

Basic image processing techniques

Byeongjoon Noh

Dept. of AI and Bigdata, SCH Univ.

powernoh@sch.ac.kr

Contents

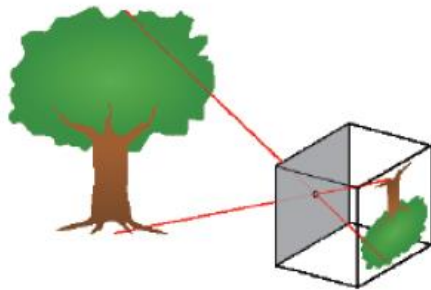
1. Sampling, Quantization, Encoding
2. Histogram
3. Binary image
4. Operations in image processing
5. Multi resolution

1. Sampling, Quantization, Encoding

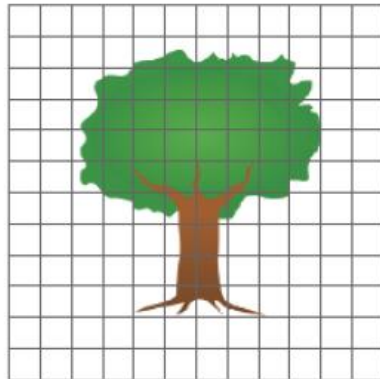
디지털 영상의 변환

표본화 (Sampling), 양자화 (Quantization), 부호화 (Encoding)

- 2차원 영상 공간은 $M \times N$ 차원으로 샘플링
 - $M \times N = \text{resolution}(\text{해상도})$
- 명암을 L 단계로 양자화
 - $L = \text{명암 단계 (0~L-1 사이에 분포)}$
- Example) $M=12, N=12, L=10$



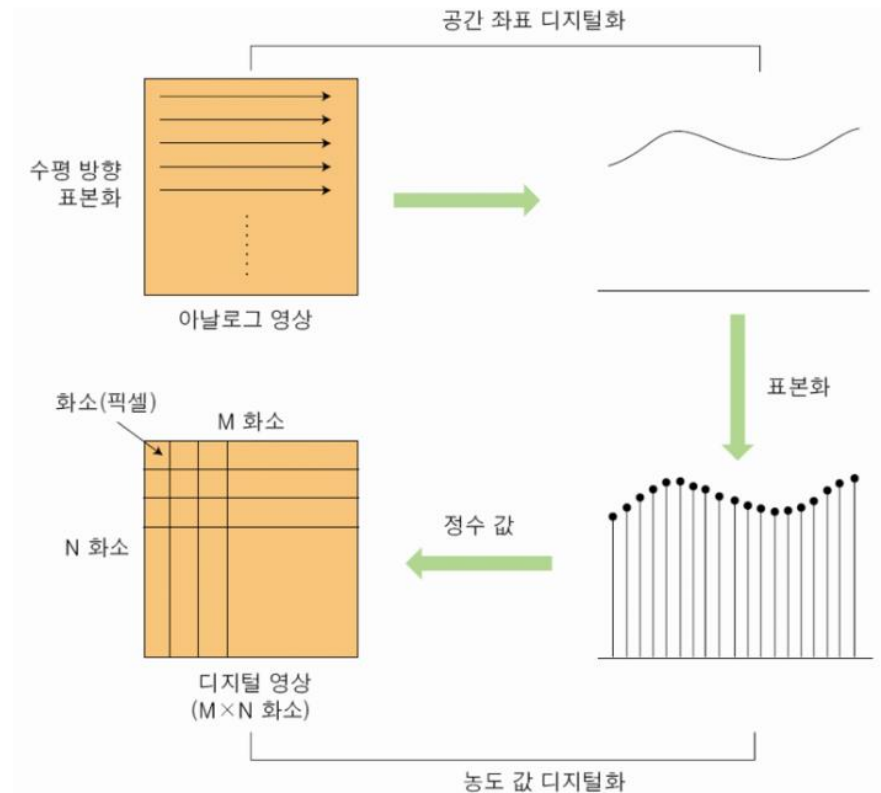
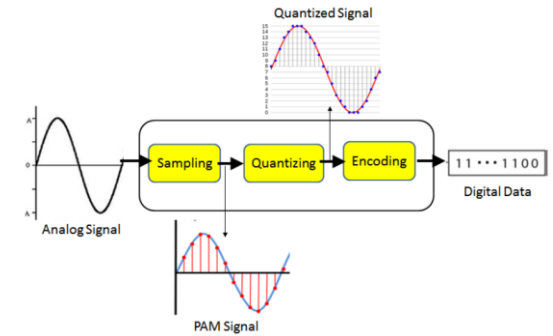
(a) 핀홀 카메라 모델



(b) 샘플링과 양자화

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

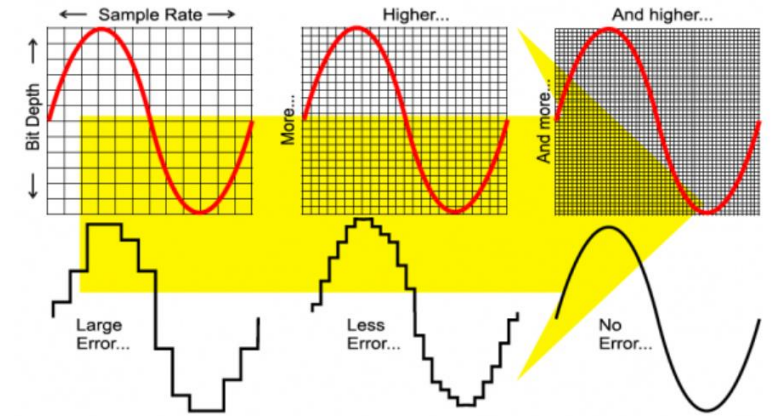
(c) 디지털 영상



디지털 영상의 변환

Sampling

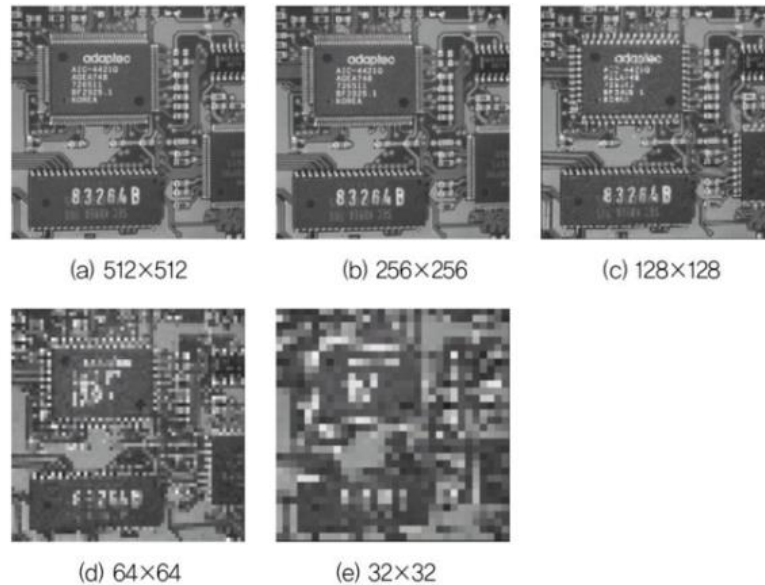
- 아날로그 신호를 일정 간격으로 표본화 하여
PAM (Puls Amplitude Modulation) 신호를 획득함. (데이터의 크기 결정)
- Sampling rate $\uparrow \rightarrow$ Data size \uparrow
- 일반적인 sampling rate = 44.1 KHz
- Nyquist Theorem: 아날로그 신호의 최고 주파수 2배 크기로 sampling 진행
 \rightarrow 디지털 신호를 원래의 아날로그 신호의 손실 없이 복원 할 수 있음



디지털 영상의 변환

Sampling in Digital image

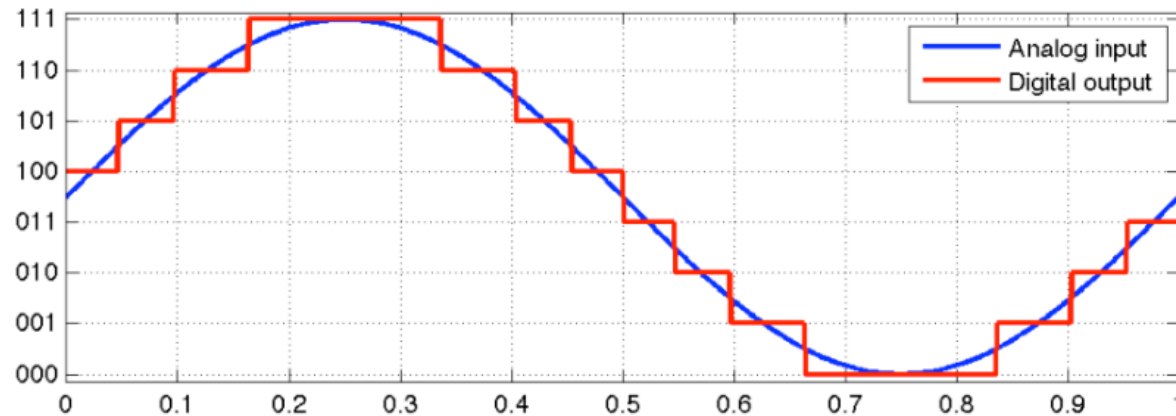
- 아날로그 영상에서 공간적, 시간적으로 연속되는 밝기 강도 (Intensity)에 따라 이산적인 점 (Pixel) 을 추출하는 과정
- Sampling rate $\uparrow \rightarrow$ High resolution (Huge size...)



