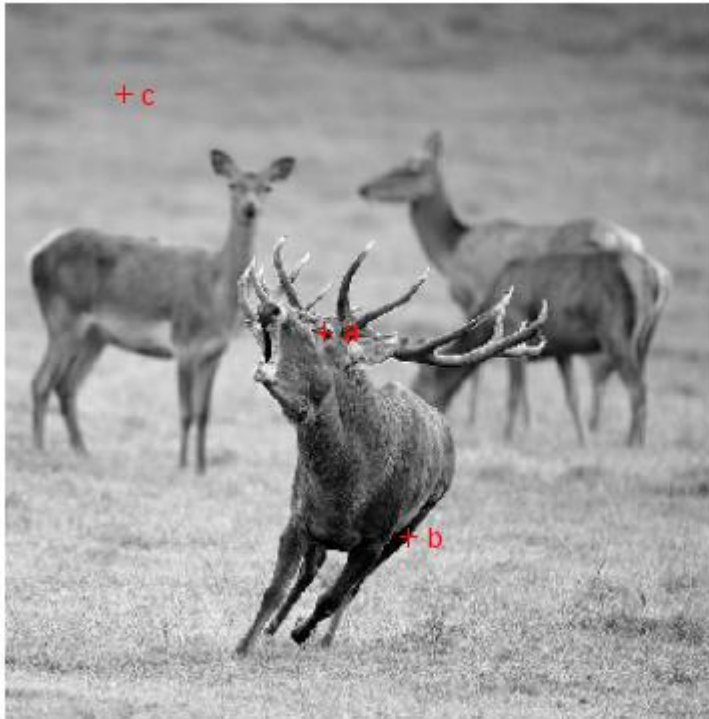
		u		
		-1	0	1
v	-1	0	0	0
	0	0	0	0
	1	0	0	0
		c		

(b) 세 지점에서 $S(v, u)$ 맵

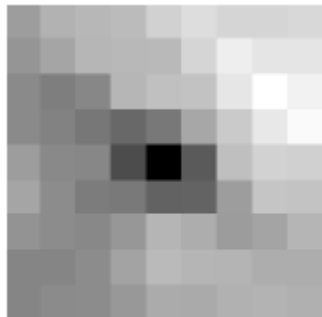
그림 4-3 $S(v, u)$ 맵

← 3*3 마스크로 측정

5.2.1 모라벡 알고리즘



> 원래 영상



> a



> b



> c

← 9*9 마스크로 측정

그림 4-4 실제 영상에서 S 맵(밝을수록 큰 값)

5.2.1 모라벡 알고리즘

■ $S(\cdot)$ 맵을 관찰해 보면,

- a와 같은 코너에서는 모든 방향으로 변화가 심함
- b와 같은 에지에서는 에지 방향으로 변화 적지만, 에지에 수직 방향으로 변화 심함
- c와 같은 곳은 모든 방향으로 변화 적음
- a에 높은 값, c는 아주 낮은 값, b는 그 사이 값을 부여하는 함수를 만들면 됨

