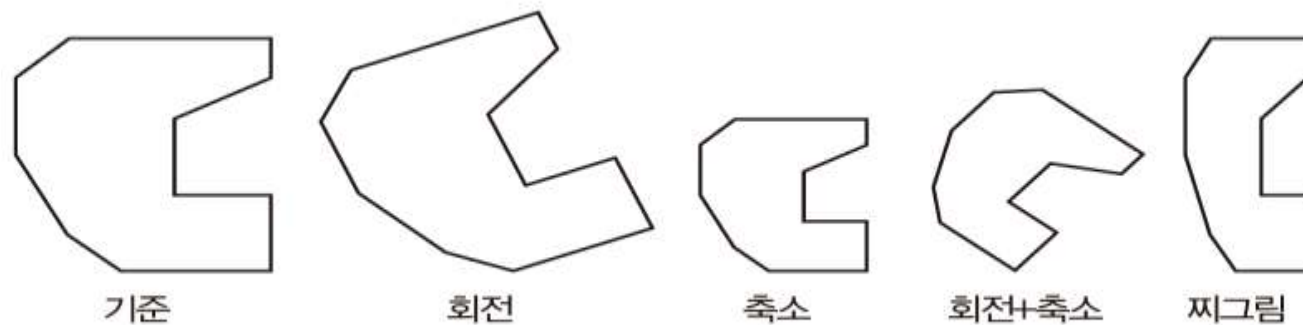


4.6 영역 특징

■ 영역의 레이블링과 기하 변환 예시



(a) 영역의 레이블링



(b) 영역의 기하 변환

그림 4-17 영역의 레이블링과 기하 변환

4.6 영역 특징

■ 높은 분별력

■ 특징의 불변성과 등변성

- 변환을 해도 값이 변하지 않으면 불변성_{invariant}이 있는 특징
 - 예) 성별이라는 특징은 나이에 불변. 근력은 나이에 불변이 아님
 - 예) 면적은 회전에 불변이지만 축소에는 불변이 아님. 주축은 회전에 불변이 아니지만 축소에는 불변
- 변환에 따라 값이 따라 변하면 등변성_{equivariant}이 있는 특징
 - 예) 면적은 축소에 등변이지만 회전에는 등변이 아님
- 명암 변화에 둔감한 광도 불변성도 중요

■ 과업에 따라 특징을 선택하는 일이 중요

- 예) 로봇이 물체를 인식한 다음 물체를 집어야 한다면 회전에 등변인 특징을 사용해야 함

4.6 영역 기술자

- 다음 단계 처리를 위한 추가적인 정보 추출(표현 및 묘사)
 - 영역 표현(representation)
 - 영역을 외부 특성(경계)으로 표현: 모양 특성을 중시=>경계기술자
 - 영역을 내부 특성(구성 화소)로 표현: 영역 특성 중시=>영역기술자
 - 영역 묘사 (description)
 - 경계: 길이, 모멘트, 연결 직선 방향, 오목한 곳의 수 등
 - 내부특성: 면적, 원형도, 모멘트, 칼라, 텍스처 등
- 영역기술자: 모멘트, 텍스처
- 경계기술자
 - 경계상자(bounding box)
 - 가장 작은 둘러싼 원(minimum enclosing circle)
 - 성분의 외곽선에 대한 다각형 근사(polygonal approximation)
 - 볼록 껍질(convex hull)

4.6.1 모멘트와 모양 특징

■ 영역 R 의 이진 영역의 $(q+p)$ 차 모멘트

- 모멘트는 화소의 값들이 분포하는 '모양'(shape)을 정량적으로 측정하는 방법
 - 0차 모멘트는 총 질량이 되고, 1차 모멘트를 총 질량으로 나눈 것은 '질량 중심'(center of mass)이 되며, 2차 모멘트는 '관성 모멘트'(moment of inertia)이다.

$$m_{qp} = \sum_{(y,x) \in R} y^q x^p$$

$$\text{면적 } a = m_{00} \qquad \text{중점 } (\bar{y}, \bar{x}) = \left(\frac{m_{10}}{a}, \frac{m_{01}}{a} \right)$$

■ 영역 R 의 이진 영역의 중심 모멘트

$$\mu_{qp} = \sum_{(y,x) \in R} (y - \bar{y})^q (x - \bar{x})^p$$

$$\text{열 분산 } v_{cc} = \frac{\mu_{20}}{a} \qquad \text{행 분산 } v_{rr} = \frac{\mu_{02}}{a} \qquad \text{혼합 분산 } v_{rc} = \frac{\mu_{11}}{a}$$