디지털 영상의 변환

Quantization

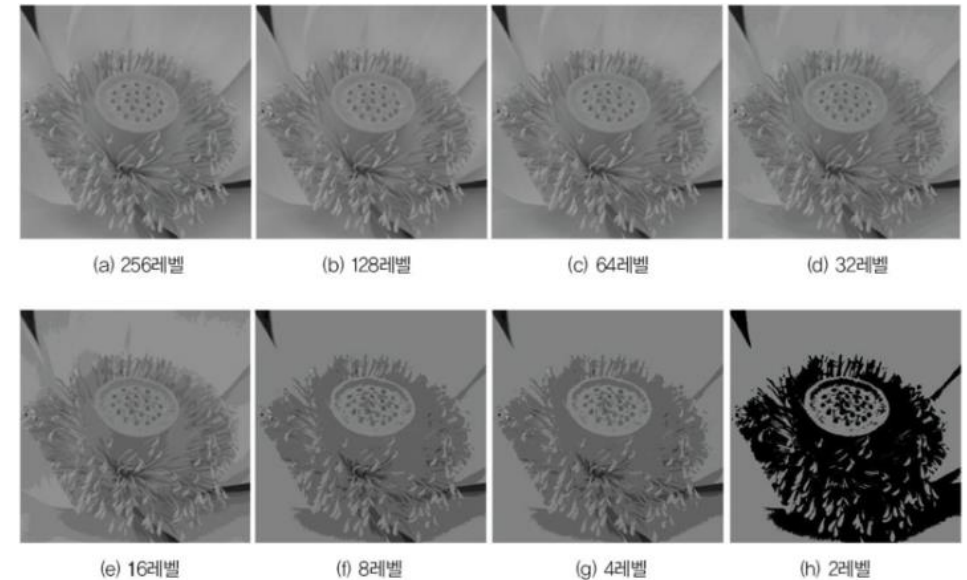
- Sampling된 값도 결국 아날로그의 형태
➔ 디지털 신호로 완전한 변환이 필요함
- 각 아날로그 값들을 맵핑 (Mapping)하여 변환
 - Quantization noise 발생



디지털 영상의 변환

Quantization in Digital image

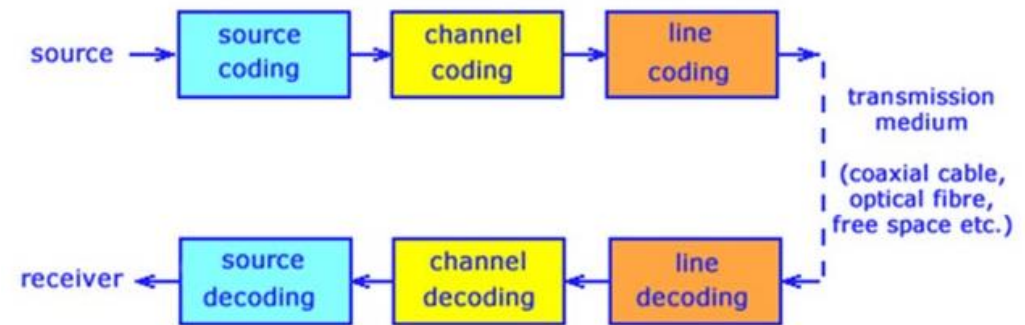
- Sampling된 영상의 각 화소의 밝기나 색을 정해진 몇 단계의 값으로 근사화
 - ➔ 각 화소의 밝기나 색이 숫자로 표현
 - ➔ 각 화소에 양자화된 표본 값을 획득
-
- Gray level 해상도(진폭)를 결정
Quantization \uparrow ➔ 표현할 수 있는 색상 \uparrow



디지털 영상의 변환

Encoding

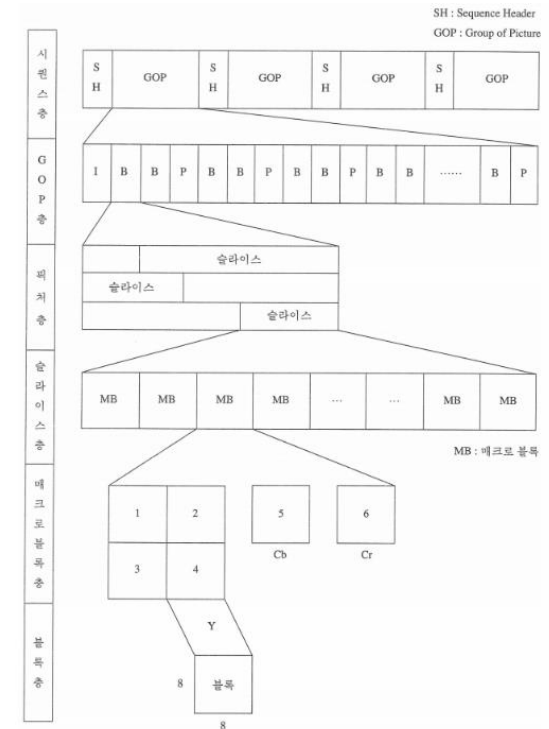
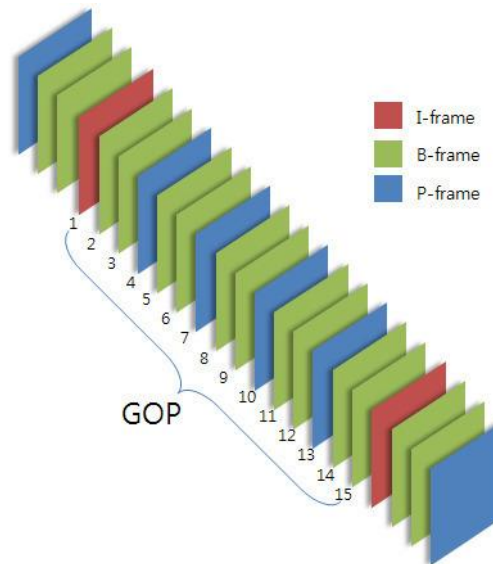
- 양자화된 값을 이진 데이터로 변환하는 과정
 - 단위 시간당 가능한 많은 데이터를 처리할 수 있는 방법을 찾고자 고안됨
- 신호처리: 소스 부호화, 채널 부호화, 라인 부호화 등
- 복호화 (Decoding) 함께 수행됨



디지털 영상의 변환

Encoding in Digital image

- 양자화된 화소의 밝기나 색 데이터를 2진수로 표현하는 과정
- 압축 부호화를 수행함
 - MPEG, JPEG, ...



