Open_cv06

영상 특징 처리 _ feature_detection

ICT , DS&DE Y-A , KIM

특징 추출 - Interest Operator, Scale Space, Wavelet, SIFT

대조 방법 - 템플릿 매칭, DP 매칭, 그래프 매칭

특징 추출 이란?

화상에서 정보 추출은 종종 특정 부분에 주목하여 나머지 부분을 무시하는 것이 필요함

물체 인식에서는 배경을 무시하고 전경의 물체에만 주목

목적에 따라 화상에서 관심 있는 정보만을 추출하고 처리해야 함

특징점과 가장자리 검출이란?

화상 인식 초기부터 밝기 또는 색상이 급격히 변화하는 지점에서 가장자리를 검출하는 방법 제안

가장자리는 물체의 윤곽에서 감지되지만, 1차원 곡선으로 표현됨

위치를 보다 정확하게 결정하기 위해 점으로 나타나는 특징 필요

특징점 추출

- 특징점: 이미지 내에서 휘도 값 또는 색이 주변 화소와 구별되어 정확한 위치 결정이 가능한 점
- Scale Space 이론: 이미지를 스케일로 파라미터화하여 모든 스케일을 동시에 처리
- 특징점은 이미지의 스케일 변화에도 안정적이어야 함
- Wavelet: 이미지를 유한 길이의 파형으로 표현, 공간 및 주파수 양 영역에서 국소 특성화 가능
- SIFT (Scale-Invariant Feature Transform) : 스케일에 불변한 특성을 가진 특징점을 식별하고 추출함

이미지 패턴 인식 기법

- 템플릿 매칭: 이미지 일부를 패턴으로 사용해 이미지 부분과 일치
- DP 매칭: 1차원적 탄성 매칭
- 그래프 매칭: 특징을 정점으로, 특징 간의 관계를 변으로 표현하여 그래프 간 대응을 실시

Interest Operator와 이미지 특징점 추출

코너 검출기의 역할

- 이미지에서 모서리 부분을 정확하게 식별
- 이미지의 세로 및 가로 방향 모두에서 큰 변화를 감지

주요 특징 점 추출 오퍼레이터

- **SUSAN 오퍼레이터:** 화상의 휘도 값 차가 작은 점의 수를 세는 방법
- **Harris 오퍼레이터:** 윈도우 영역 내의 화상을 미소 시프트시킬 때 휘도 값 차의 제곱 합이 커지는 점을 검출
- **Shi와 Tomasi의 방법:** Harris 오퍼레이터와 유사한 방법을 사용

특징 점 추출 기반

1. Moravec`s Corner Detector -> 1977년 : Moravec이 코너점 찾는 방법을 시작(1977)
2. **Harris Corner Detector** -> 1988년 : 해리스코너 방법으로 Harris/Plessey Corner Detector 제안됨
3. **Shi and Tomasi Corner Detector**
4. SUSAN(Smallest Univalue Segment Assimilating Nucleus) Corner Detector
5. Trajkovic and Hedley Corner Detector
6. Wang and Brady Corner Detector
7. High-Speed Corner Detector
8. **FAST(Accelerated Segment Test의 특징) Corner Detector**

영역 기반 접근

1. Harris Laplace Detector
2. Harris Affine Detector
3. Hessian Affine Detector
4. EBR(Edge-Based Region) Detector
5. IBR(Intensity-Based Region) Detector
6. MSER(Maximally Stable Extremal Region) Detector
7. Salient Region Detector

모서리 감지_ Harris Corner Detector

정의:

- Harris Corner Detection은 이미지 내의 모서리를 식별하는 알고리즘이다.
- 모서리는 그라디언트의 강도 변화가 큰 영역으로 판단된다.
- 특징점을 검출하는 방법 중 하나로, 이미지의 각 점에서 휘도 값의 변화를 측정한다.
- 이미지 내의 특정 점을 중심으로 윈도우 영역 내에서 미세하게 이미지를 이동시켰을 때 발생하는 휘도 값 차이의 제곱합 (SSD)을 계산한다

특징:

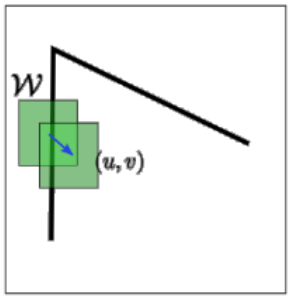
- 모든 차원과 방향에서의 강도 변화를 분석한다.
- 이미지 특징 추출에 활용되며, 다른 이미지의 특징과 매칭될 수 있다.