4.6 모멘트 모멘트와 모양 특징

- 이진 영역의 크기(스케일) 불변 모멘트

$$\eta_{qp} = \frac{\mu_{qp}}{\mu_{00}^{\left(\frac{q+p}{2}+1\right)}}$$

- 이진 영역의 회전 불변 모멘트 [Hu62]

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$







$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\ (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Hu moments

Let's look at an example. In the table below we have 6 images and their Hu Moments.

id	Image	H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	H[6]
K0		2.78871	6.50638	9.44249	9.84018	-19.593	-13.1205	19.6797
S0		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S1		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S2		2.65884	5.7358	9.66822	10.7427	-20.9914	-13.8694	21.3202
S3		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	21.8214
S4		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	-21.8214

As you can see, the image K0.png is simply the letter K, and S0.png is the letter S. Next, we have moved the letter S in S1.png, and moved + scaled it in S2.png. We added some rotation to make S3.png and further flipped the image to make S4.png.

모멘트

■ 명암 영역의 모멘트

$$m_{qp} = \sum_{(y,x) \in R} y^q x^p \underbrace{f(y,x)}_{\text{값기}}$$

$$\text{중점 } (\bar{y}, \bar{x}) = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right)$$

$$\mu_{qp} = \sum_{(y,x) \in R} (y - \bar{y})^q (x - \bar{x})^p \underbrace{f(y,x)}_{//}$$

모양 특징

■ 영역 R 의 여러 가지 모양 특징

$$\text{면적 } a = \sum_{(y,x) \in R} 1$$

$$\text{둘레 } p = n_{\text{even}} + n_{\text{odd}} \sqrt{2}$$

이때 n_{even} = 짝수 체인의 개수

n_{odd} = 홀수 체인의 개수

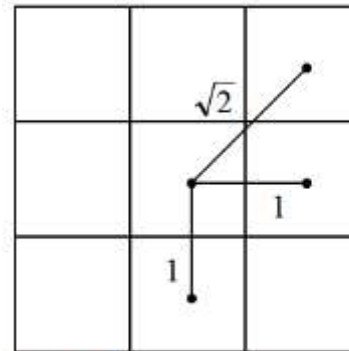


그림 6-9 이웃 화소까지의 거리

$$\text{둥근 정도 } r = \frac{4\pi a}{p^2}$$

$$\text{길쭉한 정도 } e = \frac{a}{w^2} \quad w \text{는 영역의 두께: 침식의 횟수} \times 2$$

$$\text{주축의 방향 } \theta = \frac{1}{2} \arctan \left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right)$$

모양

예제 6-1 특징 기술자 추출

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

그림 6-10 세 개의 영역

영역1의 특징 기술자

- 면적 $a = 20$
- 중점 $(\bar{y}, \bar{x}) = \left(\frac{1}{20} \sum_{(y,x) \in R} y, \frac{1}{20} \sum_{(y,x) \in R} x \right) = (3.05, 2.7)$
- 행 분산 $v_{rr} = \frac{1}{20} \sum_{(y,x) \in R} (x - 2.7)^2 = 3.01$
- 열 분산 $v_{cc} = \frac{1}{20} \sum_{(y,x) \in R} (y - 3.05)^2 = 1.848$
- 혼합 분산 $v_{rc} = \frac{1}{20} \sum_{(y,x) \in R} (y - 3.05)(x - 2.7) = -1.135$
- 둘레 $p = 10 + 6\sqrt{2} = 18.485$
- 둥근 정도 $r = \frac{4\pi \times 20}{18.485^2} = 0.736$

	면적 a	중점 (\bar{y}, \bar{x})	행 분산 v_{rr}	열 분산 v_{cc}	혼합 분산 v_{rc}	둘레 p	둥근 정도 r
영역1	20	(2.7, 3.05)	3.01	1.848	-1.135	18.485	0.736
영역2	12	(4.5, 8.5)	0.917	0.917	0.0	9.657	1.617
영역3	18	(9.5, 6)	0.25	6.667	0.0	18	0.698

성분의 모양 기술자 계산

■ 경계기술자 : 경계 상자

- Rect boundingRect(InputArray **points**)
 - Input 2D point set, stored in std::vector or Mat.
 - The function calculates and returns the minimal up-right bounding **rectangle** for the specified point set.
- void rectangle(Mat& **img**, Point **pt1**, Point **pt2**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, int **shift**=0)
- void rectangle(Mat& **img**, Rect **rec**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, int **shift**=0)
 - **thickness** – Thickness of lines that make up the rectangle. Negative values, like CV_FILLED , mean that the function has to draw a filled rectangle.

```
cv::Rect r0 = cv::boundingRect(contours[0]);  
cv::rectangle(result, r0, 0, 2);
```