■ 모라벡의 함수

- 특징 가능성 값 C

$$C = \min(S(0, 1), S(0, -1), S(1, 0), S(-1, 0)) \quad (4.2)$$

- 한계
 - 한 화소만큼 이동하여 네 방향만 봄
 - 잡음에 대한 대처 방안 없음

5.2.2 해리스 코너

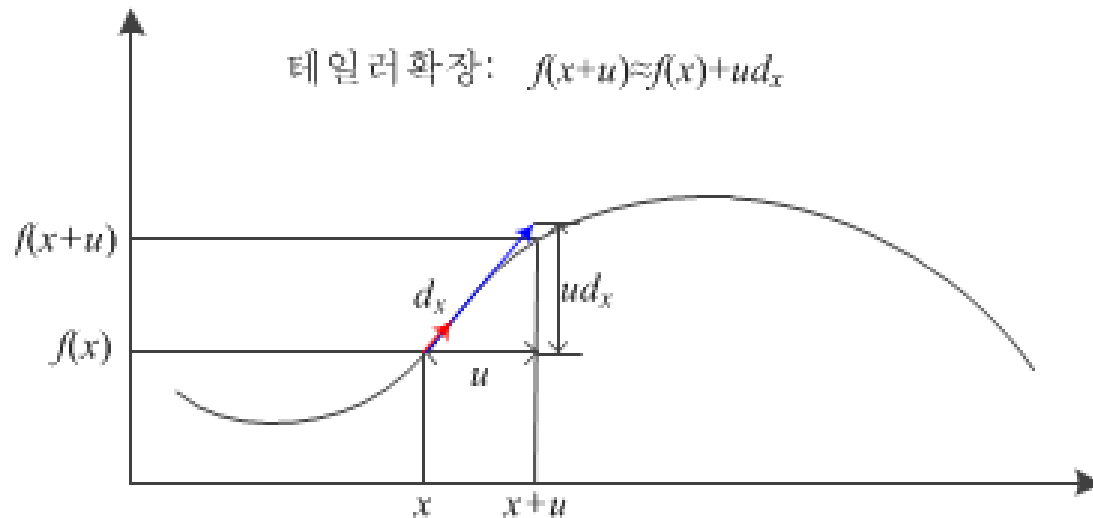
■ 해리스의 접근 [Harris88]

- 가중치 제곱차의 합을 이용한 잡음 대처

$$S(v, u) = \sum_y \sum_x G(y, x) (f(y + v, x + u) - f(y, x))^2 \quad (4.3)$$

- 테일러 확장 $f(y + v, x + u) \cong f(y, x) + vd_y(y, x) + ud_x(y, x)$ 을 대입하면,

$$S(v, u) \cong \sum_y \sum_x G(y, x) (vd_y(y, x) + ud_x(y, x))^2 \quad (4.5)$$



4.2.2 해리스 코너

■ 계속 유도하면,

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

$$S(v, u) \cong (v \ u) \begin{pmatrix} \sum_y \sum_x G(y, x) d_y^2 & \sum_y \sum_x G(y, x) d_y d_x \\ \sum_y \sum_x G(y, x) d_y d_x & \sum_y \sum_x G(y, x) d_x^2 \end{pmatrix} \begin{pmatrix} v \\ u \end{pmatrix}$$

5.2.2 해리스 코너

■ 2차 모멘트 행렬 \mathbf{A}

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

$$\mathbf{A} = \begin{pmatrix} G \circledast d_y^2 & G \circledast d_y d_x \\ G \circledast d_y d_x & G \circledast d_x^2 \end{pmatrix} \quad (4.7)$$

- (v, u) 는 실수 가능
- \mathbf{A} 를 (v, u) 무관하게 계산할 수 있음 ($\because S$ 가 \mathbf{u} 와 \mathbf{A} 의 곱으로 인수 분해되어 있으므로)
- \mathbf{A} 는 현재 위치(픽셀)의 영상 구조를 나타냄 $\rightarrow \mathbf{A}$ 를 잘 분석하면 특징 여부를 판정할 수 있음

5.2.2 해리스 코너

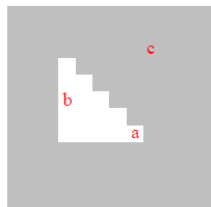
예제 4-2 2차 모멘트 행렬 A 계산

[예제 4-1]에서 사용한 [그림 4-3(a)]의 영상에서 행렬 **A**를 계산하는 과정을 살펴보자. [그림 4-5(a)]는 편의상 같은 영상을 다시 보여주는 것이고, [그림 4-5(b)]는 $d_y, d_x, d_y^2, d_x^2, d_y d_x$ 를 구한 영상이다. d_y 와 d_x 를 구하기 위해 각각 $[-1 \ 0 \ 1]^T$ 와 $[-1 \ 0 \ 1]$ 연산자를 사용하였다. [그림 4-5(c)~(e)]의 영상을 얻기 위해 다음과 같이 $\sigma=1.0$ 인 가우시안 마스크 **G**를 사용하였다.

$$G = \begin{bmatrix} .0751 & .1238 & .0751 \\ .1238 & .2042 & .1238 \\ .0751 & .1238 & .0751 \end{bmatrix}$$

이제 어떤 점의 행렬 **A**를 구할 수 있다. 예를 들어, 점 **a**의 행렬은 $\mathbf{A} = \begin{pmatrix} 0.522 & -0.199 \\ -0.199 & 0.527 \end{pmatrix}$ 이다.