영상 좌표계

화소 위치 표현

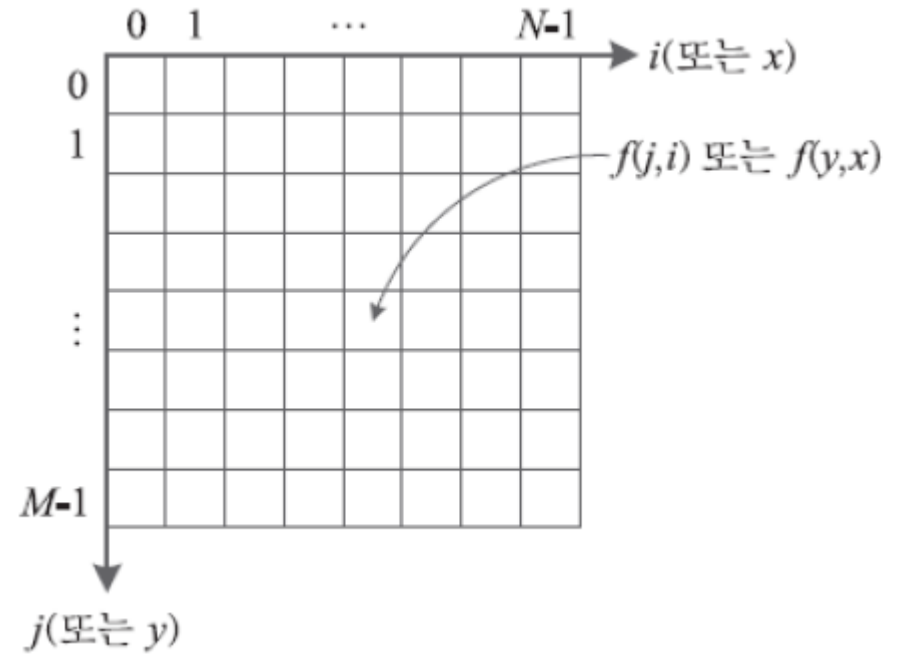
- $X(i, j)$ 또는 $X(y, x)$

영상 표현

- $f(X)$
- $f(j, i), 0 \leq j \leq M - 1, 0 \leq i \leq N - 1$

컬러 영상의 표현

- $f_c(x, y) = f_r(x, y), f_g(x, y), f_b(x, y)$



영상 좌표계

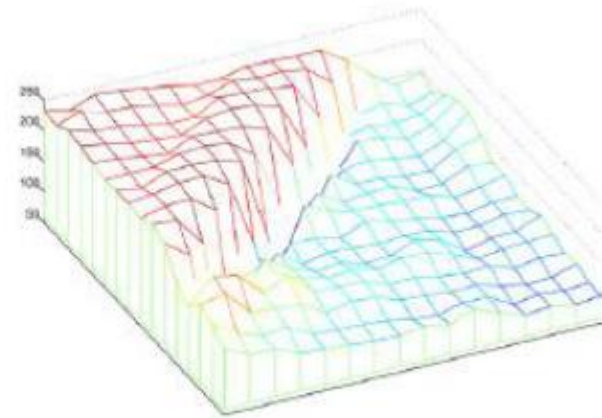
영상 표시 방법



(a) 영상

233	224	239	235	230	224	215	215	226	225	223	223	196	163	136	147
229	244	233	231	223	214	219	233	228	229	222	191	127	137	122	136
243	232	229	223	214	215	237	235	232	226	167	122	131	124	129	151
237	231	223	219	216	234	240	235	223	146	81	136	132	120	134	164
231	229	222	217	235	234	231	218	148	81	121	126	120	112	128	164
225	225	226	237	240	235	206	111	70	142	119	118	111	111	134	147
229	222	239	240	236	225	97	90	145	119	124	125	106	110	129	129
228	234	241	242	220	112	59	153	136	126	126	121	122	108	115	124
225	234	236	208	76	73	125	121	112	130	120	115	107	102	111	111
236	232	185	86	95	139	111	121	116	114	116	116	103	104	112	110
225	197	85	110	160	137	119	124	113	115	132	122	53	105	106	122
163	125	157	169	155	140	130	133	124	133	133	119	102	107	110	112
164	203	195	156	174	138	137	136	119	122	114	108	112	98	104	102
188	196	156	150	150	125	134	129	116	113	108	111	99	91	93	106
176	152	138	142	120	118	117	113	104	102	112	111	90	96	93	94
158	137	138	122	117	114	111	110	113	108	122	107	93	98	90	94