Python:

```
cv.boundingRect( array ) -> retval
```

■ 경계기술자 : 가장 작은 둘러싼 원

- void minEnclosingCircle(InputArray **points**, Point2f& **center**, float& **radius**)
 - **points** – Input vector of 2D points, stored in std::vector<> or Mat
 - **center** – Output center of the circle.
 - **radius** – Output radius of the circle.
 - The function finds the minimal enclosing circle of a 2D point set using an iterative algorithm

```
float radius;  
cv::Point2f center;  
cv::minEnclosingCircle(contours[1], center, radius);  
cv::circle(result, center, static_cast<int>(radius), cv::Scalar(0), 2);
```

Python:

```
cv.minEnclosingCircle( points ) -> center, radius
```

■ 경계기술자 : 근사 다각형

- void approxPolyDP(InputArray **curve**, OutputArray **approxCurve**, double **epsilon**, bool **closed**)
 - **curve** – Input vector of a 2D point stored in std::vector or Mat
 - **approxCurve** – Result of the approximation. The type should match the type of the input curve. In case of C interface the approximated curve is stored in the memory storage and pointer to it is returned.
 - **epsilon** – Parameter specifying the approximation accuracy. This is the maximum distance between the original curve and its approximation.
 - **closed** – If true, the approximated curve is closed (its first and last vertices are connected). Otherwise, it is not closed.
- void polylines(InputOutputArray **img**, InputArrayOfArrays **pts**, bool **isClosed**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, int **shift**=0)[1](#)

```
vector<cv::Point> poly;  
cv::approxPolyDP(contours[2], poly, 5, true);  
cv::polylines(result, poly, true, 0, 2);
```

Python:

```
cv.approxPolyDP( curve, epsilon, closed[, approxCurve] ) -> approxCurve
```

■ 경계기술자 : 볼록 껍질

- void convexHull(InputArray **points**, OutputArray **hull**, bool **clockwise**=false, bool **returnPoints**=true)
 - **points** – Input 2D point set, stored in std::vector or Mat.
 - **hull** – Output convex hull. It is either an integer vector of indices or vector of points. In the first case, the hull elements are 0-based indices of the convex hull points in the original array (since the set of convex hull points is a subset of the original point set). In the second case, hull elements are the convex hull points themselves.
 - **clockwise** – Orientation flag. If it is true, the output convex hull is oriented clockwise. Otherwise, it is oriented **counter-clockwise**. The assumed coordinate system has its X axis pointing to the right, and its Y axis pointing upwards.
 - **returnPoints** – Operation flag. In case of a matrix, when the flag is true, the function returns **convex hull points**.

```
vector<cv::Point> hull;  
cv::convexHull(contours[3], hull);  
cv::polylines(result, hull, true, 0, 2);
```

Python:

```
cv.convexHull( points[, hull[, clockwise[, returnPoints]]] ) -> hull
```

■ 경계기술자 : 모멘트

- Moments moments(InputArray **array**, bool **binaryImage**=false)
 - **array** – Raster image (single-channel, 8-bit or floating-point 2D array) or an array (or) of 2D points (Point or Point2f).
 - **binaryImage** – If it is true, all non-zero image pixels are treated as 1's. The parameter is used for images only.
 - **moments** – Output moments
 - The function computes moments, up to the 3rd order, of a vector shape or a rasterized shape. The results are returned in the structure Moments defined as: Moments(double m00, double m10, double m01, double m20, double m11, double m02, double m30, double m21, double m12, double m03);
 - the spatial moments Moments::mji are computed as

$$m_{ji} = \sum_{x,y} (\text{array}(x,y) \cdot x^j \cdot y^i)$$

- The central moments Moments::muji are computed as

$$\mu_{ji} = \sum_{x,y} (\text{array}(x,y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i) \quad \bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}}$$

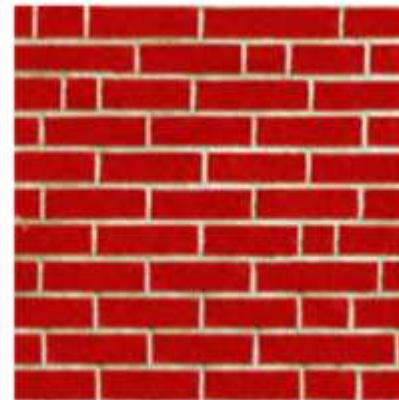
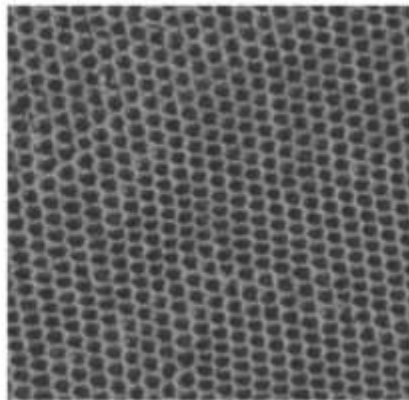
Python:

```
cv.moments( array[, binaryImage] ) -> retval
```

4.6.2 텍스처

■ 텍스처란?