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	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
3	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	1	1	0	0	0	0	0	0	0
5	0	0	0	1	1	1	0	0	0	0	0	0
6	0	0	0	1	1	1	1	0	0	0	0	0
7	0	0	0	1	1	1	1	1	0	0	0	0
8	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
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0



(a) 원래 영상 f



$> d_y$ $> d_x$ $> d_y^2$ $> d_x^2$ $> d_y d_x$

(b) 도함수 영상(흰색은 1, 회색은 0, 검은색은 -1)

5.2.2 해리스 코너

0	0	0	0	0	0	0	0	0	0	0	0
0	0	.075	.124	.075	0	0	0	0	0	0	0
0	0	.199	.403	.323	.075	0	0	0	0	0	0
0	0	.199	.527	.602	.323	.075	0	0	0	0	0
0	0	.075	.323	.602	.602	.323	.075	0	0	0	0
0	0	0	.075	.323	.602	.602	.323	.075	0	0	0
0	0	.075	.199	.349	.597	.726	.478	.124	0	0	0
0	0	.199	.527	.726	.801	.801	.522	.150	0	0	0
0	0	.199	.527	.726	.726	.651	.403	.124	0	0	0
0	0	.075	.199	.274	.274	.274	.199	.075	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



(c) $G \circledast d_y^2$

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	.075	.124	.150	.124	.075	0	0	0	0	0	0
0	.199	.403	.521	.478	.323	.075	0	0	0	0	0
0	.274	.651	.801	.726	.602	.323	.075	0	0	0	0
0	.274	.726	.801	.597	.602	.602	.323	.075	0	0	0
0	.274	.726	.726	.349	.323	.602	.602	.323	.075	0	0
0	.199	.527	.527	.199	.075	.323	.527	.403	.124	0	0
0	.075	.199	.199	.075	0	.075	.199	.199	.075	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	0	0	0	0	0	0



(d) $G \circledast d_x^2$

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	-.075	-.124	-.075	0	0	0	0	0	0
0	0	0	-.199	-.403	-.323	-.075	0	0	0	0	0
0	0	0	-.199	-.527	-.602	-.323	-.075	0	0	0	0
0	0	0	-.075	-.323	-.602	-.602	-.323	-.075	0	0	0
0	0	-.075	-.124	-.150	-.323	-.527	-.403	-.124	0	0	0
0	0	-.075	-.204	-.124	-.075	-.199	-.199	-.075	0	0	0
0	0	-.075	-.124	-.075	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	0	0	0	0	0	0	0	0	0	0	0



(e) $G \circledast d_y d_x$

그림 4-5 2차 모멘트 행렬 A를 구하는 과정

5.2.2 해리스 코너

■ 2차 모멘트 행렬의 고유값 분석

- c와 같이 두 개의 고유값 모두 0이거나 0에 가까우면 → 변화가 거의 없는 곳
- b와 같이 고유값 하나는 크고 다른 하나는 작으면 → 한 방향으로만 변화가 있는 에지
- a와 같이 고유값 두 개가 모두 크면 → 여러 방향으로 변화가 있는 지점. 특징점으로 적합!

