예를 들어 어떤 영상의 히스토그램에 Red 색이 많다는 뜻
이 영상을 equalization 하게 되면 효과
윤곽이 정확하게 보인다.

# OpenCV 3차시

ㅇ Histogram Equalization

이미지의 픽셀값 분포가 한 부분에 치중되지 않도록 하는 것이 Histogram Equalizaton이다. 이러한 처리를 거치게 되면, 픽셀들 간의 값차이가 일반적으로 증가하여, 보이지 않던 부분이 보이게 된다.

```
7   #define SCALE 0.2
8
9   int main(int argc, char** argv)
10  {
11      if (argc != 2)
12      {
13          cout << " Provide image name to read" << endl;
14          return -1;
15      }
16
17      Mat inputImg, dispImg, disp2Img;
18      Mat hlsImg;
19
20      inputImg = imread(argv[1], IMREAD_COLOR); // read in BGR format
21
22      if (inputImg.data == 0)
23      {
24          cout << "Unable to read " << argv[1] << endl;
25          return 0;
26      }
27
28      // inputImg is too large to be shown.
29      // use scaled-down dispImg instead just for display
30      // use inputImg for the histogram calculation
31      resize(inputImg, dispImg, Size(), SCALE, SCALE, CV_INTER_AREA);
32
33      cvtColor(dispImg, hlsImg, COLOR_BGR2HLS); // HLS 컬러 모델로 변경
34
35      vector<Mat> channels(hlsImg.channels());
36      split(hlsImg, channels);
37      equalizeHist(channels[1], channels[1]); // [1] = L (밝기) 밝기에 대해서 equalize
38      merge(channels, hlsImg);// 다시 합치기
39      cvtColor(hlsImg, disp2Img, COLOR_HLS2BGR);
40
41      imshow("Original", dispImg);
42      imshow("After luminacense equalization", disp2Img);
43
44      waitKey(0);
45      return 0;
46  }
```

함수 split(), merge(), equalizeHist() 의 documentation을
OpenCV 사이트에서 찾아서 확인해 보자.
* Histogram Equalization 후 변화를 그래프로 그려보자.

- gray 이미지 각 픽셀 마다 가질 수 있는단계 255 ( 0 검정 ~ 255 흰색)
- Binary Image 단계 2개 ( 0 ~ 1)

ㅇ Binary Image로 만들기

컬러이미지는 데이터량이 크기 때문에, 이를 gray 혹은 binary로 바꾸어 처리하는 것이 유리하다. Binary image는 픽셀값이 '흑' 혹은 '백'이다.

```
7   #define SCALE 0.2
8
9   int main(int argc, char** argv)
10  {
11      if (argc != 2)
12      {
13          cout << " Provide image name to read" << endl;
14          return -1;
15      }
16
17      Mat inputImg, displmg;
18      Mat grayImg, binImg;
19
20      inputImg = imread(argv[1], IMREAD_COLOR); // read in BGR format
21
22      if (inputImg.data == 0)
23      {
24          cout << "Unable to read " << argv[1] << endl;
25          return 0;
26      }
27
28      // inputImg is too large to be shown.
29      // use scaled-down displmg instead just for display
30      // use inputImg for the histogram calculation
31      resize(inputImg, displmg, Size(), SCALE, SCALE, CV_INTER_AREA);
32
33      cvtColor(displmg, grayImg, COLOR_BGR2GRAY);// color -> gray -> binary
34
35      threshold(grayImg, binImg, 128, 255, THRESH_BINARY);// (입력, 출력, 기준값, 설정할 픽셀의 값, 값을 설정하는 방법 (기준값 보다 크면 흰색))
36
37      imshow("Original", displmg);
38      imshow("Gray", grayImg);
39      imshow("Binary", binImg);
40
41      waitKey(0);
42      return 0;
43  }
```


128 값이 작아질수록 흰색이 많아짐
126 값을 어떻게 결정하겠느냐?

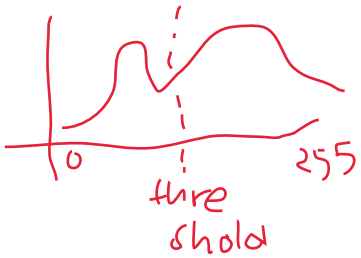
ㅇ 실험

함수 threshold()의 세 번째 인수 128을 10 혹은 255로 바꾸면 어떤 일이 발생하는지 확인해 보자.