- 일정한 패턴의 반복
- 구조적 방법과 통계적 방법



- An image obeying some statistical properties
- Similar structures repeated over and over again
- Often has some degree of randomness

4.6.2 텍스트처: 전역 기술자

■ 에지 기반 텍스트처

$$T_{edge} = (busy, mag(i), dir(j)), 0 \leq i \leq q-1, 0 \leq j \leq 7$$

$$\text{이때 } busy = \frac{\text{영역의 에지 화소 수}}{\text{영역의 화소 수}}$$

$mag(i)$ 는 에지강도를 q 단계로 양자화한 히스토그램
 $dir(j)$ 는 8단계로 양자화한 에지 방향 히스토그램

■ 명암 히스토그램 기반 텍스트처

$$\mu_r = \sum_{l=0}^{L-1} (l-m)^r \hat{h}(l)$$

$$\text{이때 } m = \sum_{l=0}^{L-1} l \hat{h}(l)$$

$$T_{histogram} = (smooth, skew, uniform, entropy)$$

$$\text{이때 } smooth = 1 - \frac{1}{1 + \frac{\mu_2}{(L-1)^2}}$$

$$skew = \mu_3$$

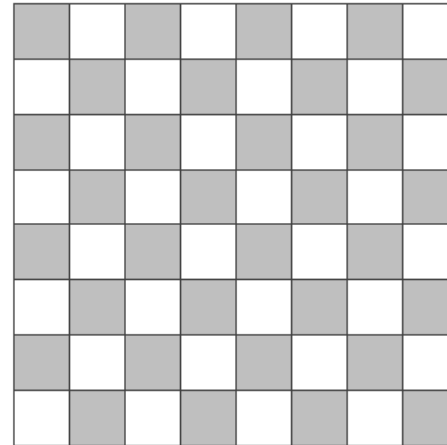
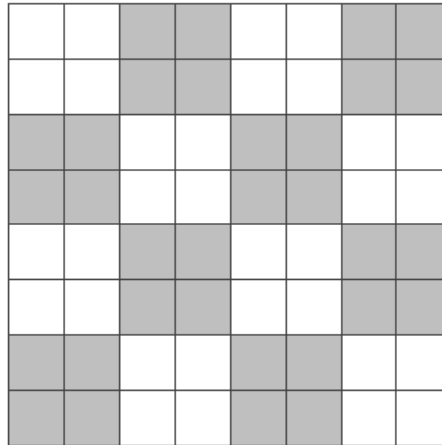
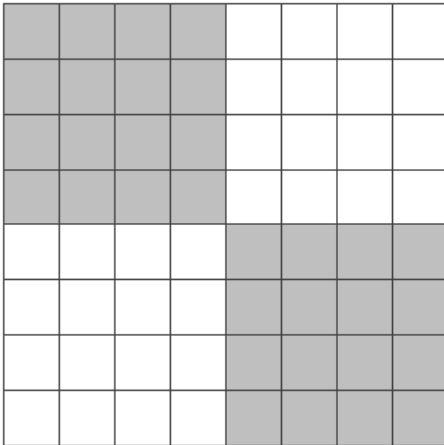
$$uniform = \sum_{l=0}^{L-1} \hat{h}(l)^2$$