활용:

Harris Corner는 정확한 정보 추출에 사용되며, 이미지 인식, 객체 추적 등 다양한 분야에서 응용된다.

모서리 감지_ Harris Corner Detector

변화량 $E(u,v)$ = 이동벡터 의 계산 과정:



입력 이미지의 휘도 값 산출	주변 패치 창의 선택	평행 이동된 패치의 계산	SSD 계산
• 입력 이미지 I 에서 좌표 (x,y)에서의 휘도 값을 (x,y)로 계산	• (x,y) 주변의 패치 (예: 6×6 패치)를 선택하고, 이를 벡터 (x,y)로 표현	• 주목점 주변의 패치를 (u,v)만큼 평행 이동하여 새로운 패치를 계산 $(u, v$ = 이동의 크기, 방향)	• 원래의 패치와 평행 이동한 패치 사이의 SSD를 계산하여 변화량 $E(u,v)$를 산출. • SSD는 두 패치 간의 픽셀 값 차이의 제곱의 합으로 계산됨.

$$E(u,v) = \sum_W (I(x,y) - I(x+u, y+v))^2$$

100,100,200,200 사각형을 200,200, 400,400으로 이동시킨 변화량은 ?

- 이동 벡터 (u,v)는 (100, 100) -> W는 패치의 영역
- 만일 두 패치의 픽셀값이 동일하다면 SSD와 변화량 E(u,v)는 0이 된다.
- 패치의 이동이 있더라도 해당 패치 내의 픽셀 값이 동일하므로 픽셀 값의 차이가 없기 때문

모서리 감지_ Harris Corner Detector

- Harris 코너 검출기는 Moravec 코너 검출기의 한계를 극복하기 위해 개발, 기본 아이디어는 이미지의 평행 이동 량이 작을 때 이미지의 편 미분 값을 사용하는 것으로 8 방향으로 평행 이동한 패치를 사용하는 Moravec 방법보다 정확도가 높다.
- 이미지의 편 미분 값을 사용하고, 에너지 함수를 이미지의 2차 모멘트 행렬로 근사하여, 고유 값을 사용해 코너를 안정적으로 식별한다

(1) $$E(u,v) = \sum_{(x,y)\in W} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

(1) 평행 이동량 (u, v)이 매우 작다고 가정하고,
(u,v)에 대한 조건 $u^2+v^2=1$ 추가.

(2) $$I(x+u,y+v) \approx I(x,y) + I_x(x,y)u + I_y(x,y)v$$

(2) 테일러 급수에 의한 함수 근사 (계산의 복잡성 줄임)

(3) $$E(u,v) \approx \sum_{(x,y)\in W} w(x,y)[I_x(x,y)u + I_y(x,y)v]^2$$

(3) 테일러 급수에 의한 E(u,v) 전체를 근사

(4) $$E(x,y) = [u \quad v]\mathbf{M}\begin{bmatrix}u\\v\end{bmatrix}$$

(4) (3)의 근사는 행렬 M을 사용하여 재작성

(5) $$\mathbf{M} = \sum_{(x,y)\in W}\begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y)\end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2(x,y) & \sum_{(x,y)\in W} I_x(x,y)I_y(x,y) \\ \sum_{(x,y)\in W} I_x(x,y)I_y(x,y) & \sum_{(x,y)\in W} I_y^2(x,y)\end{bmatrix}$$

(5) 1차 미분 화상과 2 차 미분 화상의 화소 값으로 부터 행렬 M의 원소 값을 계산

모서리 감지_ Harris Corner Detector 공식정리

$$M = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det(H) - \kappa\,\mathrm{trace}^2(H)$$ λ1과 λ2는 H(p)의 고유 값, κ는 상수

① *M*은 행렬 *H*(*p*)의 에서 계산되는 특징량, 이미지 내의 코너를 식별하는 데 사용된다.