	a	b	c
2차 모멘트 행렬	$\mathbf{A} = \begin{pmatrix} 0.522 & -0.199 \\ -0.199 & 0.527 \end{pmatrix}$	$\mathbf{A} = \begin{pmatrix} 0.075 & -0.075 \\ -0.075 & 0.801 \end{pmatrix}$	$\mathbf{A} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
고유값	$\lambda_1=0.7235, \lambda_2=0.3255$	$\lambda_1=0.8087, \lambda_2=0.0673$	$\lambda_1=0.0, \lambda_2=0.0$
특징 가능성 값	$C=0.1925$	$C=0.0237$	$C=0.0$

5.2.2 해리스 코너

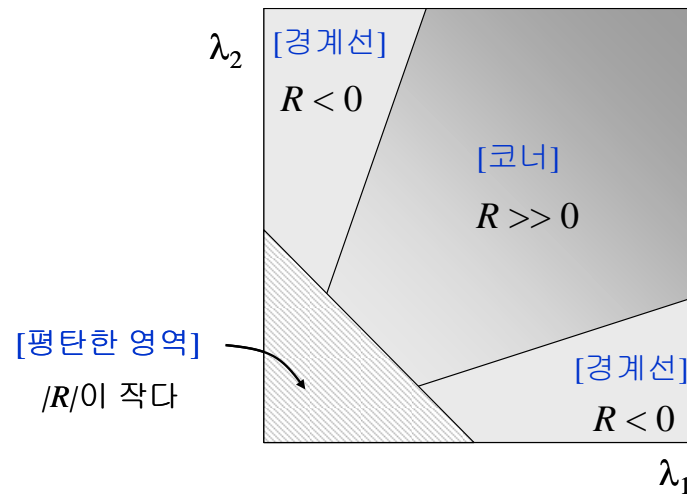
■ 특징 가능성 값 측정

$$C = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \quad (4.8)$$

- 고유값 계산을 피해 속도 향상

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

$$C = \det(\mathbf{A}) - k \times \text{trace}(\mathbf{A})^2 = (pq - r^2) - k(p + q)^2 \quad (4.9)$$



5.2.2 해리스 코너



그림 4-6 해리스 코너

← $C > 0.02$ 인 점을 검출

- 위치 찾기 문제 대두
 - 큰 C 값을 가진 큰 점들이 밀집되어 나타나므로 대표점 선택 필요
- 코너라는 용어가 적절한가?
 - 코너 → 특징점 또는 관심점

5.2 2차 미분을 사용한 방법

■ 헤시안 행렬

$$\mathbf{H} = \begin{pmatrix} d_{yy} & d_{yx} \\ d_{yx} & d_{xx} \end{pmatrix}$$

- 가우시안을 포함한 헤시안 행렬(Gaussian으로 먼저 스므딩 한 후 헤시안)

$$\mathbf{H} = \begin{pmatrix} d_{yy}(\sigma) & d_{yx}(\sigma) \\ d_{yx}(\sigma) & d_{xx}(\sigma) \end{pmatrix} \quad (4.11)$$

$$\text{이때 } d_{st}(\sigma) = \frac{\partial}{\partial t} \left(\frac{\partial}{\partial s} (G(y, x, \sigma) \circledast f(y, x)) \right), s \text{와 } t \text{는 } y \text{ 또는 } x$$

■ 2차 미분에서 특징 가능성 값 측정

- 헤시안의 행렬식(determinant)

$$C = \det(\mathbf{H}) = d_{yy}(\sigma) d_{xx}(\sigma) - d_{yx}(\sigma)^2 \quad (4.12)$$

- 가우시안 라플라시안(LOG)

$$C = \nabla^2 = \text{trace}(\mathbf{H}) = d_{yy}(\sigma) + d_{xx}(\sigma) \quad (4.13)$$

OpenCV 함수

- void cornerHarris(InputArray **src**, OutputArray **dst**, int **blockSize**, int **ksize**, double **k**, int **borderType**=BORDER_DEFAULT)
 - **src** – Input single-channel 8-bit or floating-point image.
 - **dst** – Image to store the Harris detector responses. It has the type CV_32FC1 and the same size as src .
 - **blockSize** – Neighborhood size : 가우시안 필터의 커널 크기
 - **ksize** – Aperture parameter for the [Sobel\(\)](#) operator.(dx, dy연산 커널 크기)
 - **k** – Harris detector free parameter. See the formula below.