$$entropy = - \sum_{l=0}^{L-1} \hat{h}(l) \log_2 \hat{h}(l)$$

전역 기술자

■ 한계

- 전역 텍스처 기술자는 니들의 차이를 구별하지 못함
- 3영상의 히스토그램이 같아서 지역적인 정보를 반영하지 못함



4.6.2 텍스처: 지역 관계 기술자

■ 원리

- 화소 사이의 이웃 관계를 규정하고, 그들이 형성하는 패턴을 표현

■ 동시 발생 행렬

- 이웃 관계를 이루는 화소 쌍의 명암이 (j,i) 인 빈도수 세어, 행렬 O 의 요소 o_{ji} 에 기록

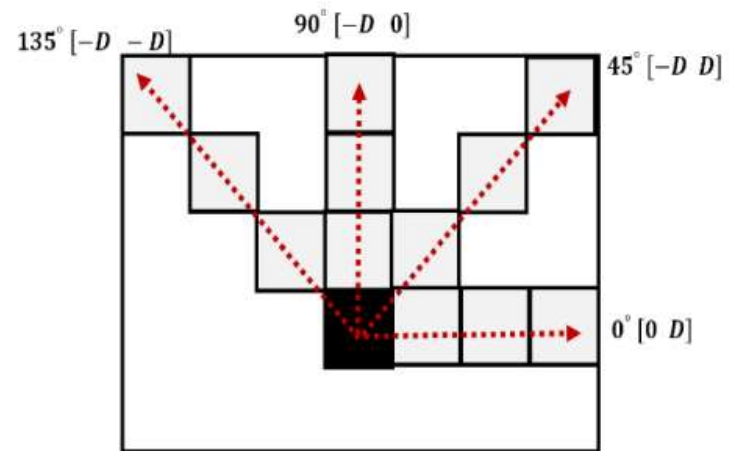
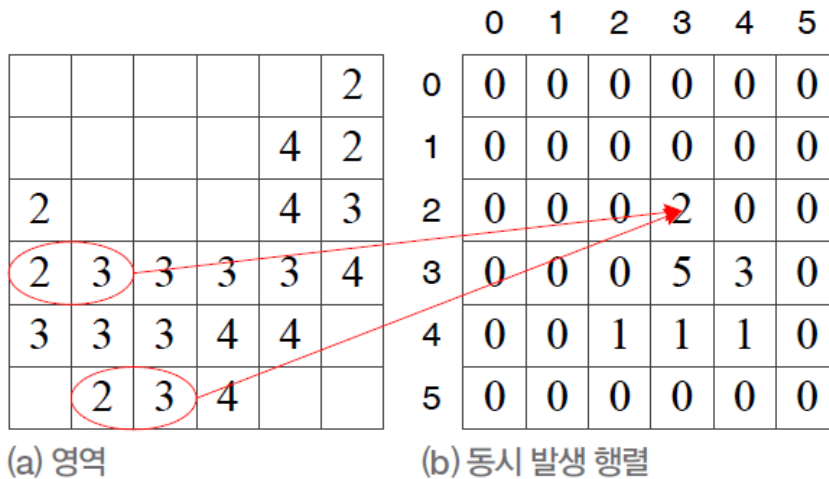


그림 6-16 동시 발생 행렬(이웃 관계는 '바로 오른쪽 이웃')

Four different directions with displacement 3 between two pixels

4.6.2 텍스트처: 지역 관계 기술자

■ 동시 발생 행렬에서 특징 추출

$$\hat{O}_{ji} = \frac{O_{ji}}{n} \quad (6.26)$$

$$\text{균일성(에너지) } energy = \sum_{j=0}^{q-1} \sum_{i=0}^{q-1} \hat{O}_{ji}^2 \quad (6.27)$$

$$\text{엔트로피 } entropy = - \sum_{j=0}^{q-1} \sum_{i=0}^{q-1} \hat{O}_{ji} \log_2 \hat{O}_{ji} \quad (6.28)$$

$$\text{대비 } contrast = \sum_{j=0}^{q-1} \sum_{i=0}^{q-1} (j-i)^2 \hat{O}_{ji} \quad (6.29)$$

$$\text{동질성 } homogeneity = \sum_{j=0}^{q-1} \sum_{i=0}^{q-1} \frac{\hat{O}_{ji}}{1 + |j-i|} \quad (6.30)$$

$$\text{공관계 } correlation = \sum_{j=0}^{q-1} \sum_{i=0}^{q-1} \frac{(j-\mu_r)(i-\mu_c)\hat{O}_{ji}}{\sigma_j\sigma_i}$$

$$\text{이때 } \mu_r = \sum_{j=0, q-1} j \sum_{i=0, q-1} \hat{O}_{ji}, \quad \mu_c = \sum_{i=0, q-1} i \sum_{j=0, q-1} \hat{O}_{ji} \quad (6.31)$$

$$\sigma_r^2 = \sum_{j=0, q-1} (j-\mu_c)^2 \sum_{i=0, q-1} \hat{O}_{ji}, \quad \sigma_c^2 = \sum_{i=0, q-1} (i-\mu_r)^2 \sum_{j=0, q-1} \hat{O}_{ji}$$

4.6.2 텍스트처: 지역 관계 기술자

■ 지역 이진 패턴 (Local Binary Pattern: LBP) [Ojala96]