(b) 숫자 배열

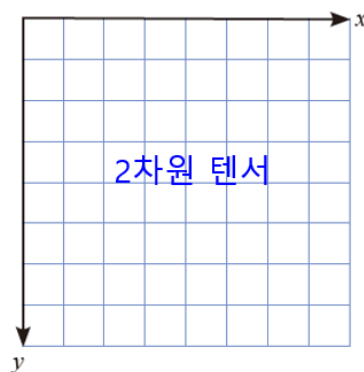


(c) 지형

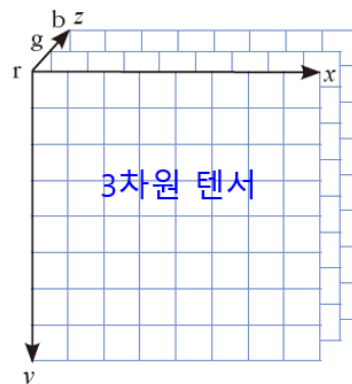
명암(그레이) 영상
명암 단계 $L = 256$ (0~255)
흑 백

영상 좌표계

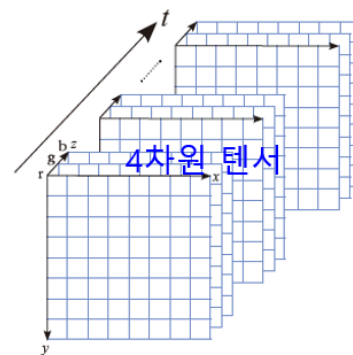
영상 표시 방법



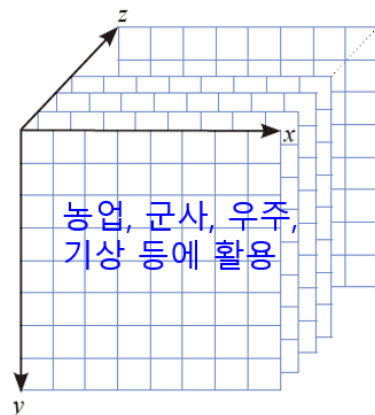
(a) 명암 영상



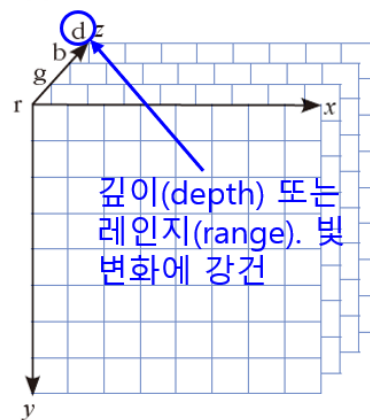
(b) 컬러 영상



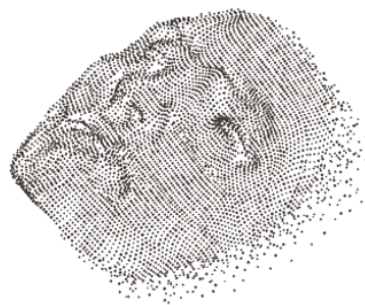
(c) 컬러 동영상



(d) 다분광/초분광/MR/CT 영상



(e) RGB-D 영상

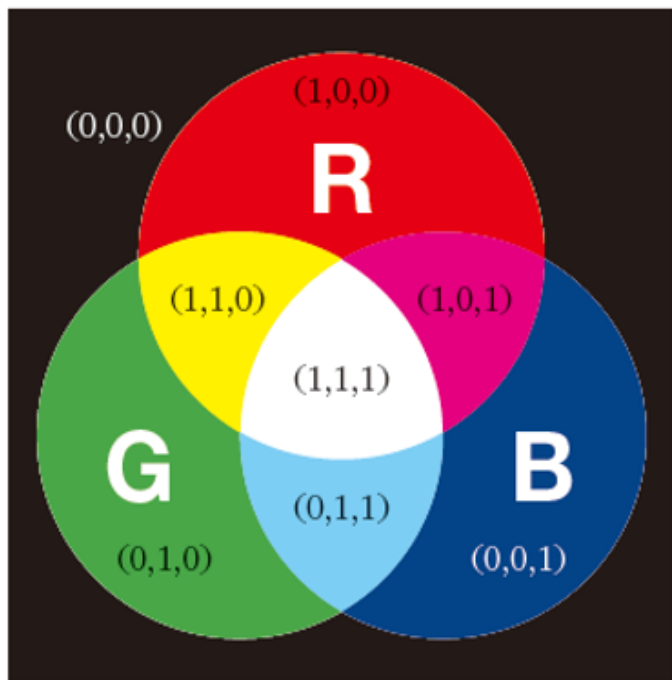


LiDAR로부터 획득

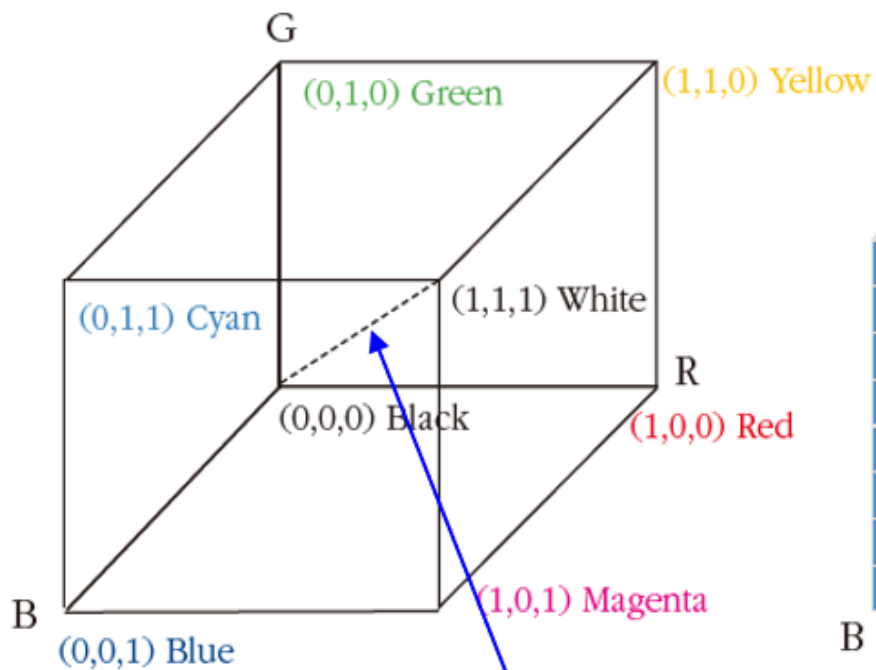
(f) 점 구름 영상

영상 좌표계

영상 표시 방법 – RGB 컬러 모델

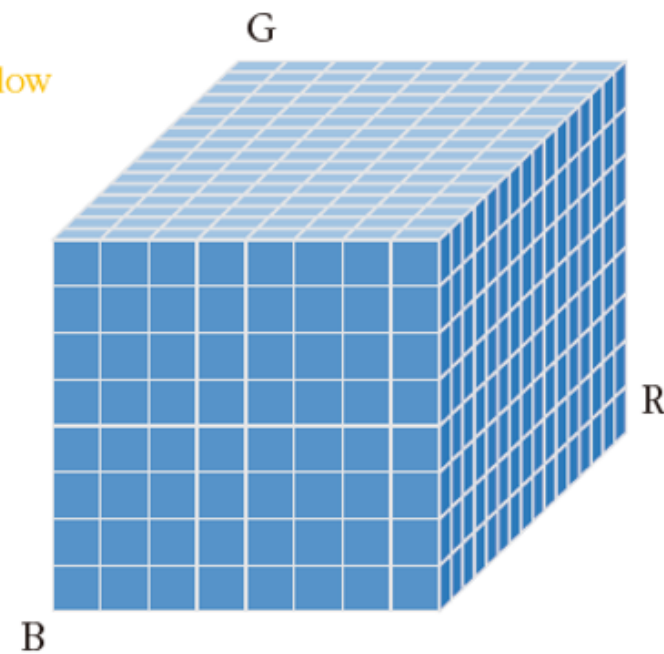


(a) RGB 삼원색의 혼합



(b) RGB 큐브

명암(gray scale)



(c) 양자화된 RGB 큐브

2. Histogram

히스토그램 (Histogram)

[0, L-1] 사이의 명암값 각각이 영상에 몇 번 나타나는지 표시하는 방법

- 히스토그램 h 와 정규화 히스토그램

$$h(l) = |\{j, i\} | (f(j, i) = l)$$

$$\hat{h}(l) = \frac{h(l)}{M * N}$$

알고리즘 2-1 명암 영상에서 히스토그램 계산

입력: 명암 영상 $f(j, i)$, $0 \leq j \leq M-1$, $0 \leq i \leq N-1$

출력: 히스토그램 $h(l)$ 과 정규 히스토그램 $\hat{h}(l)$, $0 \leq l \leq L-1$

```
1  for (l=0 to L-1)  h(l) = 0; // 초기화
2  for (j=0 to M-1)
3    for (i=0 to N-1) // f의 화소 (j, i) 각각에 대해
4      h(f(j, i))++; // 그곳 명암값에 해당하는 히스토그램 칸을 1만큼 증가
5  for (l=0 to L-1)
6     $\hat{h}(l) = h(l) / (M \times N)$ ; // 정규화한다.
```


히스토그램 (Histogram)

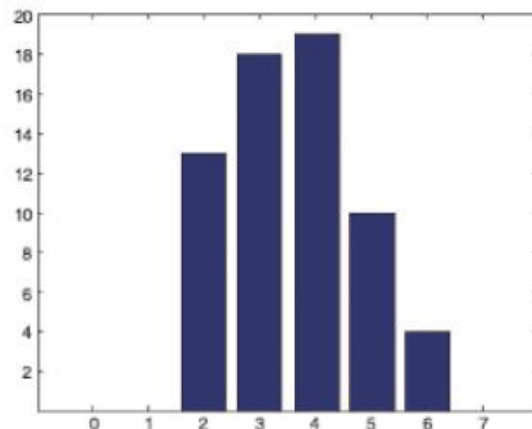
히스토그램 계산

예제 2-1 명암 영상에서 히스토그램 계산

[그림 2-7(a)]는 M 과 N 이 8이고 $L=8$ 인 아주 작은 영상이다. 이 영상에서 명암값이 2인 화소는 13개이므로 $h(2)=13$ 이다. 다른 명암값에 대해서도 화소의 개수를 세어보면 $h=(0,0,13,18,19,10,4,0)$ 이고, $\hat{h}(l)=(0,0,0.203,0.281,0.297,0.156,0.063,0)$ 이다. 이것을 그래프로 그리면 [그림 2-7(b)]와 같다.

3	2	2	2	2	3	3	4
3	2	2	2	3	4	3	3
4	3	3	4	4	4	3	3
5	4	4	4	5	4	3	3
5	4	3	4	5	4	3	2
6	5	4	4	5	4	3	2
6	6	5	5	4	3	2	2
6	5	4	5	4	3	2	2

(a) 8×8 영상 (8 명암 단계)

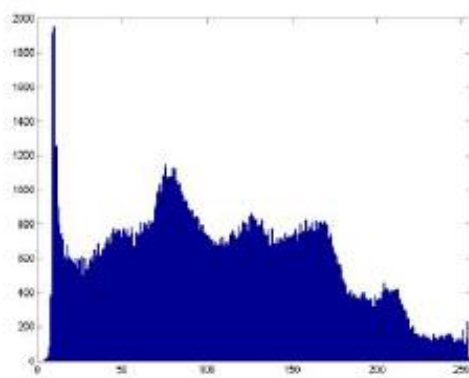
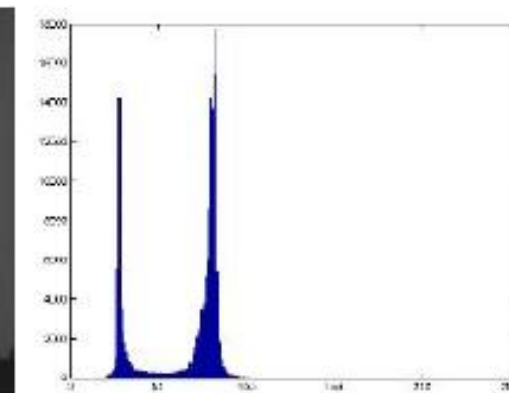
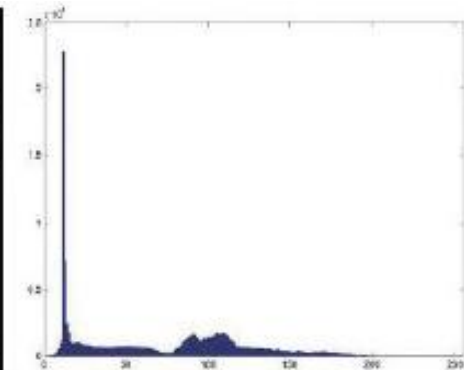


(b) 히스토그램

그림 2-7 히스토그램 예

히스토그램 용도

영상의 특성 파악



히스토그램 평활화 (Equalized)

