OpenCV 1차시

ㅇ 이미지 파일을 읽어들여 화면에 표시하는 프로그램

```cpp
#include <opencv2/opencv.hpp>  .hpp -> cpp용 헤더 파일
#include <iostream>

using namespace cv; // cv (computer vision)
using namespace std;

int main(int argc, char** argv) // command line에서 argument
{                               // argument count, argument value
                                // 이미지 이름을 command 라인에 주기 위해서 설정
                                // ex)  >> a.exe img.jpg  -> argument 두개
    if (argc != 2)
    {
        cout << " Provide image name to read" << endl;
        return -1;
    }

    Mat inputImg; // Mat (Matrix Class)

    inputImg = imread(argv[1], IMREAD_UNCHANGED);  // image read

    if (inputImg.empty()) // 읽고 나서 제대로 읽었는지 ★반드시★ 확인
    {                     // 비어 있으면 제대로 못 열었다
        cout << "Unable to open file " << endl;
        return 0;
    }

          윈도우 이름     Mat
    imshow("Original", inputImg);  // 화면에 띄울꺼

    waitKey(0); // 밀리세컨 단위 0이면? 키를 누를 때 까지 계속 그대로 있음
    return 0;
}
```

.hpp : c++ 헤더파일임을 의미 docs.opencv.org/3.4.1

namespace cv : OpenCV의 대부분 클래스가 정의된 namespace

Mat: 사진 이미지를 저장하기 위한 클래스

§ imread()

```
Mat cv::imread ( const String &  filename,
                 int             flags = IMREAD_COLOR
                                  // default argument
               )
```

§ empty()

bool cv::Mat::empty ( ) const

Returns true if the array has no elements.

ㅇ 이미지 크기 조절

```
7   int main(int argc, char** argv)
8   {
9
10      if (argc != 2)
11      {
12          cout << " Provide image name to read" << endl;
13          return -1;
14      }
15
16      Mat inputImg;
17
18      inputImg = imread(argv[1], IMREAD_UNCHANGED);
19
20      if (inputImg.empty())
21      {
22          cout << "Unable to open file " << endl;
23          return 0;
24      }
25
26      Mat resizedOne; // 새로운 객체 생성          가로 세로 2배씩 -> 4배 증가된다.
27      resize(inputImg, resizedOne, Size(), 2.0, 2.0);
28              => input image ,새로만들거, 비어있는 사이즈, 세로,가로 (변경할 사이즈)
29      imshow("Original", resizedOne);
30
31      waitKey(0);
32      return 0;
33  }
```

§ resize()

```
void cv::resize ( InputArray  src,
                  OutputArray dst,
                  Size        dsize,
                  double      fx = 0,
                  double      fy = 0,
                  int         interpolation = INTER_LINEAR
                )
```

이미지 크기를 절반으로 줄이기 위해서는 함수 resize( )의 4번째와 5번째의 인수를 0.5로 설정해야 한다. 만약 옆으로 퍼진 형태의 사진을 만들기 위해서는, 4번째를 줄이고, 5번째를 크게하면 되다. 왜냐하면, 4번째는 세로, 5번째는 가로를 의미하기 때문이다.

ㅇ 컬러이미지를 Gray 이미지로 전환

```
7   int main(int argc, char** argv)
8   {
9
10      if (argc != 2)
11      {
12          cout << " Provide image name to read" << endl;
13          return -1;
14      }
15
16      Mat inputImg;
17
18      inputImg = imread(argv[1], IMREAD_UNCHANGED);
19
20      if (inputImg.empty())
21      {
22          cout << "Unable to open file " << endl;
23          return 0;
24      }
25                                                  BGR 순서로 컬러 저장한다.
26      Mat grayOne;
27      cvtColor(inputImg, grayOne, COLOR_BGR2GRAY);
28              인풋이미지, 새로 할거, BGR->GRAY로 변경
29      imshow("Original", grayOne);
30
31      waitKey(0);
32      return 0;
33  }
```

§ cvtColor() // 이미지의 컬러 바꾸는거

```
void cv::cvtColor ( InputArray  src,
                    OutputArray dst,
                    int         code,
                    int         dstCn = 0
                  )
```

code는 컬러모델을 바꾸는 방식을 의미한다. Mat는 BGR (Blue, Green, Red) 형태로 이미지를 저장하기 때문에, 이를 Gray로 바꾸기 위해서, COLOR_BGR2GRAY code를 사용한다.