- 8개 이웃과 대소관계에 따라 이진열을 만든 후 [0,255] 사이의 십진수로 변환
- 이러한 변환을 모든 화소에 적용한 후 결과를 가지고 히스토그램 구성
- 명암이 균일한 부근에서는 불안정한 단점이 있다
 - 작은 오차가 섞인 148, 149, 150, 151, 152와 같은 명암이 임의로 분포

22px에서 해결

15보다 커서 1

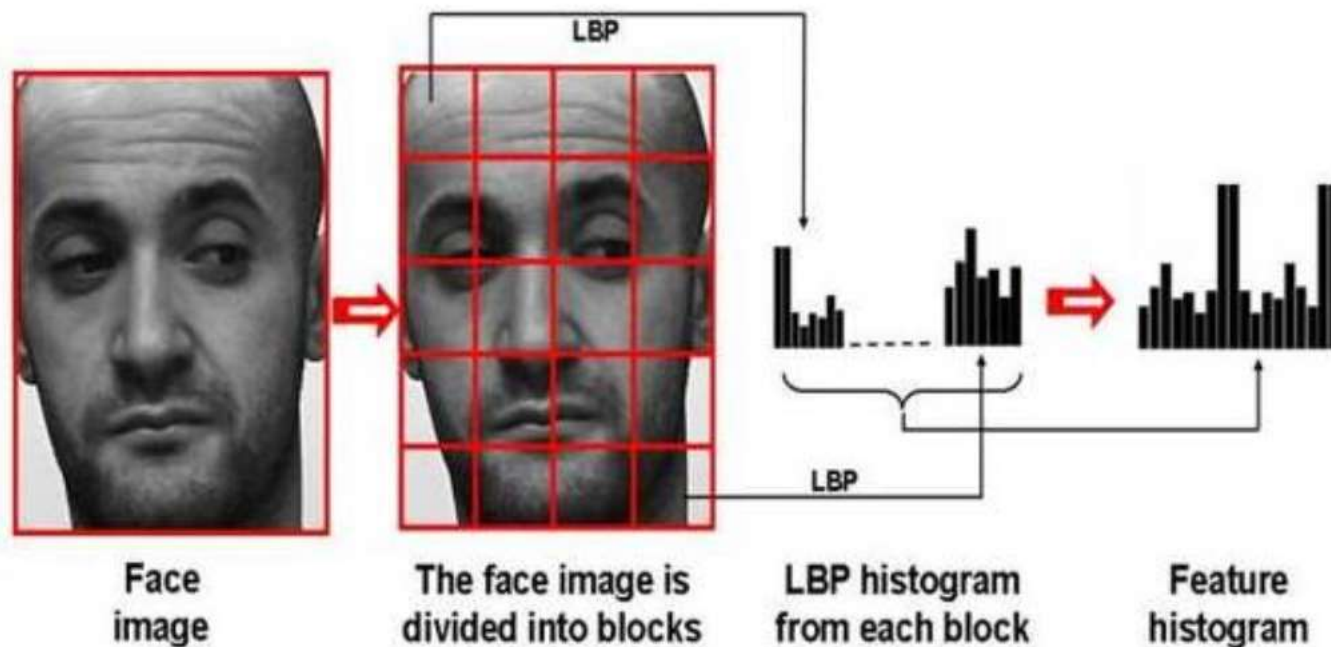
136	165	147	133	139	125
105	150	142	143	163	140
113	153	160	176	177	140
160	186	204	200	177	139
184	178	188	188	166	148
147	139	146	148	140	136