② *H*(*p*)는 특정 픽셀 *p* 주변의 휘도 값 분포와 그래디언트 정보를 나타내며, 이 정보를 바탕으로 *M* 값을 계산하여 코너를 감지한다.

③ 이미지의 그래디언트의 x와 y 방향의 곱을 이용해 계산되며, 그래디언트의 방향과 크기 정보

행렬 M 고유벡터는 이미지 패치의 구조와 방향을 나타낸다

- 두 고유치 모두 큰 값을 가질 때: 이 경우 해당 영역은 이미지의 '코너'에 해당하며 코너는 두 엣지가 교차하는 지점에서 발견된다.
- 두 고유치 중 하나만 큰 값을 가질 때: 이 경우 해당 영역은 '엣지'에 해당하며. 엣지는 이미지에서 밝기나 색상이 급격하게 변하는 부분을 말한다.
- 두 고유치 모두 작은 값을 가질 때: 이 경우 해당 영역은 '평탄'한 영역으로, 특별한 텍스처나 패턴이 없는 부분을 말한다.

고유 값이란 주어진 행렬 A에 대한 특성 방정식 도출

$$\det(A - \lambda I) = 0$$

- A : 주어진 행렬
- λ : 고유값
- I : 단위행렬

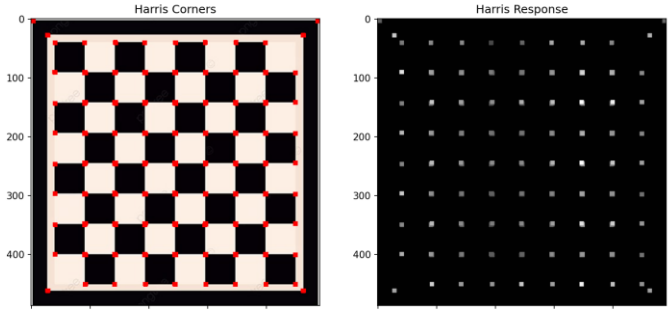
$$A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \rightarrow \det \begin{bmatrix} 2-\lambda & 1 \\ 1 & 3-\lambda \end{bmatrix} = 0 \rightarrow \begin{matrix} (2-\lambda)(3-\lambda) - (1)(1) = 0 \\ \lambda^2 - 5\lambda + 5 = 0 \end{matrix}$$

$$\rightarrow \lambda = \frac{5 \pm \sqrt{5^2 - 4 \cdot 1 \cdot 5}}{2 \cdot 1} \rightarrow \lambda = \frac{5 \pm \sqrt{25 - 20}}{2} = \frac{5 \pm \sqrt{5}}{2}$$

$$\lambda_1 = \frac{5 + \sqrt{5}}{2}, \quad \lambda_2 = \frac{5 - \sqrt{5}}{2}$$

모서리 감지_ Harris Corner Detector

cv2.cornerHarris(src, dest, blockSize, kSize, freeParameter, borderType)



a_cornerHarris.py

```
dst = cv2.cornerHarris(gray, 2, 3, 0.05)
dst = cv2.dilate(dst, None, iterations=3)

img[dst > 0.01 * dst.max()] = [0, 0, 255]
```

-1	0	1
-2	0	2
-1	0	1

[수평 커널]

