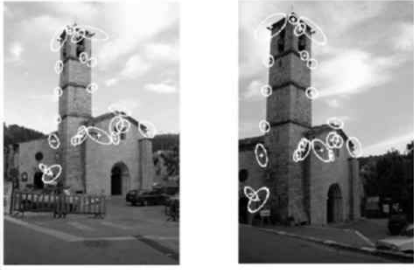



1

### 10.2 코너 검출






- ◆에지나 직선
  - ❖ 영상 구조 파악 및 객체 검출에는 도움이 되지만, 영상 매칭에는 큰 도움이 되지 않음
  - ❖ 에지 강도와 방향 정보만 가지므로 영상 매칭하기엔 정보 부족
- ◆영상 매칭
  - ❖ Vision 작업은 둘 이상의 시점에서 해당 특징점을 찾아야 함.



- ❖ Matching시킬 단위는 고정된 크기의 영상 patch임.



- ❖ 작업 : 두 번째 이미지에서 가장 유사한 patch를 찾음.


?
=





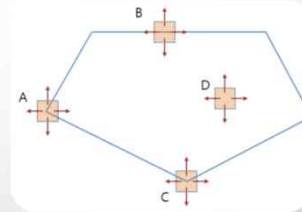
2

## 10.2 코너 검출

### ◆꼭지점 혹은 코너(corner)

- ❖ 직관적으로, 외곽선의 교차점.
- ❖ 일반적으로 시점의 변화에 대해 보다 안정된 특징점들.
- ❖ 직관적으로, 모든 방향으로 점의 이웃에 큰 변화.

=> matching에 좋은 특징점!



### ❖영상의 밝기 변화

- A, C - 모든 방향에서 밝기변화 큼
- B, D - 밝기 변화가 상대적으로 적음

### ◆코너 검출: Basic Idea

- ❖ patch 내에서 밝기변화를 보고 그 점을 쉽게 찾을 수 있어야 함.
- ❖ patch 를 어느 방향으로 이동하든 색상에 큰 밝기변화가 생김.

3

## 10.2 코너 검출

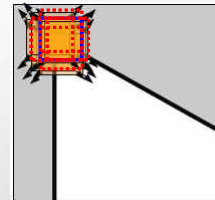
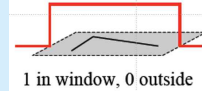
### ◆모라벡(Moravec)

- ❖ 영상 변화량(SSD: Sum of Squared Difference)

$$E(u, v) = \sum_y \sum_x w(x, y) \cdot (I(x+u, y+v) - I(x, y))^2$$

- 현재 화소에서 u, v 방향으로 이동했을 때의 밝기 변화량의 제곱
- (u, v)를 (1, 0), (0, 1), (0, -1), (-1, 0)의 4개 방향으로 한정
  - ✓ 거의 일정한 patch의 경우 0에 가까움.
  - ✓ 아주 독특한 patch의 경우, 값이 커짐.
  - ✓ 따라서 E(u,v)가 큰 patch에 코너 존재 가능성.

patch를 지정하는 window 함수



### ◆문제점

- ❖ 0과 1의 값만 갖는 이진 윈도우 사용으로 노이즈에 취약
- ❖ 4개 방향으로 한정시켰기 때문에 45도 간격의 에지만 고려

4

## 10.2 코너 검출

### ◆해리스(Harris) 검출

- ❖ 이진 윈도우  $\omega(u, v)$  대신에 점진적으로 변화하는 가우시안 마스크  $G(x, y)$  적용

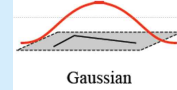
$$E(u, v) = \sum_y \sum_x G(x, y) \cdot (I(x+u, y+v) - I(x, y))^2$$