<사람 피부영역 구별>
1. 이미지 읽기
2. HLS 컬러 모델로 변경
3. 아래 세가지 조건을 만족 시키면 피부영역

ㅇ 사람 피부영역 구별하기 // HLS (Hue Luminance Saturation)

사람의 피부영역은 HLS color model을 이용하면 쉽게 찾아낼 수 있다. 기존 연구에서는 다음 조건을 만족시키는 픽셀이 피부영역에 해당한다고 한다.

(Saturation >= 50) AND
(0.5 < Luminance/Saturation < 3.0) AND
(Hue <= 14 or Hue >= 165)

이를 이용하려면, 이미지를 HLS color model 이미지로 바꾼 후, 픽셀 별로 위 조건을 충족시키는지 확인해야 한다. 충족하는 픽셀들은 그대로 두지만, 아닌 픽셀들은 모두 검정색으로 바꾼다. 결과를 확인하기 위해서는 이미지를 화면에 표시하려면 다시 BGR 모델로 바꾸어야 한다.

```
26      Mat hlsOne;  // HLS Color로 바꾸기
27      cvtColor(inputImg, hlsOne, COLOR_BGR2HLS);// BGR->HLS
28      // 모든 픽셀 하나씩 값을 뽑아낸다.
29      for (int row = 0; row < hlsOne.size().height; row++) // 이미지에서 가로는 세로
30      {
31          for (int col = 0; col < hlsOne.size().width; col++) // 가로
32          {
33              uchar H = hlsOne.at<Vec3b>(row, col)[0]; // row,col에 있는 픽셀 하나를 뽑는다.
34              uchar L = hlsOne.at<Vec3b>(row, col)[1];     -> 값 세개가 뭉쳐 있음 그래서 Vec3b 사용
35              uchar S = hlsOne.at<Vec3b>(row, col)[2];
36
37              double LS_ratio = ((double)L) / ((double)S);
38              bool skin_pixel = (S >= 50) && (LS_ratio > 0.5) && (LS_ratio < 3.0) && ((H <= 14) || (H >= 165));
39
40              if (skin_pixel == false) // 피부가 아닌 부분
41              {
42                  hlsOne.at<Vec3b>(row, col) = Vec3b(0, 0, 0); // 그 부분은 검정으로`
43              }
44          }
45      }
46      Mat rgbImg;
47      cvtColor(hlsOne, rgbImg, COLOR_HLS2BGR);
48                                  // imshow는 RGB만 보여줄 수 있기때문에  HLS에서 BGR로 변경후
49      imshow("Original", rgbImg);  함수 실행
```

Vec3b
H | L | S

*at
: 인수가 없음 템플릿

*vec3b
: RGB,HLS 넣는

Vec3b는 3개 바이트의 집합을 나타내는 데이터 형이다.
Mat 클래스의 함수 at( )는 row와 col로 주어진 좌표의 픽셀값을 read/write할 수 있다.

인물이 나온 사진을 이용해서 위 프로그램을 실행해보자.

ㅇ Red Eye detection

실험결과에 의하면 인물사진에 나온 red-eye (빨간 눈)는 다음과 같은 조건에 해당하는 픽셀영역이라고 한다.
Luminance >= 64 AND
Saturation >= 100 AND
0.5 < Luminance/Saturation < 1.5 AND
(Hue <= 7 OR Hue >= 162)

```
26      Mat hlsOne;
27      cvtColor(inputImg, hlsOne, COLOR_BGR2HLS);
28
29      for (int row = 0; row < hlsOne.size().height; row++)
30      {
31          for (int col = 0; col < hlsOne.size().width; col++)
32          {
33              uchar H = hlsOne.at<Vec3b>(row, col)[0];
34              uchar L = hlsOne.at<Vec3b>(row, col)[1];
35              uchar S = hlsOne.at<Vec3b>(row, col)[2];
36
37              double LS_ratio = ((double)L) / ((double)S);
38              bool red_eye = (L >= 64) && (S >= 100) && (LS_ratio > 0.5) && (LS_ratio < 1.5) && ((H <= 7) || (H >= 162));
39
40              if (red_eye == true)
41              {
42                  hlsOne.at<Vec3b>(row, col) = Vec3b(0, 0, 0);
43              }
44          }
45      }
46      Mat rgbImg;
47      cvtColor(hlsOne, rgbImg, COLOR_HLS2BGR);
48
49      imshow("Original", rgbImg);
```

red eye가 발생한 사진을 이용해서, 해당영역이 제대로 검정색으로 변하는지 확인해 보자.