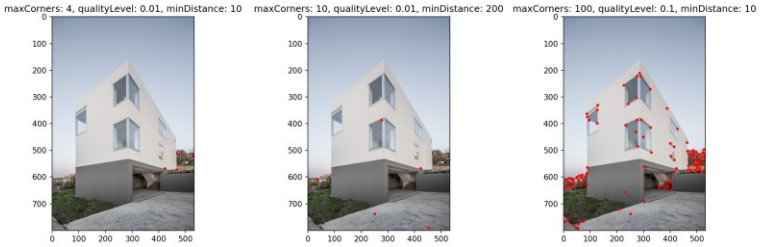
-1	-2	-1
0	0	0
1	2	1

[수직 커널]

$$\theta = \arctan 2(G_y, G_x)$$

모서리 감지 _Shi and Tomasi Corner Detector

- Shi-Tomasi 코너 감지는 J.Shi와 C.Tomasi의 논문 ' *Good Feature to Track* '에 게재
- $M = \min(\lambda_1, \lambda_2)$ 고유 값 중 작은 값 만을 사용하여 코너를 판별
- 이미지의 각 픽셀에서 2차 모멘트 행렬 $H(p)$를 계산 한 다음 고유 값 $\lambda 1$와 $\lambda 2$를 찾아서 작은 값이 임계 보다 크면 해당 픽셀로 코너 판별한다.

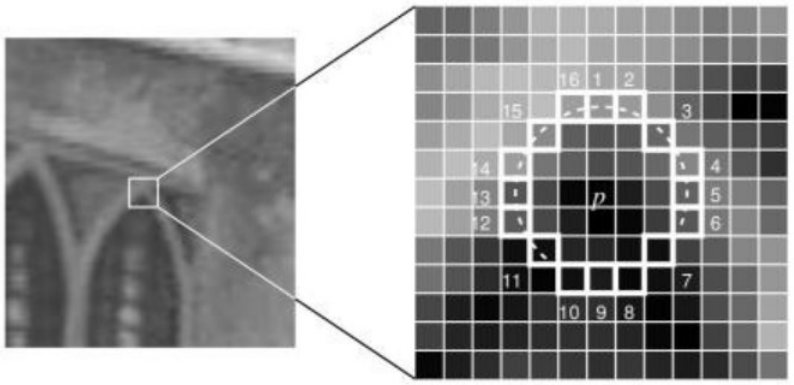


d_goodFeaturesToTrack.py

```
configs = [
    {"maxCorners": 4, "qualityLevel": 0.01, "minDistance": 10},
    {"maxCorners": 10, "qualityLevel": 0.01, "minDistance": 200},
    {"maxCorners": 100, "qualityLevel": 0.1, "minDistance": 10}
]

plt.figure(figsize=(15, 5))

for i, config in enumerate(configs):
    temp_img = img.copy()
    corners = cv2.goodFeaturesToTrack(
        gray,
        maxCorners=config['maxCorners'],
        qualityLevel=config['qualityLevel'],
        minDistance=config['minDistance']
    )
```

cv2.goodFeaturesToTrack(대상이미지, 탐지코너 최대 개수, 품질 수준, 유클리드 최소 거리)

모서리 감지 _FAST(Features from Accelerated Segment Test)



https://arxiv.org/pdf/0810.2434.pdf

$$I(n) > I(p) + T \qquad I(n) < I(p) - T$$

$I(n)$은 주변 픽셀 n의 강도
$I(p)$은 중심 픽셀 p의 강도
T는 설정된 임계값

1. **픽셀 선택 :**이미지에서 특정 픽셀(p)를 선택
2. **밝기 임계값 :**중심 픽셀의 밝기와 비교하기 위한 임계값을 설정
3. **주변 픽셀 검사 :** 중심 픽셀 주변의 16개 픽셀을 검사하여, 이들 중 일부가 중심 픽셀보다 훨씬 밝거나 어두운지 확인
4. **코너 판정:** 주변 픽셀 중 연속된 12개 이상이 중심 픽셀보다 훨씬 밝거나 어두울 경우, 해당 픽셀을 코너로 판단
5. **최종 검증 :** 추가적인 알고리즘과 테스트를 사용하여 더 정확하고 안정적인 코너를 식별

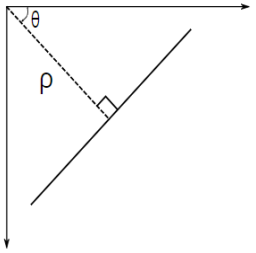
라인감지 _허프 변환(Hough Transform)