히스토그램을 평평하게 만들어주는 연산

명암의 동적 범위를 확장하여 영상의 품질 향상에 기여

누적 히스토그램 $c(\cdot)$ 를 맵핑 함수로 사용 함

$$l_{out} = T(l_{in}) = \text{round}(c(l_{in}) \times (L - 1));$$
$$c(l_{in}) = \sum_{l=0}^{l_{in}} \hat{h}(l)$$

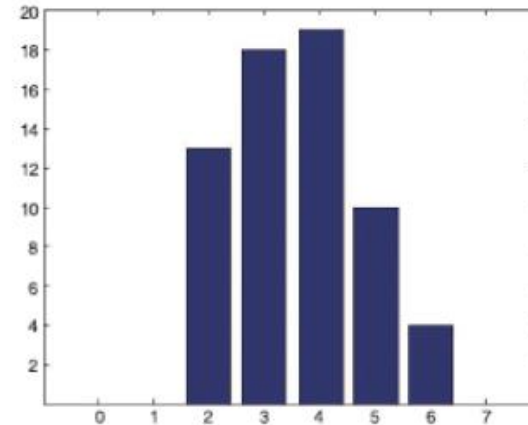
히스토그램 평활화 (Equalized)

평활화 예시

```

3 2 2 2 2 3 3 4
3 2 2 2 3 4 3 3
4 3 3 4 4 4 3 3
5 4 4 4 5 4 3 3
5 4 3 4 5 4 3 2
6 5 4 4 5 4 3 2
6 6 5 5 4 3 2 2
6 5 4 5 4 3 2 2
    
```

(a) 8×8 영상 (8 명암 단계)



(b) 히스토그램

Equalized

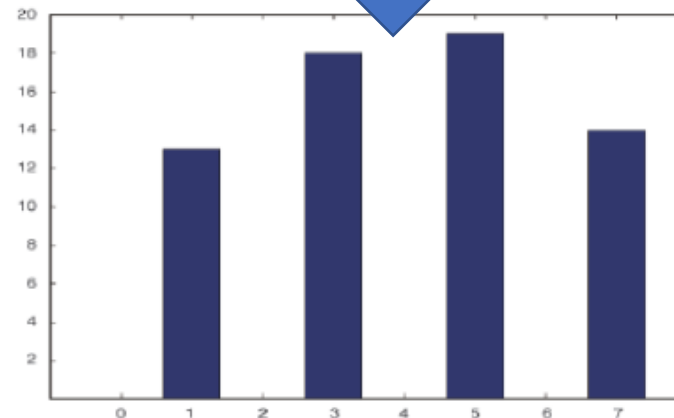
I_{in}	$\hat{h}(I_{in})$	$c(I_{in})$	$c(I_{in}) \times 7$	I_{out}
0	0.0	0.0	0.0	0
1	0.0	0.0	0.0	0
2	0.203	0.203	1.421	1
3	0.281	0.484	3.388	3
4	0.297	0.781	5.467	5
5	0.156	0.937	6.559	7
6	0.063	1.0	7.0	7
7	0.0	1.0	7.0	7

(a) 매핑 표 $T(.)$

```

3 1 1 1 1 3 3 5
3 1 1 1 3 5 3 3
5 3 3 5 5 5 3 3
7 5 5 5 7 5 3 3
7 5 3 5 7 5 3 1
7 7 5 5 7 5 3 1
7 7 7 7 5 3 1 1
7 7 5 7 5 3 1 1
    
```

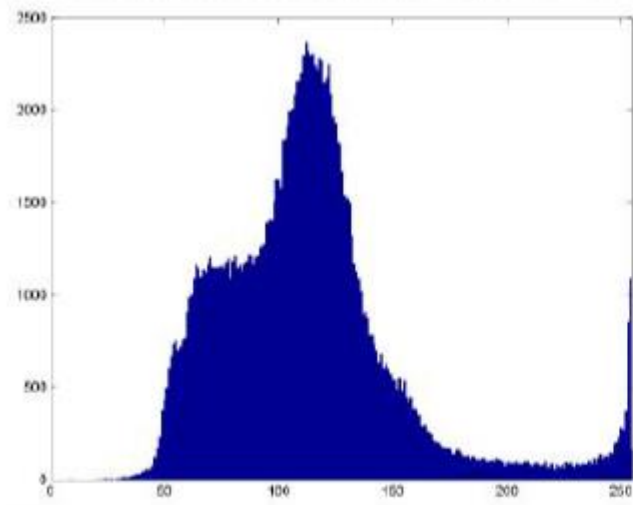
(b) 평활화된 영상



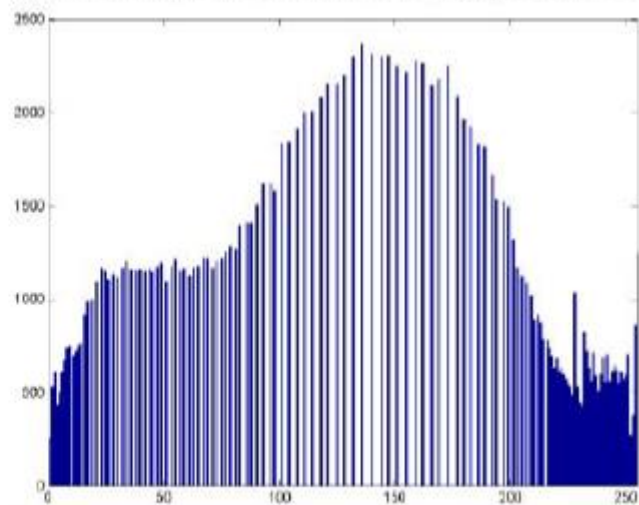
(c) 평활화된 영상의 히스토그램

히스토그램 평활화 (Equalized)

평활화를 통한 품질 향상 예시



(a) 원래 영상



(b) 히스토그램 평활화된 영상

히스토그램 평활화 (Equalized)

평활화를 통한 품질 저하(시각적 피로도 상승) 예시



(a) 원래 영상



(b) 히스토그램 평활화된 영상

➔ 영상처리 연산은 분별력을 가지고 활용 여부를 결정해야 함!

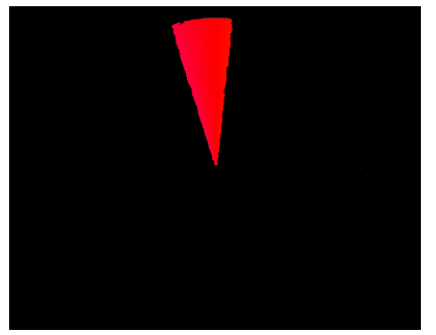
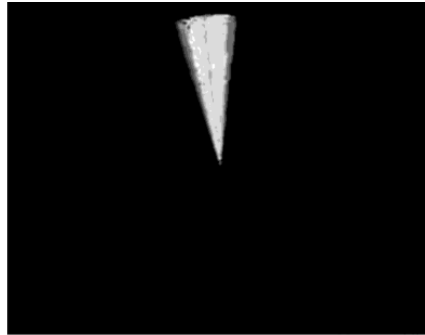
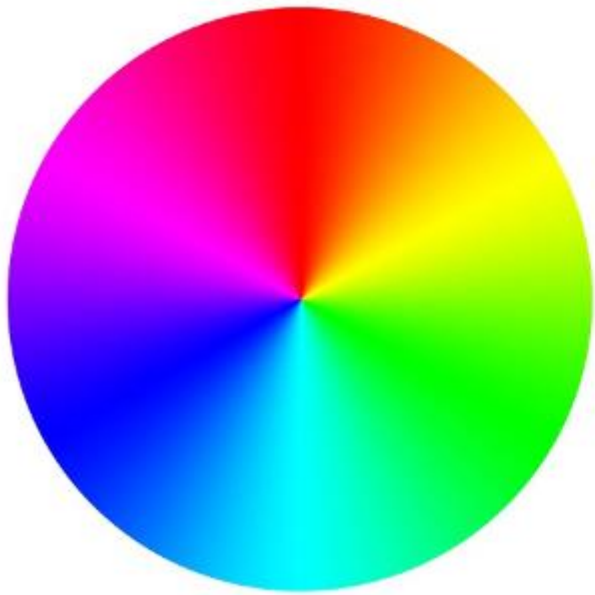
히스토그램 역투영

히스토그램 역투영 (Back projection)

- 히스토그램을 맵핑 함수로 사용하여, 화소 값을 신뢰도 값으로 변환
- 관심 영역 (RoI, Region of Interest)의 히스토그램과 유사한 히스토그램을 갖는 영역을 찾아내는 기법
 - 임의의 색상 영역을 검출할 때 효과적
- 이미지 내에서 특정 물체나 배경을 분리할 수 있음