Python:

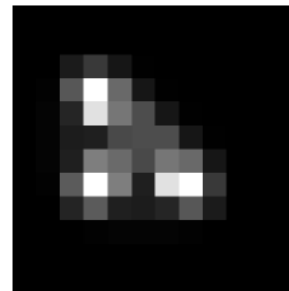
```
cv.cornerHarris( src, blockSize, ksize, k[, dst[, borderType]] ) -> dst
```

5.2 위치 찾기 알고리즘(localization)

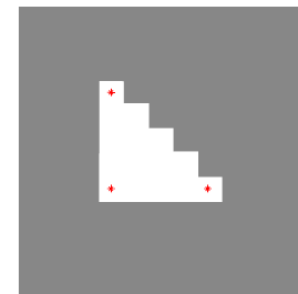
■ 해리스 적용 예

- 큰 값이 밀집되어 나타남 → 해당 영역에 한 점(대표점) 선택 필요

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	-.001	0	0	0	0	0	0	0	0
0	0	.021	.043	.017	-.001	0	0	0	0	0	0
0	-.002	.066	.191	.079	-.017	-.001	0	0	0	0	0
0	-.003	.028	.169	.089	-.058	-.017	-.001	0	0	0	0
0	-.003	-.021	.024	.055	-.058	-.058	-.017	-.001	0	0	0
0	-.003	.023	.095	.080	.055	.089	.079	.017	0	0	0
0	-.002	.068	.192	.095	.024	.169	.191	.043	-.001	0	0
0	0	.028	.068	.023	-.021	.028	.066	.021	0	0	0
0	0	0	-.002	-.003	-.003	-.003	-.002	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



(a) 식 (4.9)로 계산한 특징 가능성 맵(굵게 표시된 부분은 지역 최대점)



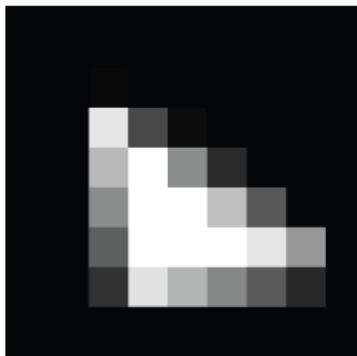
(b) 비최대 억제로 찾은 특징점

그림 4-9 특징 가능성 맵과 특징점 검출

5.2.2 해리스 특징점

■ 이동과 회전에 불변인가?

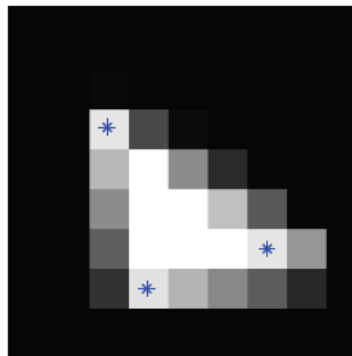
- 이동이나 회전 변환이 발생하여도 같은 지점에서 관심점이 검출되나?
⇒ 해리스 코너점: yes! 검출된다.
- 아래 그림은 삼각형을 10도 회전한 그림임.



(a) 회전한 영상



(b) 가능성 맵



(c) 관심점 검출

그림 4-10 회전한 삼각형에서 관심점 검출

5.2.2 해리스 특징점

■ 스케일에 불변인가?

- 스케일이 변해도 같은 지점에서 관심점이 검출되나?