그리고, THRESH_BINARY 외에 사용할 수 있는 인자가 무엇인지 확인해 보자.

THRESH_OTSU는 어떠한 역할을 하는지 알아보자.

color->gray->히스토그램
봉우리 가운데로 잘라는거



ㅇ Adaptive thresholding

사진 전체에 대해서 하나의 기준값으로 흑백을 결정하지 않고, 주변 픽셀들의 평균값을 기준으로 결정하는 방식.

```
7   #define SCALE 0.2
8
9   int main(int argc, char** argv)
10  {
11      if (argc != 2)
12      {
13          cout << " Provide image name to read" << endl;
14          return -1;
15      }
16
17      Mat inputImg, displmg;
18      Mat grayImg, binImg;
19
20      inputImg = imread(argv[1], IMREAD_COLOR); // read in BGR format
21
22      if (inputImg.data == 0)
23      {
24          cout << "Unable to read " << argv[1] << endl;
25          return 0;
26      }
27
28      // inputImg is too large to be shown.
29      // use scaled-down displmg instead just for display
30      // use inputImg for the histogram calculation
31      resize(inputImg, displmg, Size(), SCALE, SCALE, CV_INTER_AREA);
32
33      cvtColor(displmg, grayImg, COLOR_BGR2GRAY);
34
35      //threshold(grayImg, binImg, 128, 255, THRESH_OTSU);
36      adaptiveThreshold(grayImg, binImg, 255, ADAPTIVE_THRESH_GAUSSIAN_C, THRESH_BINARY, 5, 3.0);
37
38      imshow("Original", displmg);
39      imshow("Gray", grayImg);
40      imshow("Binary", binImg);
41
42      waitKey(0);
43      return 0;
44  }
```

어떤 픽셀을 검정? 흰색인지 기준값이 아니라
주변 픽셀들이랑 평균을 내고 평균보다 크면 흰색, 작으면 검정
흰색을 쓸때는 255 를 쓴다.
4번째 인수 : 나한테 가중치를 젱ㄹ많이
5번째 인수 : 평균보다 크면 흰색, 작으면 검정
6번째 인수 : 평균을 낼 픽셀 5x5
7번째 인수 : 기준값?

라인 36의 adaptiveThreshold( )에서

ADAPTIVE_THRESH_GAUSSIAN_C가 의미하는 바는?

5가 의미하는 바는?

3.0이 의미하는 바는?

OpenCV의 documentation에서 해당 함수의 설명을 찾아보자.

ㅇ Morphology – dilation

Binary image에서 흰색 영역을 확장시킨다.

```
7   #define SCALE 0.2
8
9   int main(int argc, char** argv)
10  {
11      if (argc != 2)
12      {
13          cout << " Provide image name to read" << endl;
14          return -1;
15      }
16
17      Mat inputImg, dispImg;
18      Mat grayImg, binImg;
19
20      inputImg = imread(argv[1], IMREAD_COLOR); // read in BGR format
21
22      if (inputImg.data == 0)
23      {
24          cout << "Unable to read " << argv[1] << endl;
25          return 0;
26      }
27
28      // inputImg is too large to be shown.
29      // use scaled-down dispImg instead just for display
30      // use inputImg for the histogram calculation
31      resize(inputImg, dispImg, Size(), SCALE, SCALE, CV_INTER_AREA);
32
33      cvtColor(dispImg, grayImg, COLOR_BGR2GRAY);
34
35      threshold(grayImg, binImg, 128, 255, THRESH_BINARY);
36
37      Mat dilatedImg; // output 저장 이미지
38      Mat kernel(3, 3, CV_8U); // 3X3 안에 흰색이 하나라도 있음 내가 흰색이 된다.노이즈를 없앨 때
39      dilate(binImg, dilatedImg, kernel); // (입력항상 binary,출력, kernel)      주로 사용한다
40
41      imshow("Binary", binImg);
42      imshow("Dilated", dilatedImg);
43
44      waitKey(0);
45      return 0;
46  }
```

라인 38의  kernel을 3x3에서 5x5, 7x7로 했을 때 어떤 점이 달라지는지 확인해 보자.