히스토그램 역투영

히스토그램 역투영 예제



히스토그램 역투영

Color space (이미지 색상 표현 방식)

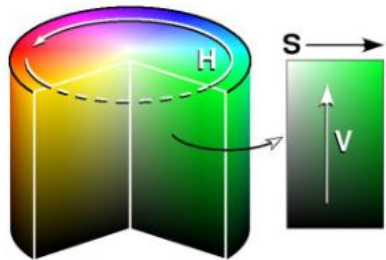
- RGB / BGR / RGBA
 - Red, Green, Blue, Alpha (투명도)
 - 0-255 사이의 값으로 표현
 - 값이 커질수록 해당 값이 두드러짐
 - (255, 0, 0) 빨간색
 - (255, 255, 255) 흰색
 - (0, 0, 0) 검은색



히스토그램 역투영

Color space (이미지 색상 표현 방식)

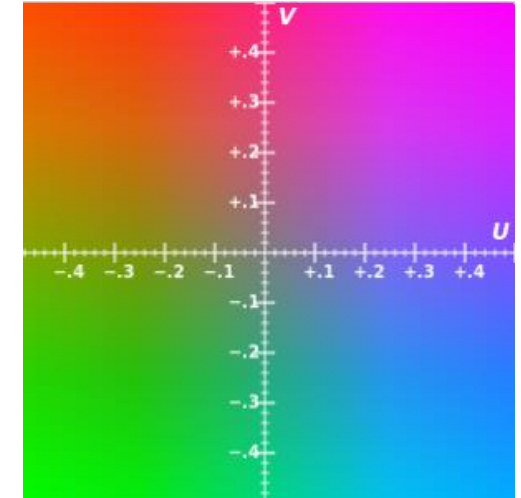
- Gray scale
 - 회색조 이미지
 - 0~255 사이의 값
- HSV
 - Hue (색조), Saturation (채도), Value (명도)



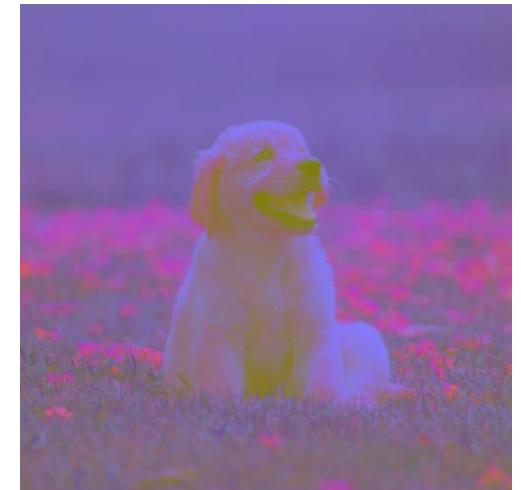
히스토그램 역투영

Color space (이미지 색상 표현 방식)

- YUV, YCbCr 방식
 - Y: 밝기 (Luma)
 - U: 밝기와 파란색 색상 차 (Chroma blue, Cb)
 - V: 밝기와 빨간색 색상 차 (Chroma red, Cr)
- Y에 많은 비트 수를 할당, U, V에 적은 비트 수 할당
- ➔ 데이터 압축 효과



Y=0.5



히스토그램 역투영

Color space 예제

- 1_1_colorSpace.py

```
src = cv2.imread(path, cv2.IMREAD_COLOR)

src = cv2.imread(path, cv2.IMREAD_GRAYSCALE)

gray2 = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

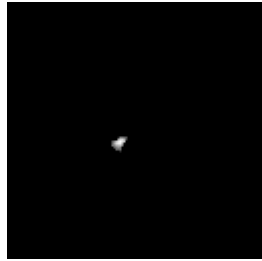
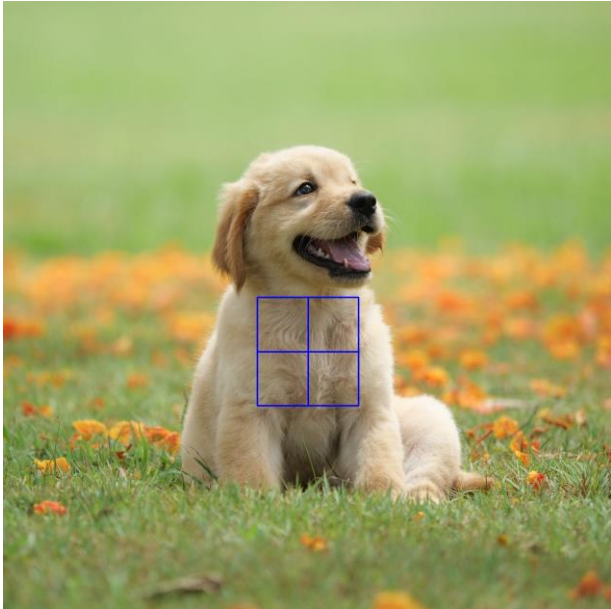
src = src.astype(np.uint16)
b,g,r = cv2.split(src)
gray3 = ((b + g + r)/3).astype(np.uint8)

ycbcr1 = cv2.cvtColor(src, cv2.COLOR_BGR2YUV)
```

히스토그램 역투영

히스토그램 역투영 예제 (OpenCV)

- 1_2_Backprojection.py



히스토그램 역투영

히스토그램 역투영 예제 (OpenCV)

- 1_2_BackProjection.py

```
# Histogram calculation
```

```
hist = cv2.calcHist([crop], channels, None, histSize, ranges)
```

```
hist_norm = cv2.normalize(cv2.log(hist + 1), None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)
```

```
# Histogram back-projection
```

```
backproj = cv2.calcBackProject([src_ycrcb], channels, hist, ranges, 1)
```

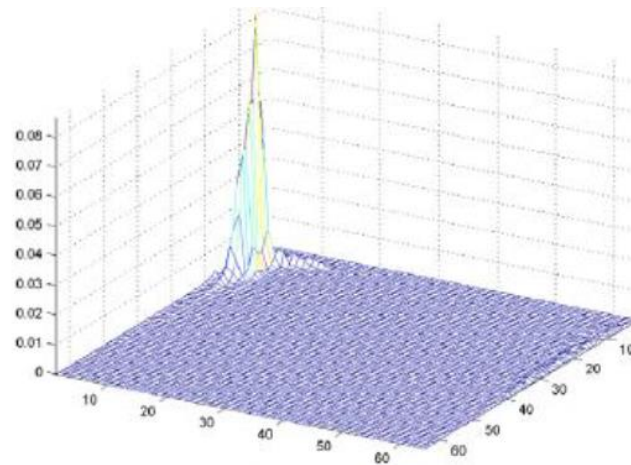
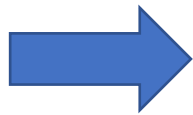
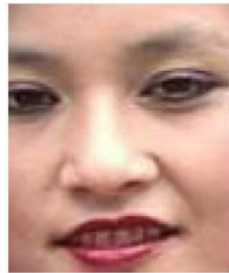
```
# Masking
```

```
dst = cv2.copyTo(src, backproj)
```

히스토그램 역투영

얼굴 검출 예

- 모델 얼굴과 2차원 히스토그램
- 조명 변화를 고려하여 I (Intensity)를 제외한 HS (Hue-Saturation) 공간에서 처리



2차원 히스토그램 (HS공간)

히스토그램 역투영

2차원 히스토그램

알고리즘 2-2 2차원 히스토그램 계산(HS 공간)

입력 : H 와 S 채널 영상 $f_H(j, i), f_S(j, i), 0 \leq j \leq M-1, 0 \leq i \leq N-1$

출력 : 히스토그램 $h(j, i)$ 와 정규 히스토그램 $\hat{h}(j, i), 0 \leq j, i \leq q-1$ // L 단계를 q 단계로 양자화

64

```
1   $h(j, i), 0 \leq j, i \leq q-1$  을 0으로 초기화한다.
2  for( $j=0$  to  $M-1$ )
3    for( $i=0$  to  $N-1$ ) // 화소  $(j, i)$  각각에 대해
4       $h(\text{quantize}(f_H(j, i)), \text{quantize}(f_S(j, i)))++$ ; // 해당 칸을 1 증가시킴
5  for( $j=0$  to  $q-1$ )
6    for( $i=0$  to  $q-1$ )
7       $\hat{h}(j, i) = h(j, i) / (M \times N)$ ; // 정규화
```