허프 변환

1. 이미지 처리에서 형태를 감지하는 기법

2. 특히, 노이즈에 강하고 불완전한 정보로부터 직선, 원, 또는 다른 간단한 형태를 감지하는데 유용

라인 감지 원리



1. 이미지의 각 픽셀에 대해 가능한 모든 직선의 방정식을 생성

2. 직선의 파라미터를 누적하여 주어진 임계 값을 초과하는 경우 라인으로 감지

3. $y=mx+c$ 방정식을 극좌표로 변환하여 $\rho=x\cos\theta+y\sin\theta$로 표현 / 각 θ에 대해 ρ 값을 계산하고 누적

허프 공간에서의 라인 감지

1. 허프 공간에서 누적된 값이 임계값을 초과할 때 해당 파라미터 (ρ, θ)로 라인 감지

2. 결과적으로 이미지에서 라인의 위치와 방향을 식별

라인감지 _허프 변환(Hough Transform) HoughLine

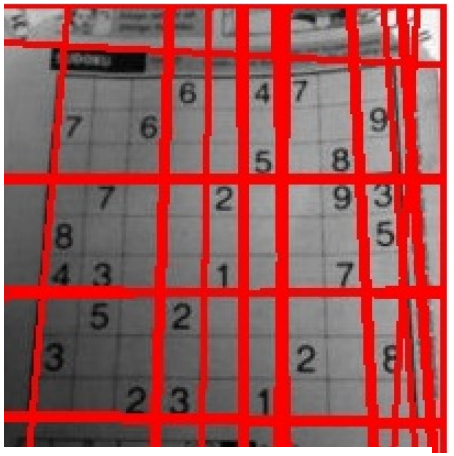
```
import cv2
import numpy as np

img = cv2.imread('dave.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150, apertureSize=3)

lines = cv2.HoughLines(edges, 1, np.pi/180, 100)
if lines is not None:
    for line in lines:
        for rho, theta in line:
            a = np.cos(theta)
            b = np.sin(theta)
            x0 = a*rho
            y0 = b*rho
            x1 = int(x0 + 1000*(-b))
            y1 = int(y0 + 1000*(a))
            x2 = int(x0 - 1000*(-b))
            y2 = int(y0 - 1000*(a))
            cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)

cv2.imwrite('houghlines.jpg', img)
```

허프 변환을 사용하여 에지 이미지에서 직선을 검출

1과 **np.pi/180**은 ρ와 θ의 해상도이며, 1**00**은 직선 검출을 위한 임곗값

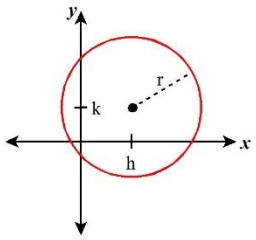
$r = x\cos(\theta) + y\sin(\theta)$

행은 r, 열은 (θ)세타



g_HoughLine.py

원감지 _허프 변환(Hough Transform) HoughCircles



$$(x - h)^2 + (y - k)^2 = r^2$$

h_HoughCircles.py

1. 매개변수

a: 원의 중심 x 좌표 (범위: 1 ~ 행의 수)

b: 원의 중심 y 좌표 (범위: 1 ~ 열의 수)

r: 원의 반지름 (범위: 1 ~ √(행^2 + 열^2))

2. 방법

3차원 누산기 행렬을 사용하여 가능한 a, b, r 값들을 저장

이미지 내 각 점에 대하여 가능한 원들을 투표 방식으로 검출

3. 원 감지

누산기 행렬에서 가장 높은 투표를 받은 매개 변수 (a, b, r)을 선택

선택된 매개 변수를 사용해 원을 정확하게 감지 및 표시

```
import cv2
import numpy as np


image = cv2.imread('cat_with_glasses.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

gray = cv2.medianBlur(gray, 7)
circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
                           param1=100, param2=30, minRadius=70, maxRadius=100)


if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0, :]:
        cv2.circle(image, (i[0], i[1]), i[2], (0, 255, 0), 2)


cv2.imshow('Glasses Detection', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```