○ Morphology - erosion

Binary image에서 흰색 영역을 축소시킨다.

```
 7   #define SCALE 0.2
 8
 9   int main(int argc, char** argv)
10   {
11       if (argc != 2)
12       {
13           cout << " Provide image name to read" << endl;
14           return -1;
15       }
16
17       Mat inputImg, dispImg;
18       Mat grayImg, binImg;
19
20       inputImg = imread(argv[1], IMREAD_COLOR); // read in BGR format
21
22       if (inputImg.data == 0)
23       {
24           cout << "Unable to read " << argv[1] << endl;
25           return 0;
26       }
27
28       // inputImg is too large to be shown.
29       // use scaled-down dispImg instead just for display
30       // use inputImg for the histogram calculation
31       resize(inputImg, dispImg, Size(), SCALE, SCALE, CV_INTER_AREA);
32
33       cvtColor(dispImg, grayImg, COLOR_BGR2GRAY);
34
35       threshold(grayImg, binImg, 128, 255, THRESH_BINARY);
36
37       Mat erodedImg
38       Mat kernel(3, 3, CV_8U);
39       erode(binImg,erodedImg, kernel);// 내 주변이 다 흰색이어야 흰색이 된다.
40                                       아니면 검정색이된다.
41       imshow("Binary", binImg);       흰색이 노이즈일 경우 주로 사용한다.
42       imshow("Dilated", dilatedImg);
43
44       waitKey(0);
45       return 0;
46   }
```

라인 38에서 kernel 크기를 3x3 외의 크기로 변화시켜 보자.

○ Morphology – erosion and dilation // 이미지를 깔끔하게 적용 간으

Binary image에서 흰색 영역을 깍아낸 후, 다시 키운다. 이 과정에서 노이즈가 제거된다.

```
7   #define SCALE 0.2
8
9   int main(int argc, char** argv)
10  {
11      if (argc != 2)
12      {
13          cout << " Provide image name to read" << endl;
14          return -1;
15      }
16
17      Mat inputImg, displmg;
18      Mat grayImg, binImg;
19
20      inputImg = imread(argv[1], IMREAD_COLOR); // read in BGR format
21
22      if (inputImg.data == 0)
23      {
24          cout << "Unable to read " << argv[1] << endl;
25          return 0;
26      }
27
28      // inputImg is too large to be shown.
29      // use scaled-down displmg instead just for display
30      // use inputImg for the histogram calculation
31      resize(inputImg, displmg, Size(), SCALE, SCALE, CV_INTER_AREA);
32
33      cvtColor(displmg, grayImg, COLOR_BGR2GRAY);
34
35      threshold(grayImg, binImg, 128, 255, THRESH_BINARY);
36
37      Mat openedImg;
38      Mat kernel(3, 3, CV_8U);
39      morphologyEx(binImg, openedImg, MORPH_OPEN, kernel);
40                  (입력, 출력, erosed->dilated,kernel)
41      imshow("Binary", binImg);
42      imshow("Opened", openedImg);
43
44      waitKey(0);
45      return 0;
46  }
```

라인 38에서 kernel 크기를 3x3 외의 크기로 변화시켜 보자.

ㅇ Morphology - dilation and erosing

Binary image에서 흰색 영역이 부각된 후, 노이즈가 제거된다.

```
7   #define SCALE 0.2
8
9   int main(int argc, char** argv)
10  {
11      if (argc != 2)
12      {
13          cout << " Provide image name to read" << endl;
14          return -1;
15      }
16
17      Mat inputImg, dispImg;
18      Mat grayImg, binImg;
19
20      inputImg = imread(argv[1], IMREAD_COLOR); // read in BGR format
21
22      if (inputImg.data == 0)
23      {
24          cout << "Unable to read " << argv[1] << endl;
25          return 0;
26      }
27
28      // inputImg is too large to be shown.
29      // use scaled-down dispImg instead just for display
30      // use inputImg for the histogram calculation
31      resize(inputImg, dispImg, Size(), SCALE, SCALE, CV_INTER_AREA);
32
33      cvtColor(dispImg, grayImg, COLOR_BGR2GRAY);
34
35      threshold(grayImg, binImg, 128, 255, THRESH_BINARY);
36
37      Mat closedImg;
38      Mat kernel(3, 3, CV_8U);
39      morphologyEx(binImg, closedImg, MORPH_CLOSE, kernel);
40
41      imshow("Binary", binImg);
42      imshow("Closed", closedImg);
43
44      waitKey(0);
45      return 0;
46  }
```

라인 38에서 kernel 크기를 3x3 외의 크기로 변화시켜 보자.