- ❖ 모든 방향에서 검출할 수 있도록 미분 도입

$$I(x+u, y+v) \cong I(x, y) + v d_y(x, y) + u d_x(x, y)$$

$$E(u, v) \cong \sum_y \sum_x G(x, y) \cdot (v d_y(x, y) + u d_x(x, y))^2$$

patch를 지정하는 window 함수



$$\begin{aligned} E(u, v) &\cong \sum_y \sum_x G(x, y) \cdot (v d_y + u d_x)^2 \\ &= \sum_y \sum_x G(x, y) \cdot (v^2 d_y^2 + u^2 d_x^2 + 2v u d_x d_y) \\ &= \sum_y \sum_x G(x, y) \cdot (u \ v) \begin{pmatrix} d_x^2 & d_x d_y \\ d_x d_y & d_y^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \\ &= (u \ v) M \begin{pmatrix} u \\ v \end{pmatrix}, \quad M = \sum_y \sum_x G(x, y) \begin{pmatrix} d_x^2 & d_x d_y \\ d_x d_y & d_y^2 \end{pmatrix} \end{aligned}$$

특징 가능성을 직접 계산하는 대신 행렬 M의 식으로 정리

5

## 10.2 코너 검출

### ❖ 응답 함수(R) 계산

- ① 행렬 M에서 고유벡터를 구하면 경계선 방향에 수직인 벡터 두 개 얻음  
행렬 M의 고유값( $\lambda_1, \lambda_2$ )으로 코너 응답 함수 계산

$$R = \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2$$

k는 경험적 상수로써  
0.04 ~ 0.06 사이의 값

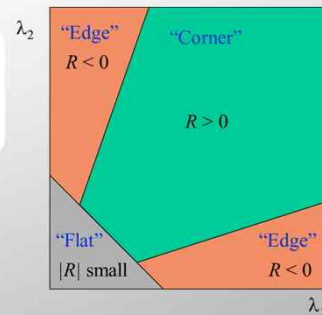
- ② 고유값 대신 행렬식(det)과 대각합(trace)을 통해 코너 응답 함수로 이용

- 고유값 계산은 고유값 분해의 복잡한 과정 거침

$$M = \begin{pmatrix} d_x^2 & d_x d_y \\ d_x d_y & d_y^2 \end{pmatrix} = \begin{pmatrix} a & c \\ c & b \end{pmatrix}$$

$$R = \det(M) - k \cdot \text{trace}(M)^2 = (ab - c^2) - k \cdot (a + b)^2$$

- $R > 0$ 면 코너점,  
 $R < 0$ 면 edge,  
 $|R|$ 이 매우 작은 값이면, R 부호에 관계없이 flat으로 판단

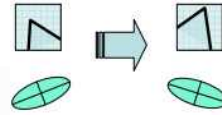


6

## Harris Corner 특성

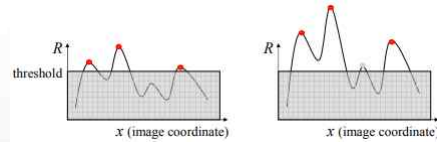
### ◆회전 불변성

- ❖ 타원은 회전하지만 그 모양은 동일하게 유지
- ❖ 코너 응답 R은 이미지 회전에 불변

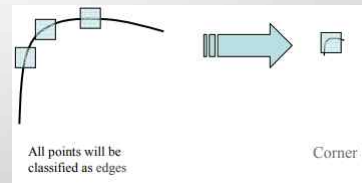


### ◆어파인 색상 변화에 대한 부분 불변성

- ❖ 미분 성분만 사용  $I \rightarrow I + b$  색상 변화 불변성



### ◆그러나 : 이미지 스케일에는 영향을 받음!!!



7

## 10.2 코너 검출

### ◆전체 과정

1. 소벨 마스크로 미분 행렬 계산 ( $dx, dy$ )
2. 미분 행렬의 곱 계산 ( $dx^2, dy^2, dxy$ )
3. 곱 행렬에 가우시안 마스크 적용
4. 코너 응답함수  $C = \det(M) - k \cdot \text{trace}(M)^2$  계산
5. 비최대치 억제