0	1	0
0		0
0	1	1

$$00110010 = 2 + 16 + 32 = 50$$

지역 관계 기술자

- 지역 이진 패턴 (LBP) [Ahonen06]를 얼굴인식에 사용



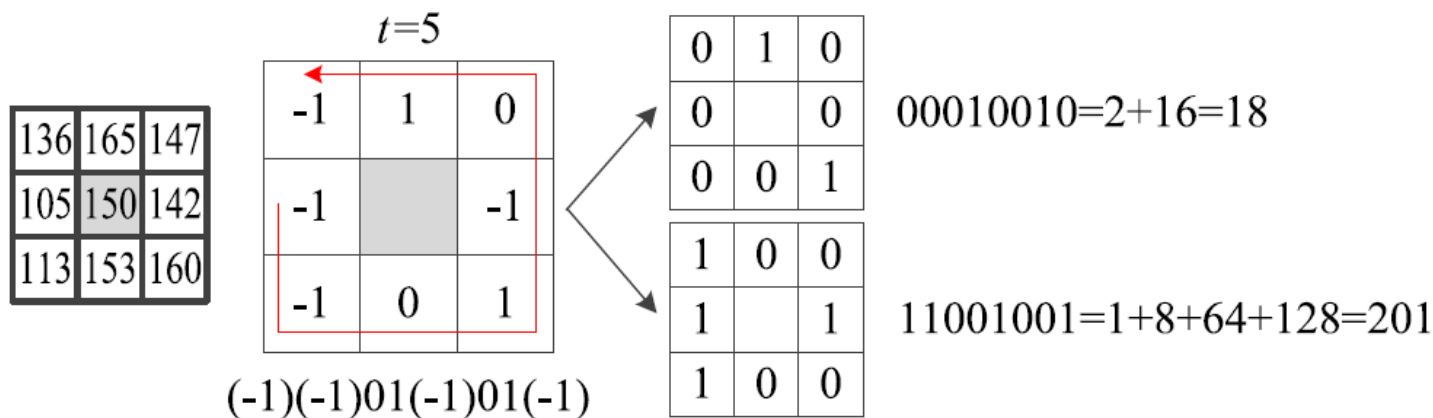
<그림 6> 출처: http://www.scholarpedia.org/article/Local_Binary_Patterns

The facial image is divided into local regions and LBP texture descriptors are extracted from each region independently. The descriptors are then **concatenated** to form a global description of the face, as shown in Fig. 4.

지역 관계 기술자

■ 지역 삼진 패턴 (LTP)

- 화소 값이 p 라면, $p-t$ 보다 작으면 -1 , $p+t$ 보다 크면 1 , $[p-t, p+t]$ 사이면 0 을 부여
- 두 개의 LBP로 분리
- 모든 화소를 가지고 히스토그램 구성: 512차원 특징벡터
- 예: $t=5$, 주변화소가 $145 \sim 155 \Rightarrow 0$, $>155 \Rightarrow 1$, $<145 \Rightarrow -1$



지역 관계 기술자

■ LBP와 LTP의 확장

- 조명 변환에 불변이나, 8이웃만 보면 스케일 변화에 대처 하지 못함
- 다양한 이웃을 이용한 스케일 불변 달성
 - 실수 좌표는 선형보간 사용하여 화소값 추정

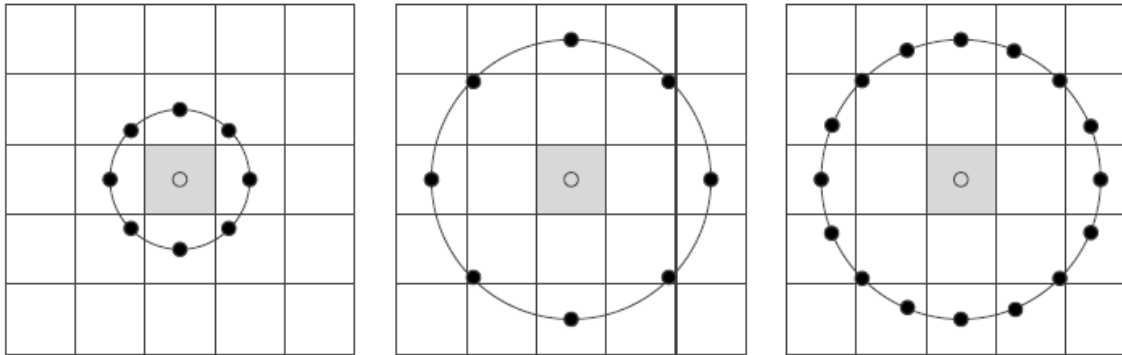


그림 6-19 LBP와 LTP가 사용하는 여러 가지 이웃

■ LBP와 LTP의 응용

- 예) 얼굴 검출, 사람 검출, 자연 영상에서 글자 추출 등

4.6 영역 특징

프로그램 4-8

이진 영역의 특징을 추출하는 함수 사용하기

```
01  import skimage
02  import numpy as np
03  import cv2 as cv
04
05  orig=skimage.data.horse()
06  img=255-np.uint8(orig)*255
07  cv.imshow('Horse',img) ①
08
09  contours,hierarchy=cv.findContours(img,cv.RETR_EXTERNAL,cv.CHAIN_APPROX_NONE)
10
11  img2=cv.cvtColor(img,cv.COLOR_GRAY2BGR)                # 컬러 디스플레이용 영상
12  cv.drawContours(img2,contours,-1,(255,0,255),2)
13  cv.imshow('Horse with contour',img2) ②
14
15  contour=contours[0]
16
```