⇒ 연산자 크기가 고정되어 있어 그렇지 않다. 스케일 불변이 아니다.
스케일 변화에 대처하려면 연산자 크기를 조절하는 기능이 필수적임

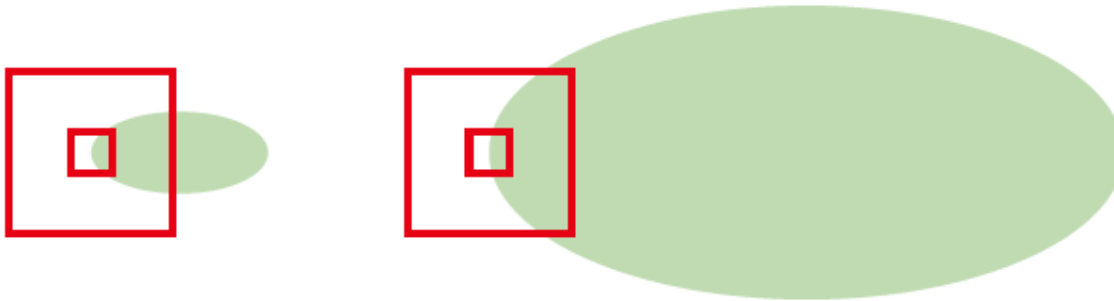


그림 5-6 물체의 크기에 따라 마스크의 크기를 적절하게 정해야 하는 상황

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. ]]
```

	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	^c 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

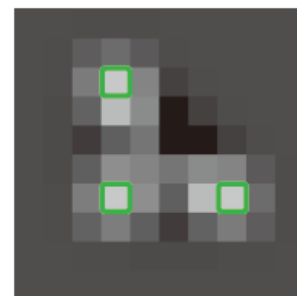


그림 5-5 특징 가능성 맵 C 와 비최대 억제로 찾은 지역 특징점

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([[0,0,0,0,0,0,0,0,0,0],
05               [0,0,0,0,0,0,0,0,0,0],
06               [0,0,0,1,0,0,0,0,0,0],
07               [0,0,0,1,1,0,0,0,0,0],
08               [0,0,0,1,1,1,0,0,0,0],
09               [0,0,0,1,1,1,1,0,0,0],
10               [0,0,0,1,1,1,1,1,0,0],
11               [0,0,0,0,0,0,0,0,0,0],
12               [0,0,0,0,0,0,0,0,0,0],
13               [0,0,0,0,0,0,0,0,0,0]],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
```

Syntax: `cv.getGaussianKernel(ksize, sigma[, ktype])`

Parameters:

- *Ksize:* It is the Aperture size. Ksize value should be odd and positive.
- *sigma:* Sigma is the Gaussian standard deviation. It is computed from ksize as $\sigma = 0.3 * ((ksize - 1) * 0.5 - 1) + 0.8$ if it is non-positive.
- *ktype:* It is the type of filter coefficients. It can be CV_32F or CV_64F.

가우시안 커널 (1차원이므로 ret * ret.T 형태로 사용해야 함)

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  gddy=cv.filter2D(dyy,cv.CV_32F,g)
26  gdxx=cv.filter2D(dxx,cv.CV_32F,g)
27  gdyx=cv.filter2D(dyx,cv.CV_32F,g)
28  C=(gddy*gdxx-gdyx*gdyx)-0.04*(gddy+gdxx)*(gddy+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

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. ] ]
```

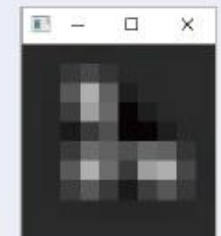
⑨

```
[ [ 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



numpy.outer()

`numpy.outer(a, b, out=None)`

[\[source\]](#)

Compute the outer product of two vectors.

Given two vectors a and b of length M and N , respectively, the outer product [1] is:

```
[[a_0*b_0  a_0*b_1  ...  a_0*b_{N-1}]
 [a_1*b_0      .
 [ ...      .
 [a_{M-1}*b_0      a_{M-1}*b_{N-1} ]]
```

Parameters: a : $(M,)$ *array_like*


First input vector. Input is flattened if not already 1-dimensional.

b : $(N,)$ *array_like*

Second input vector. Input is flattened if not already 1-dimensional.

out : (M, N) *ndarray, optional*

A location where the result is stored

 *New in version 1.9.0.*

Returns: out : (M, N) *ndarray*

$out[i, j] = a[i] * b[j]$