히스토그램 역투영

2차원 히스토그램

- 양자화(Quantization)란?
 - 표본화된 각 화소의 밝기나 색을 정해진 몇 단계의 값으로 근사화하는 과정
 - 각 화소의 밝기나 색이 숫자로 표현되어 양자화된 표본 값이 생성됨

히스토그램 역투영

히스토그램 역투영 결과

- 얼굴 영역, 손 영역에서 높은 신뢰도 값
- 장점: 배경을 조정할 수 있는 상황에 적합
 - 이동과 회전에 불변, 가림 (Occlusion)에 강인
- 한계: 비슷한 색 분포를 갖는 다른 물체 구별 어려움
 - 검출 대상이 여러 색 분포를 갖는 경우 오류 확률 상승

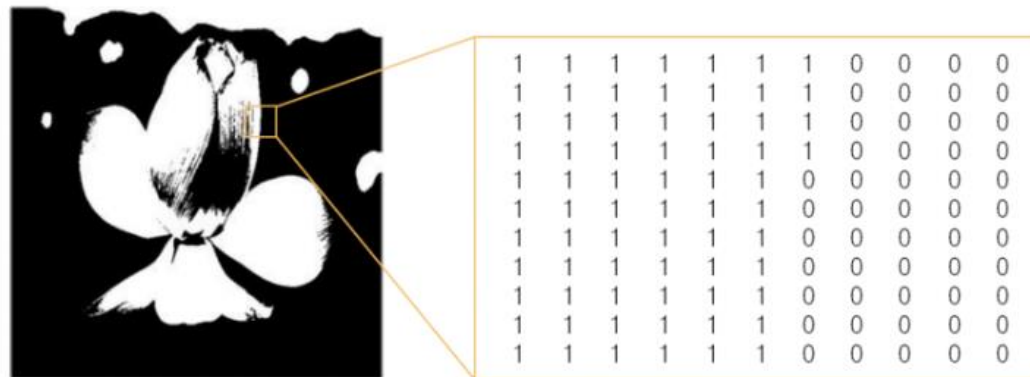


3. Binary Image

Binary Image

디지털 영상의 종류

- Binary image
 - 화소값이 흑/백으로만 구성된 영상 ➔ 처리 속도 빠름
 - Quantization 진행 시 양자화 bit 수를 1로 지정함
 - 경계 구분이 정확하지 않은 영상(이미지)에서는 영상 정보 손실됨
 - 지문, 팩스, 문자 영상 등



Binary Image

디지털 영상의 종류

- Gray-level image
 - Binary image보다 조금 더 밝게 표현됨 (bit 수 ↑)
 - 화소의 밝기가 여러 단계 (흑백 사진)
 - 밝기의 단계: 검정색~회색~흰색 (단계 수는 quantization bit 수가 결정)

$$0 \leq f(x, y) \leq 2^n - 1$$



Binary Image

디지털 영상의 종류

- Color image
 - 실제로 눈에 보이는 모습과 유사한 밝기와 색상을 가진 영상
 - **빛의 3원색**을 이용
 - 각 색을 독립적인 형태로 처리하고 합침
 - 각 색의 상호작용이 큼 ➔ 처리 어려움



158	158	160	159	163	167	126	59	52	53	52
158	159	159	157	161	166	124	58	52	54	53
159	159	158	157	161	166	122	56	52	54	53
160	160	158	159	164	166	119	55	51	55	55
159	160	158	160	166	164	114	54	53	55	55
157	159	157	160	165	163	109	52	54	55	56
159	161	156	159	164	162	108	50	52	55	56
160	162	156	157	165	163	107	50	53	56	56
160	161	155	157	165	162	103	50	53	56	54
160	161	156	158	164	160	100	49	51	55	54
158	159	156	159	164	160	98	49	51	55	54

103	103	105	106	119	129	111	75	80	80	78
103	105	105	104	117	130	111	75	81	81	80
104	104	103	104	116	130	110	77	81	82	81
105	104	103	107	117	129	107	77	83	83	82
105	105	104	108	120	127	103	76	85	83	82
106	106	102	108	121	123	100	77	83	82	84
107	106	101	108	120	123	100	75	82	83	85
107	108	101	108	121	126	98	75	83	84	87
107	108	102	109	121	126	96	76	82	84	87
105	108	102	108	120	124	94	75	82	84	86
105	108	101	107	120	125	93	77	83	84	86

132	132	132	131	140	148	112	34	28	29	29
133	132	130	129	138	148	110	34	28	31	29
133	133	130	130	138	149	108	34	29	31	30
134	133	132	135	141	148	105	32	28	33	31
133	134	131	136	145	146	99	31	30	33	31
132	132	130	136	142	144	91	31	32	32	33
135	133	130	135	142	143	91	30	30	32	34
136	136	130	137	144	145	89	30	31	33	33
136	136	131	137	144	145	85	28	31	33	32
134	136	131	137	144	143	82	28	31	33	33
132	135	129	135	144	141	79	29	31	34	33

Binary Image

이진화 (Binarization) $f(x, y) = 0, 1$

- “명암” 영상을 흑과 백만 가진 “이진” 영상으로 변환
- 다양한 이진화 방법이 존재
- 임계값 (Threshold) 방법

$$b(j, i) \begin{cases} 1 & f(j, i) \geq T \\ 0 & f(j, i) < T \end{cases}$$



Binary Image

$$h(l) = |\{j, i\} | (f(j, i) = l)$$

$$\hat{h}(l) = \frac{h(l)}{M * N}$$

Otsu algorithm

- Idea: 이진화 시 Black/White 그룹 각각이 균일할수록 좋다!
 - 균일성 \rightarrow 분산으로 측정 (분산이 작을수록 균일성 높음)
- 분산의 가중치 합 $v_{within}(\cdot)$ 을 목적 함수로 이용한 최적화 알고리즘

$$T = \operatorname{argmin}_{t \in \{0, 1, \dots, L-1\}} v_{within}(t)$$

$$v_{within}(t) = w_0(t)v_0(t) + w_1(t)v_1(t)$$

Binary Image

$$h(l) = |\{j, i\} | (f(j, i) = l)|$$

$$\hat{h}(l) = \frac{h(l)}{M * N}$$

Otsu algorithm

$$T = \operatorname{argmin}_{t \in \{0, 1, \dots, L-1\}} v_{\text{within}}(t)$$

$$v_{\text{within}}(t) = w_0(t)v_0(t) + w_1(t)v_1(t)$$

$$w_0(t) = \sum_{i=0}^t \hat{h}(i), \quad w_1(t) = \sum_{i=t+1}^{L-1} \hat{h}(i)$$

$$v_0(t) = \frac{1}{w_0(t)} \sum_{i=0}^t \hat{h}(i)(i - \mu_0(t))^2$$

$$\mu_0(t) = \frac{1}{w_0(t)} \sum_{i=0}^t i \hat{h}(i), \quad \mu_1(t) = \frac{1}{w_1(t)} \sum_{i=t+1}^{L-1} i \hat{h}(i)$$

$$v_1(t) = \frac{1}{w_1(t)} \sum_{i=t+1}^{L-1} \hat{h}(i)(i - \mu_1(t))^2$$

Binary Image

Otsu algorithm

- $t - 1$ 번째 계산 결과를 t 번째에 활용하여 빠르게 계산 (Dynamic programming)

$$T = \operatorname{argmin}_{t \in \{0, 1, \dots, L-1\}} v_{between}(t)$$

$$v_{between}(t) = w_0(t)(1 - w_0(t))(\mu_0(t) - \mu_1(t))^2$$

Initial value ($t = 0$): $w_0(0) = \hat{h}(0), \mu_0(t) = 0$

Recursive exp. ($t > 0$):

$$w_0(t) = w_0(t-1) + \hat{h}(t)$$

$$\mu_0(t) = \frac{w_0(t-1)\mu_0(t-1) + t\hat{h}(t)}{w_0(t)}$$

$$\mu_1(t) = \frac{\mu - w_0(t)\mu_0(t)}{1 - w_0(t)}$$

Binary Image

Practice

- 1_3_binaryImage.py

```
src = cv2.imread(path)
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
ret, dst = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
```

cv2.THRESH_BINARY	임계값 이상 = 최댓값, 임계값 이하 = 0
cv2.THRESH_BINARY_INV	위의 반전, 임계값 이상 = 0, 임계값 이하 = 최댓값
cv2.THRESH_TOZERO	임계값 이상 = 원본값, 임계값 이하 = 0
cv2.THRESH_TOZERO_INV	위의 반전, 임계값 이상 = 0, 임계값 이하 = 원본값
cv2.THRESH_TRUNC	임계값 이상 = 임계값 임계값 이하 = 원본값
cv2.THRESH_MASK	흑색 이미지로
cv2.THRESH_OTSU	otsu 알고리즘
cv2.THRESH_TRIANGLE	triangle 알고리즘