8

## 10.2 코너 검출

예제 10.2.1 헤리스 코너 검출 - 03.harris\_detect.py

```

01 import numpy as np, cv2
02 from Common.utils import put_string
03
04 def cornerHarris(image, ksize, k):
05     dx = cv2.Sobel(image, cv2.CV_32F, 1, 0, ksize)
06     dy = cv2.Sobel(image, cv2.CV_32F, 0, 1, ksize)
07
08     a = cv2.GaussianBlur(dx * dx, (5, 5), 0)
09     b = cv2.GaussianBlur(dy * dy, (5, 5), 0)
10     c = cv2.GaussianBlur(dx * dy, (5, 5), 0)
11
12     corner = (a * b - c**2) - k * (a + b)**2
13     return corner
14
15 def drawCorner(corner, image, thresh):
16     cnt = 0
17     corner = cv2.normalize(corner, 0, 100, cv2.NORM_MINMAX)
18     corners = []
19     for i in range(1, corner.shape[0]-1):
20         for j in range(1, corner.shape[1]-1):
21             neighbor = corner[i-1:i+2, j-1:j+2].flatten()
22             max = np.max(neighbor[1::2])
23             if thresh < corner[i, j] > max: corners.append((j, i))
24
25     for pt in corners:
26         cv2.circle(image, pt, 3, (0, 255, 0), -1)
27     print("임계값: %2d, 코너 개수: %2d" % (thresh, len(corners)))
28     return image

```

x방향 마스크

y방향 마스크

가우시안 블러링 수행

곱 행렬 계산 ( $dx^2, dy^2, dxy$ )

코너 응답 함수-행렬 연산 적용

임계값 이상 코너 표시

헤리스 검출 행렬 0~100 정규화

비최대치 억제

이웃 화소 가져옴

상하좌우 값만

코너 확정 좌표 저장

반지름 3인 녹색 원 표시

코너 확정 좌표 순회

좌표 표시

9

## 10.2 코너 검출

```

30 def onCornerHarris(thresh):
31     img1 = drawCorner(corner1, np.copy(image), thresh)
32     img2 = drawCorner(corner2, np.copy(image), thresh)
33
34     put_string(img1, "USER", (10, 30), "")
35     put_string(img2, "OpenCV", (10, 30), "")
36     dst = cv2.repeat(img1, 1, 2)
37     dst[:, img1.shape[1]:, :] = img2
38     cv2.imshow("harris detect", dst)
39
40 image = cv2.imread("images/harris.jpg", cv2.IMREAD_COLOR)
41 if image is None: raise Exception("영상파일 읽기 에러")
42
43 blockSize = 4
44 apertureSize = 3
45 k = 0.04
46 thresh = 2
47 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
48 corner1 = cornerHarris(gray, apertureSize, k)
49 corner2 = cv2.cornerHarris(gray, blockSize, apertureSize, k)
50
51 onCornerHarris(thresh)
52 cv2.createTrackbar("Threshold", "harris detect", thresh, 20, onCornerHarris)
53 cv2.waitKey(0)

```

트랙바 콜백 함수 - 이벤트시마다 영상에 코너 표시

트랙바 콜백 함수

영상에 문자표시 함수 호출

두 개 영상을 한 윈도우 표시

오른쪽 영역

이웃 화소 범위

소벨 마스크 크기

코너 응답 임계값

사용자 정의 함수

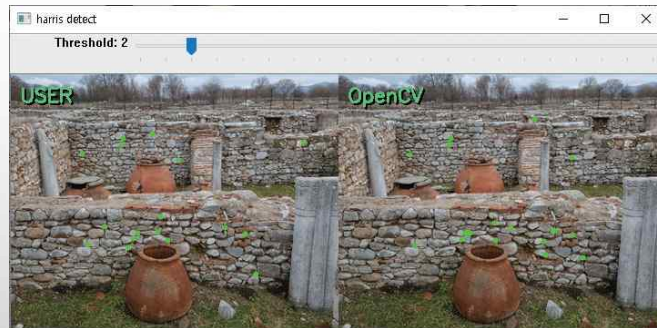
OpenCV 제공 함수

트랙바 콜백 함수 등록

10

## 10.2 코너 검출

### ◆ 실행결과



11

## 단원 요약

- ◆ 영상 처리에서 중요한 특징 정보로 사용되는 코너는 영상에서 경계가 만나는 지점의 특정한 모양을 갖는 곳을 가리킨다. 이 코너 정보들 중에서 영상의 왜곡에도 불변하는 특징을 가진 지점들이 영상 매칭에 유용하게 사용될 수 있다.
- ◆ 해리스 코너 검출 방법은 영상의 평행이동, 회전 변환에는 불변(invariant)하는 특징이 있고, 어파인(affine) 변환이나 조명(illumination) 변화에도 어느 정도는 강인성이 있다.

12