Digital Image Processing

LabWork #CV3

Professor : Jin-Woo Jung

* Using the given CFilter class, design a program that can process the following :

1. Convert an image to a gray scale image (Input : lenna.jpg, jenny.jpg)

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
|  |  |  |  |

| code |
| --- |
| 1. lenna.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("lenna.png", IMREAD\_COLOR); // lenna.png 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_BGR2GRAY); //image 변수에 저장된 이미지를 convert 변수에 8bit gray-scale로 변환하여 저장한다. 이때 3번째 파라미터에는 HSV로 변환하는 COLOR\_BGR2HSV, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.  imshow("image", convert); // convert된 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  }   1. jenny.jpg   #include <opencv2/opencv.hpp>  #include "Pixel.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_BGR2GRAY); //image 변수에 저장된 이미지를 convert 변수에 8bit gray-scale로 변환하여 저장한다. 이때 3번째 파라미터에는 HSV로 변환하는 COLOR\_BGR2HSV, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.  imshow("image", convert); // convert된 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } |

1. Make a blurred image using the gray scale image

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
|  |  |  |  |
| code | | | |
| 1. lenna.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CFilter filtering; //filtering 작업을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("lenna.png", IMREAD\_COLOR); // lenna.png 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_BGR2GRAY); //image 변수에 저장된 이미지를 convert 변수에 8bit gray-scale로 변환하여 저장한다. 이때 3번째 파라미터에는 HSV로 변환하는 COLOR\_BGR2HSV, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.  Mat blurred = filtering.GS\_blurring(convert, 0, 2); //blur 처리를 한 이미지를 저장한다. 이때 사용된 함수의 첫번째 파라미터는 source image, 두번째 파라미터는 어떤 방식으로 blur 처리를 할 것인지에 대한 option, 세번째 파라미터는 mask의 크기이다.  //두 번째 파라미터의 숫자에 따라 0 : 직접 구현한 blurring 함수 사용, 1 : opencv에서 제공하는 blur 함수, 2 : opencv에서 제공하는 GaussianBlur 함수, 3 : opencv에서 제공하는 medianBlur 함수, 4 : opencv에서 제공하는 bilateralFilter 함수가 실행된다.  //세 번째 파라미터의 숫자에 따라 mask의 크기는 0 : 3 \* 3, 1 : 5 \* 5, 2 : 7 \* 7이 된다.  imshow("image", blurred); // blur된 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  }   1. jenny.jpg   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CFilter filtering; //filtering 작업을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_BGR2GRAY); //image 변수에 저장된 이미지를 convert 변수에 8bit gray-scale로 변환하여 저장한다. 이때 3번째 파라미터에는 HSV로 변환하는 COLOR\_BGR2HSV, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.  Mat blurred = filtering.GS\_blurring(convert, 0, 2); //blur 처리를 한 이미지를 저장한다. 이때 사용된 함수의 첫번째 파라미터는 source image, 두번째 파라미터는 어떤 방식으로 blur 처리를 할 것인지에 대한 option, 세번째 파라미터는 mask의 크기이다.  //두 번째 파라미터의 숫자에 따라 0 : 직접 구현한 blurring 함수 사용, 1 : opencv에서 제공하는 blur 함수, 2 : opencv에서 제공하는 GaussianBlur 함수, 3 : opencv에서 제공하는 medianBlur 함수, 4 : opencv에서 제공하는 bilateralFilter 함수가 실행된다.  //세 번째 파라미터의 숫자에 따라 mask의 크기는 0 : 3 \* 3, 1 : 5 \* 5, 2 : 7 \* 7이 된다.  imshow("image", blurred); // blur된 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } | | | |

1. Subtract the blurred image from original image

|  |  |  |
| --- | --- | --- |
| **Input** | | **Output** |
|  |  |  |
|  |  |  |
| code | | |
| 1. lenna.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CFilter filtering; //filtering 작업을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("lenna.png", IMREAD\_COLOR); // lenna.png 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_BGR2GRAY); //image 변수에 저장된 이미지를 convert 변수에 8bit gray-scale로 변환하여 저장한다. 이때 3번째 파라미터에는 HSV로 변환하는 COLOR\_BGR2HSV, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.  Mat blurred = filtering.GS\_blurring(convert, 0, 2); //blur 처리를 한 이미지를 저장한다. 이때 사용된 함수의 첫번째 파라미터는 source image, 두번째 파라미터는 어떤 방식으로 blur 처리를 할 것인지에 대한 option, 세번째 파라미터는 mask의 크기이다.  //두 번째 파라미터의 숫자에 따라 0 : 직접 구현한 blurring 함수 사용, 1 : opencv에서 제공하는 blur 함수, 2 : opencv에서 제공하는 GaussianBlur 함수, 3 : opencv에서 제공하는 medianBlur 함수, 4 : opencv에서 제공하는 bilateralFilter 함수가 실행된다.  //세 번째 파라미터의 숫자에 따라 mask의 크기는 0 : 3 \* 3, 1 : 5 \* 5, 2 : 7 \* 7이 된다.  Mat subtract = processing.GS\_subtract\_image(convert, blurred); //원본 이미지에서 blur 작업을 거친 이미지를 subtract 한다. 이 작업을 수행하면 edge만 남을 것이다.  imshow("image", subtract); // edge만 남은 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  }   1. jenny.jpg   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CFilter filtering; //filtering 작업을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_BGR2GRAY); //image 변수에 저장된 이미지를 convert 변수에 8bit gray-scale로 변환하여 저장한다. 이때 3번째 파라미터에는 HSV로 변환하는 COLOR\_BGR2HSV, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.  Mat blurred = filtering.GS\_blurring(convert, 0, 2); //blur 처리를 한 이미지를 저장한다. 이때 사용된 함수의 첫번째 파라미터는 source image, 두번째 파라미터는 어떤 방식으로 blur 처리를 할 것인지에 대한 option, 세번째 파라미터는 mask의 크기이다.  //두 번째 파라미터의 숫자에 따라 0 : 직접 구현한 blurring 함수 사용, 1 : opencv에서 제공하는 blur 함수, 2 : opencv에서 제공하는 GaussianBlur 함수, 3 : opencv에서 제공하는 medianBlur 함수, 4 : opencv에서 제공하는 bilateralFilter 함수가 실행된다.  //세 번째 파라미터의 숫자에 따라 mask의 크기는 0 : 3 \* 3, 1 : 5 \* 5, 2 : 7 \* 7이 된다.  Mat subtract = processing.GS\_subtract\_image(convert, blurred); //원본 이미지에서 blur 작업을 거친 이미지를 subtract 한다. 이 작업을 수행하면 edge만 남을 것이다.  imshow("image", subtract); // edge만 남은 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } | | |

1. Add the original image with the image from above step (iii)

|  |  |  |
| --- | --- | --- |
| **Input** | | **Output** |
|  |  |  |
|  |  |  |
| code | | |
| 1. lenna.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CFilter filtering; //filtering 작업을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("lenna.png", IMREAD\_COLOR); // lenna.png 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_BGR2GRAY); //image 변수에 저장된 이미지를 convert 변수에 8bit gray-scale로 변환하여 저장한다. 이때 3번째 파라미터에는 HSV로 변환하는 COLOR\_BGR2HSV, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.  Mat blurred = filtering.GS\_blurring(convert, 0, 2); //blur 처리를 한 이미지를 저장한다. 이때 사용된 함수의 첫번째 파라미터는 source image, 두번째 파라미터는 어떤 방식으로 blur 처리를 할 것인지에 대한 option, 세번째 파라미터는 mask의 크기이다.  //두 번째 파라미터의 숫자에 따라 0 : 직접 구현한 blurring 함수 사용, 1 : opencv에서 제공하는 blur 함수, 2 : opencv에서 제공하는 GaussianBlur 함수, 3 : opencv에서 제공하는 medianBlur 함수, 4 : opencv에서 제공하는 bilateralFilter 함수가 실행된다.  //세 번째 파라미터의 숫자에 따라 mask의 크기는 0 : 3 \* 3, 1 : 5 \* 5, 2 : 7 \* 7이 된다.  Mat subtract = processing.GS\_subtract\_image(convert, blurred); //원본 이미지에서 blur 작업을 거친 이미지를 subtract 한다. 이 작업을 수행하면 edge만 남을 것이다.  Mat addition = processing.GS\_add\_image(convert, subtract); //원본 이미지에 위에서 subtract 작업을 거친 이미지, 즉 edge만 남은 이미지를 더한다. 이 작업은 이미지를 sharpening하는 작업이다.  imshow("image", addition); // sharpening된 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  }   1. jenny.jpg   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CFilter filtering; //filtering 작업을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_BGR2GRAY); //image 변수에 저장된 이미지를 convert 변수에 8bit gray-scale로 변환하여 저장한다. 이때 3번째 파라미터에는 HSV로 변환하는 COLOR\_BGR2HSV, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.  Mat blurred = filtering.GS\_blurring(convert, 0, 2); //blur 처리를 한 이미지를 저장한다. 이때 사용된 함수의 첫번째 파라미터는 source image, 두번째 파라미터는 어떤 방식으로 blur 처리를 할 것인지에 대한 option, 세번째 파라미터는 mask의 크기이다.  //두 번째 파라미터의 숫자에 따라 0 : 직접 구현한 blurring 함수 사용, 1 : opencv에서 제공하는 blur 함수, 2 : opencv에서 제공하는 GaussianBlur 함수, 3 : opencv에서 제공하는 medianBlur 함수, 4 : opencv에서 제공하는 bilateralFilter 함수가 실행된다.  //세 번째 파라미터의 숫자에 따라 mask의 크기는 0 : 3 \* 3, 1 : 5 \* 5, 2 : 7 \* 7이 된다.  Mat subtract = processing.GS\_subtract\_image(convert, blurred); //원본 이미지에서 blur 작업을 거친 이미지를 subtract 한다. 이 작업을 수행하면 edge만 남을 것이다.  Mat addition = processing.GS\_add\_image(convert, subtract); //원본 이미지에 위에서 subtract 작업을 거친 이미지, 즉 edge만 남은 이미지를 더한다. 이 작업은 이미지를 sharpening하는 작업이다.  imshow("image", addition); // sharpening된 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } | | |