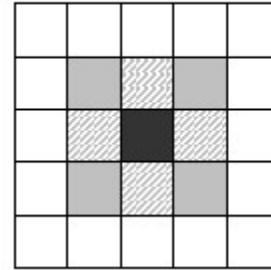


Connected component

연결 요소

- 화소의 모양

Pixels Connectivity Patterns



Pixels, 8-connected to black one

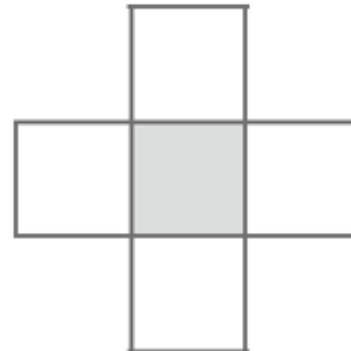


Pixels, 4- and 8-connected to black one

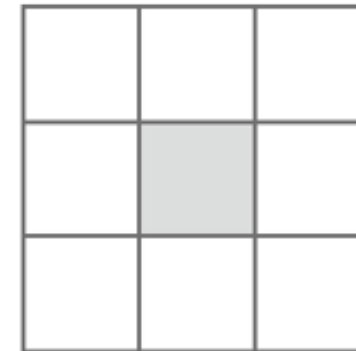
- 화소의 연결성

	$i-1$	i	$i+1$
$j-1$	nw	n	ne
j	w		e
$j+1$	sw	s	se

4-연결성



8-연결성



Connected component

연결 요소 indexing

- 4-connectivity, 8-connectivity, ...

0	0	0	0	0	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	1	0	0	0	0
0	1	1	0	0	1	0	1	1	0
0	1	0	1	0	1	1	0	1	0
0	1	0	1	0	1	0	0	1	0
0	1	0	1	0	1	0	0	1	0
0	1	1	0	0	1	0	0	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	1	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	2	2	0	0	1	0	3	3	0
0	2	0	4	0	1	1	0	3	0
0	2	0	4	0	1	0	0	3	0
0	2	0	4	0	1	0	0	3	0
0	2	2	0	0	1	0	0	3	0
0	0	0	0	0	0	0	0	0	0

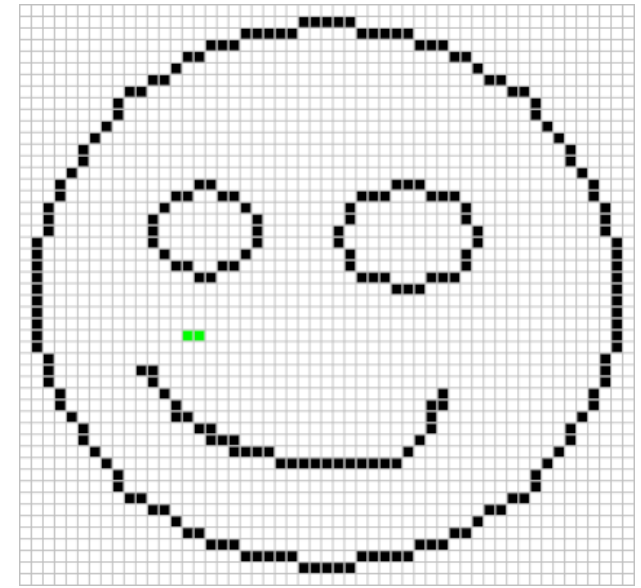
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	2	2	0	0	1	0	1	1	0
0	2	0	2	0	1	1	0	1	0
0	2	0	2	0	1	0	0	1	0
0	2	0	2	0	1	0	0	1	0
0	2	2	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0	0	0

Connected component

Flood-fill (범람채움?) algorithm

```
// b(j, i) - 이진 영상  
// I(j, i) - 결과 값(Flood Fill)  
// I의 경계를 0으로, 내부는 -1로 설정해준다.
```

```
label = 1;  
for(i = 1 to M - 2) {  
    for(i = 1 to N - 2) {  
        if(I(j, i) = -1) { // 만약 I가 labeling 되지 않은 상태라면  
            flood_fill4(I, j, i, label); label++; // 다음으로 번호로 넘어가기  
        }  
    }  
}  
function flood_fill4(I, j, i, label) {  
    if(I(j, i) = -1) {  
        I(j, i) = label; // labeling  
        flood_fill4(I, j, i + 1, label); // 오른쪽  
        flood_fill4(I, j - 1, i, label); // 위  
        flood_fill4(I, j, i - 1, label); // 왼쪽  
        flood_fill4(I, j + 1, i, label); // 아래  
    }  
}
```

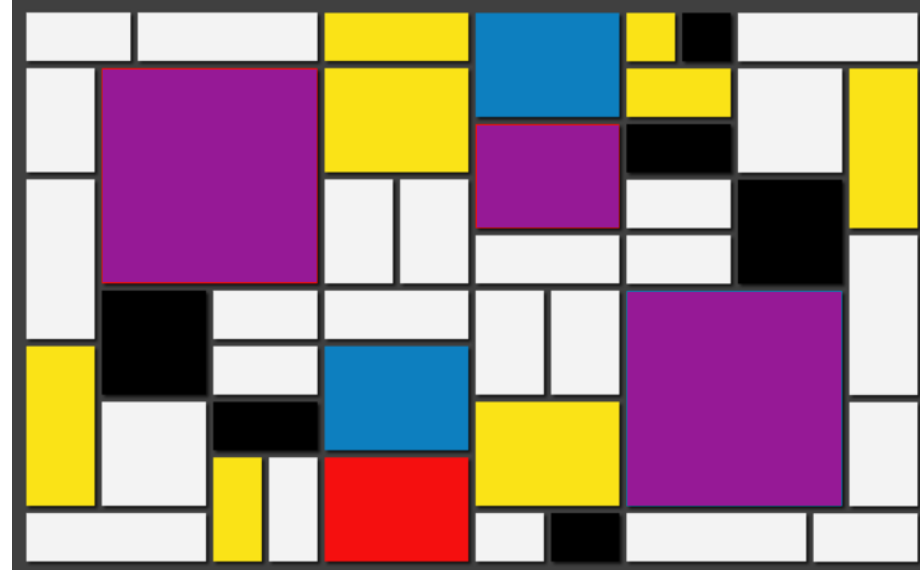
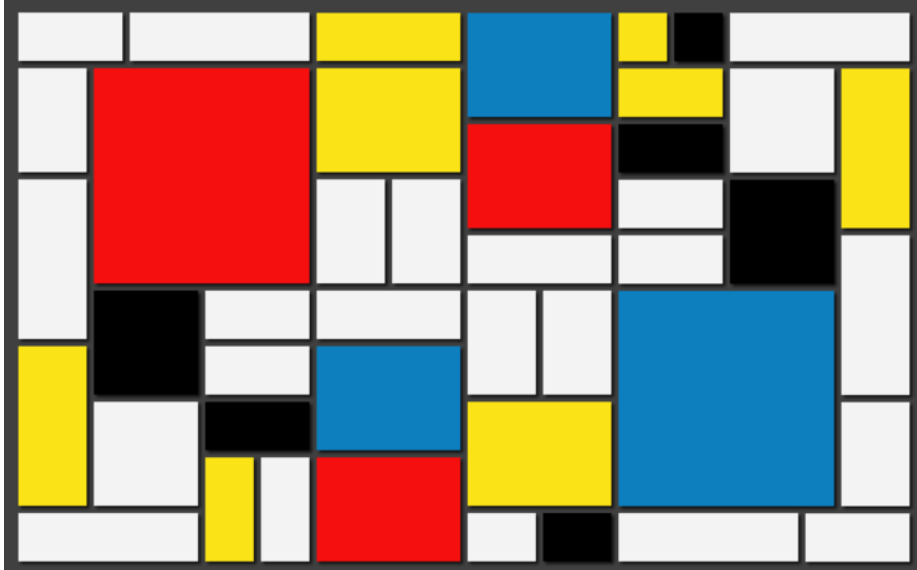


Connected component

Flood-fill (범람채움?) algorithm

- 1_4_floodFill.py

```
retval = cv2.floodFill(src, mask, seed, newVal, loDiff, upDiff)
```



End of slide