4.6 영역 특징

```
17 m=cv.moments(contour) # 몇 가지 특징
18 area=cv.contourArea(contour)
19 cx,cy=m['m10']/m['m00'],m['m01']/m['m00']
20 perimeter=cv.arcLength(contour,True)
21 roundness=(4.0*np.pi*area)/(perimeter*perimeter)
22 print('면적=',area,'\n중점=(,cx,',',cy,')','\nd둘레=',perimeter,'\nd둥근 정도=',
      roundness) ③
23
24 img3=cv.cvtColor(img,cv.COLOR_GRAY2BGR) # 컬러 디스플레이용 영상
25
26 contour_approx=cv.approxPolyDP(contour,8,True) # 직선 근사
27 cv.drawContours(img3,[contour_approx],-1,(0,255,0),2)
28
29 hull=cv.convexHull(contour) # 볼록 헐
30 hull=hull.reshape(1,hull.shape[0],hull.shape[2])
31 cv.drawContours(img3,hull,-1,(0,0,255),2)
32
33 cv.imshow('Horse with line segments and convex hull',img3) ④
34
35 cv.waitKey()
36 cv.destroyAllWindows()
```

4.6 영역 특징

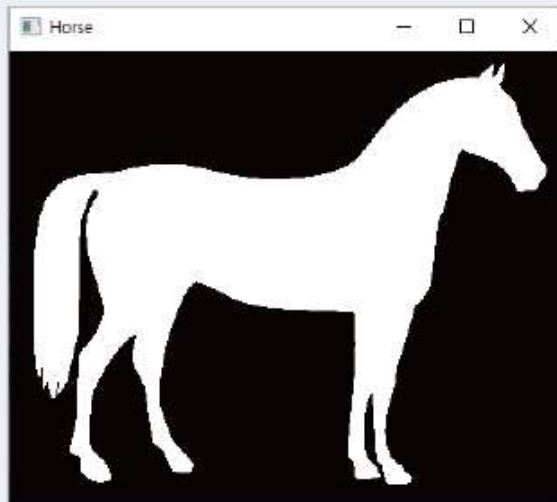
면적= 42390.0 ③

중점=(187.72464024534088 , 144.43640402610677)

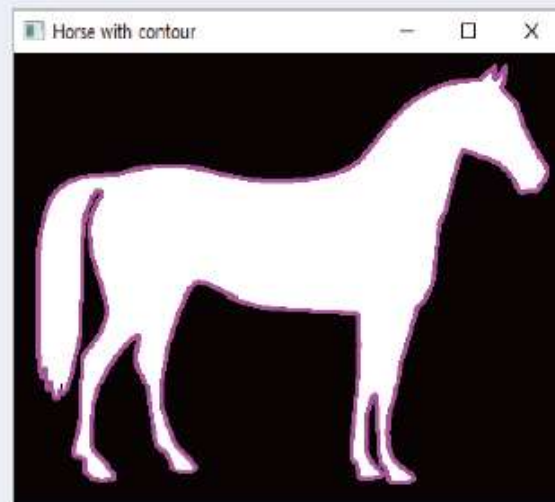
둘레= 2296.7291333675385

둥근 정도= 0.1009842680321435

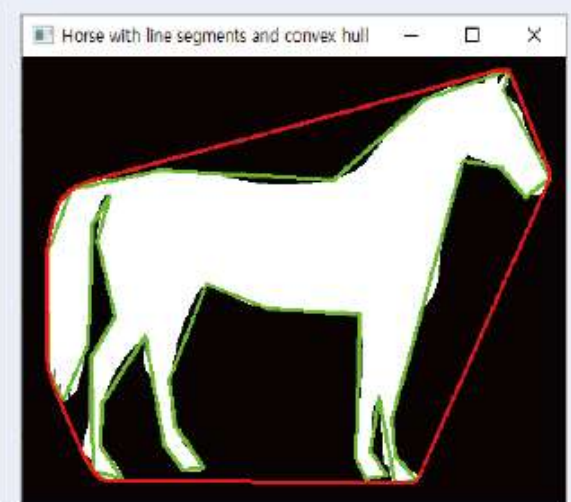
①



②



④



과제

- binaryGroup 영상에 작거나(둘레길이 50 이하) 큰 물체(둘레 길이 1000 이상)의 외곽선을 제거한 후 남은 영상에 boundingbox, enclosing circle, polygonal 근사, 그리고 Convex hull을 이용해서 물체를 묘사하시오. 중심점에 두께 및 반경이 2인 원을 그리고 각 물체의 길이와 중심점을 출력하시오.