* Using the given CColor and CPixel class, design a program with the following steps :

1.

* 1. Split the RGB image to R image, G image, B image. (Input : jenny.jpg)

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | | |
|  |  |  |  |
| code | | | |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CColor color; //color 모델 작업을 위한 클래스를 선언한다.  Mat rgb[3]; //원본 이미지를 R, G, B 프레임 별로 분리 후 저장하기 위한 배열을 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  color.GS\_splitRGB(image, rgb); //컬러 이미지를 R, G, B 각 프레임별로 분리한다. 이때 첫번째 파라미터는 분리할 원본 이미지, 두번째 파라미터는 분리후 저장할 배열이다.  for (auto x : rgb)  {  imshow("image", x); //분리된 이미지가 배열에 저장되어 있기 때문에 for문을 이용하여 분리된 이미지를 화면에 출력한다.  //이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey();//키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  }  return 0;  } | | | |

* 1. Make equalized images using histogram equalization : R’ image, G’ image, B’ image

|  |  |  |
| --- | --- | --- |
| **Input** | | |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Output** | | |
|  |  |  |
| code | | |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CColor color; //color 모델 작업을 위한 클래스를 선언한다.  Mat rgb[3]; //원본 이미지를 R, G, B 프레임 별로 분리 후 저장하기 위한 배열을 선언한다.  Mat equlization[3]; //평활화 작업을 거친 이미지를 저장하기 위한 배열을 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  color.GS\_splitRGB(image, rgb); //컬러 이미지를 R, G, B 각 프레임별로 분리한다. 이때 첫번째 파라미터는 분리할 원본 이미지, 두번째 파라미터는 분리후 저장할 배열이다.  for (int i = 0; i < 3; i++)  {  equlization[i] = processing.GS\_histeq(rgb[i]); //for문을 이용하여 rgb 배열에 저장된 이미지를 차례대로 평활화한다.  }  for (auto x : equlization)  {  imshow("image", x); //평활화된 이미지가 배열에 저장되어있기 때문에 for문을 이용하여 분리된 이미지를 화면에 출력한다.  //이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey();//키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  }  return 0;  } | | |

* 1. Composite R’ image, G’ image, and B’ image : R’G’B’ image

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | | **Output** |
|  |  |  |  |
| code | | | |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CColor color; //color 모델 작업을 위한 클래스를 선언한다.  Mat rgb[3]; //원본 이미지를 R, G, B 프레임 별로 분리 후 저장하기 위한 배열을 선언한다.  Mat equlization[3]; //평활화 작업을 거친 이미지를 저장하기 위한 배열을 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  color.GS\_splitRGB(image, rgb); //컬러 이미지를 R, G, B 각 프레임별로 분리한다. 이때 첫번째 파라미터는 분리할 원본 이미지, 두번째 파라미터는 분리후 저장할 배열이다.  for (int i = 0; i < 3; i++)  {  equlization[i] = processing.GS\_histeq(rgb[i]); //for문을 이용하여 rgb 배열에 저장된 이미지를 차례대로 평활화한다.  }  Mat composite = color.GS\_compositeRGB(equlization); //평활화된 r, g, b 이미지를 다시 composite한다. 이때 함수의 파라미터는 r, g, b 이미지가 저장된 배열이다.  imshow("image", composite); //composite된 이미지를 화면에 출력한다.  //이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey();//키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } | | | |

2.

* 1. Convert RGB color image to HSV image (input : jenny.jpg)

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |
| code | |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CColor color; //color 모델 작업을 위한 클래스를 선언한다.  Mat convert; //input image를 hsv 이미지로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_RGB2HSV); //image 변수에 저장된 이미지를 convert 변수에 hsv로 변환하여 저장한다. 이때 3번째 파라미터에는 gray-scale로 변환하는 CV\_BGR2GRAY, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.    imshow("image", convert); //convert된 이미지를 화면에 출력한다.  //이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey();//키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } | |

* 1. Split the HSV image to H image, S image, V image.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | | |
|  |  |  |  |

| code |
| --- |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CColor color; //color 모델 작업을 위한 클래스를 선언한다.  Mat convert; //input image를 hsv 이미지로 변환하고 저장하기 위한 변수를 선언한다.  Mat hsv[3]; //h, s, v로 분리한 이미지를 저장하기 위한 배열을 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_RGB2HSV); //image 변수에 저장된 이미지를 convert 변수에 hsv로 변환하여 저장한다. 이때 3번째 파라미터에는 gray-scale로 변환하는 CV\_BGR2GRAY, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.    split(convert, hsv); // hsv이미지를 색상, 채도, 명암 이미지로 분리한다.  for (auto x : hsv)  {  imshow("image", x); //for문을 이용하여 배열에 저장된 색상, 채도, 명암 이미지를 화면에 출력한다.  //이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey();//키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  }  return 0;  } |

* 1. Make an equalized image using histogram equalization : V’ image

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |

| code |
| --- |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CColor color; //color 모델 작업을 위한 클래스를 선언한다.  Mat convert; //input image를 hsv 이미지로 변환하고 저장하기 위한 변수를 선언한다.  Mat equailzation; //평활화된 이미지를 저장하기 위한 변수를 선언한다.  Mat hsv[3]; //h, s, v로 분리한 이미지를 저장하기 위한 배열을 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_RGB2HSV); //image 변수에 저장된 이미지를 convert 변수에 hsv로 변환하여 저장한다. 이때 3번째 파라미터에는 gray-scale로 변환하는 CV\_BGR2GRAY, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.    split(convert, hsv); // hsv이미지를 색상, 채도, 명암 이미지로 분리한다.    equailzation = processing.GS\_histeq(hsv[2]); //hsv 공간에서 v, 즉 명암 이미지를 평활화한다. 이때 파라미터로 평활화하려는 이미지가 오는데, 명암 이미지는 hsv[2]에 존재한다.  imshow("image", equailzation); // 평활화된 이미지를 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } |

* 1. Composite H image, S image, and V’ image : HSV’ image

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | | **Output** |
|  |  |  |  |

| code |
| --- |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CColor color; //color 모델 작업을 위한 클래스를 선언한다.  Mat convert; //input image를 hsv 이미지로 변환하고 저장하기 위한 변수를 선언한다.  Mat equailzation; //평활화된 이미지를 저장하기 위한 변수를 선언한다.  Mat hsv[3]; //h, s, v로 분리한 이미지를 저장하기 위한 배열을 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_RGB2HSV); //image 변수에 저장된 이미지를 convert 변수에 hsv로 변환하여 저장한다. 이때 3번째 파라미터에는 gray-scale로 변환하는 CV\_BGR2GRAY, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.    split(convert, hsv); // hsv이미지를 색상, 채도, 명암 이미지로 분리한다.    equailzation = processing.GS\_histeq(hsv[2]); //hsv 공간에서 v, 즉 명암 이미지를 평활화한다. 이때 파라미터로 평활화하려는 이미지가 오는데, 명암 이미지는 hsv[2]에 존재한다.  hsv[2] = equailzation; // 평활화된 명암 이미지를 hsv 배열에 다시 저장한다.  merge(hsv, 3, convert); // h, s, v 별로 따로 분리된 이미지를 다시 하나로 합병한다.  //이때 첫번째 파라미터로 분리된 이미지가 저장된 배열, 두번째 파라미터는 이미지를 합치려는 개수, 세번째 파라미터는 합친 이미지를 저장할 변수이다.  imshow("image", convert); // merge된 이미지를 화면에 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } |

* 1. Convert the HSV’ image to RGB image : HSV’2RGB image

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |

| code |
| --- |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Filter.h"  #include "Color.h"  #include "Util.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CColor color; //color 모델 작업을 위한 클래스를 선언한다.  Mat convert; //input image를 hsv 이미지로 변환하고 저장하기 위한 변수를 선언한다.  Mat equailzation; //평활화된 이미지를 저장하기 위한 변수를 선언한다.  Mat hsv[3]; //h, s, v로 분리한 이미지를 저장하기 위한 배열을 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg 를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  cvtColor(image, convert, CV\_RGB2HSV); //image 변수에 저장된 이미지를 convert 변수에 hsv로 변환하여 저장한다. 이때 3번째 파라미터에는 gray-scale로 변환하는 CV\_BGR2GRAY, YCrCb로 변환하는 COLOR\_BGR2YCrCb, Luv로 변환하는 COLOR\_BGR2Luv 등등이 올 수 있다.    split(convert, hsv); // hsv이미지를 색상, 채도, 명암 이미지로 분리한다.    equailzation = processing.GS\_histeq(hsv[2]); //hsv 공간에서 v, 즉 명암 이미지를 평활화한다. 이때 파라미터로 평활화하려는 이미지가 오는데, 명암 이미지는 hsv[2]에 존재한다.  hsv[2] = equailzation; // 평활화된 명암 이미지를 hsv 배열에 다시 저장한다.  merge(hsv, 3, convert); // h, s, v 별로 따로 분리된 이미지를 다시 하나로 합병한다.  //이때 첫번째 파라미터로 분리된 이미지가 저장된 배열, 두번째 파라미터는 이미지를 합치려는 개수, 세번째 파라미터는 합친 이미지를 저장할 변수이다.  cvtColor(convert, image, CV\_HSV2RGB); //convert 변수에 저장된 이미지를 image 변수에 rgb로 변환하여 저장한다.  imshow("image", image); // RGB로 변환된 이미지를 화면에 출력한다. 이때 첫번째 파라미터는 이미지를 출력하기 위해 띄우는 창의 제목표시줄에 출력할 제목이고, 두 번째 파라미터는 출력하기 위한 이미지를 가리키는 변수이다.  waitKey(); //키 입력이 있을 때까지 창을 유지하기 위해 waitkey() 함수를 이용한다.  return 0;  } |

3. Compare R’G’B’ image and HSV’2RGB image

텍스트, 사람, 보드이(가) 표시된 사진

자동 생성된 설명 텍스트, 사람, 드레스이(가) 표시된 사진

자동 생성된 설명

R’G’B’ image HSV’2RGB

* R’G’B’ image는 원본보다 색채가 밝아졌다는 느낌이 들었습니다. 여자아이의 피부는 더욱 붉어지고, 그림자의 색깔에서 푸른 빛이 강해지는 등 변화가 확실하게 느껴졌습니다. 왜냐하면 R, G, B각각에 평활화를 했기 때문에 R, G, B에서 밝았던 부분은 더욱 밝아지고, 어두웠던 부분은 더욱 어두워졌습니다. 그래서 이를 다시 합쳤을 때 3개의 프레임에서 밝기 변화가 일어났기 때문에 색상 변화의 정도가 하나의 이미지를 평활화 했을 때보다 색감이 진해지지 않았을까 생각합니다. 그에 비해 HSV’2RGB는 명암만 평활화를 수행했음에도 원본 이미지보다 색감이 칙칙해지고 탁해지는 효과가 있었습니다. 왜냐하면 HSV 모델은 2개의 원추를 붙인 구조 형태로 표현하고 있는데, 명암의 값에 따라 색상과 채도를 표현할 수 있는 범위가 달라지게 됩니다. 그래서 명암 이미지에서 원래 큰 밝기 값을 가지고 있던 부분은 밝기 값이 커지면서 채도와 색상의 값이 줄어들게 되고, 반대로 작은 밝기 값을 가지고 있던 부분은 밝기 값이 더 작아지면서 채도와 색상 값이 줄어들게 되는 것입니다. 그렇기 때문에 명암 이미지를 평활화하여 값의 분포를 변화시킨 것만으로도 원본 이미지에 비해 탁하고 전체적으로 어두워지는 이미지를 결과로 얻어낸 것입니다.