Digital Image Processing

LabWork #CV4

Professor : Jin-Woo Jung

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

1.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
|  |  |  |  |
| code | | | |
| 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. rice.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("rice.png", IMREAD\_COLOR); // rice.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. Extract Sobel edge using GS\_sobel\_edge\_Sobel() or cv::Sobel() method

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
|  |  |  |  |
| code | | | |
| 1. jenny.jpg   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CEdge edge; //edge detect를 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 sobel = edge.GS\_sobel\_edge\_Sobel(convert, 2); //gray-scale로 convert된 이미지에 sobel operator를 수행한다. 이때 첫번째 파라미터는 연산을 수행할 이미지, 두 번째 파라미터는 연산을 수행할 방향이다.  //두 번째 파라미터의 숫자에 따라 연산의 수행 방향이 달라지는데 , 0 : 수직 엣지, 1 : 수평 엣지, 2 : 수직 / 수평 엣지 둘 다 수행햔다.  //수직, 수평 엣지 하나로만 연산을 수행할 경우 표현이 안되는 엣지들이 일부 존재했지만, 수직/ 수평 연산 둘 다 할 경우에는 하나만 한 것보다 엣지가 잘 표현된다.  imshow("image", sobel); // sobel edge를 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. rice.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CEdge edge; //edge detect를 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("rice.png", IMREAD\_COLOR); // rice.png를 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 sobel = edge.GS\_sobel\_edge\_Sobel(convert, 2); //gray-scale로 convert된 이미지에 sobel operator를 수행한다. 이때 첫번째 파라미터는 연산을 수행할 이미지, 두 번째 파라미터는 연산을 수행할 방향이다.  //두 번째 파라미터의 숫자에 따라 연산의 수행 방향이 달라지는데 , 0 : 수직 엣지, 1 : 수평 엣지, 2 : 수직 / 수평 엣지 둘 다 수행햔다.  //수직, 수평 엣지 하나로만 연산을 수행할 경우 표현이 안되는 엣지들이 일부 존재했지만, 수직/ 수평 연산 둘 다 할 경우에는 하나만 한 것보다 엣지가 잘 표현된다.  imshow("image", sobel); // sobel edge를 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. Extract Laplacian edge using GS\_laplacian\_edge\_Laplacian or cv::Laplacian() method

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
|  |  |  |  |
| code | | | |
| 1. jenny.jpg   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CEdge edge; //edge detect를 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg를 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 laplacian = edge.GS\_laplacian\_edge\_Laplacian(convert); //gray-scale로 convert된 이미지에 laplacian edge를 추출한다. 이때 파라미터는 연산을 수행할 이미지이다.  imshow("image", laplacian); // Laplacian edge를 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. rice.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CEdge edge; //edge detect를 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("rice.png", IMREAD\_COLOR); // rice.png를 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 laplacian = edge.GS\_laplacian\_edge\_Laplacian(convert); //gray-scale로 convert된 이미지에 laplacian edge를 추출한다. 이때 파라미터는 연산을 수행할 이미지이다.  imshow("image", laplacian); // Laplacian edge를 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. Extract Canny edge using GS\_canny\_edge\_Canny or cv::Canny() method

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
|  |  |  |  |
| code | | | |
| 1. jenny.jpg   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CEdge edge; //edge detect를 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("jenny.jpg", IMREAD\_COLOR); // jenny.jpg를 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 canny = edge.GS\_canny\_edge\_Canny(convert, 50, 100); //gray-scale로 convert된 이미지에 canny edge를 추출한다. 이때 첫번째 파라미터는 연산을 수행할 이미지, 두번째 파라미터는 low threshold, 세번째 파라미터는 high threshold이다.  //low threshold가 커질수록 탐색되는 edge의 수가 줄어든다.  //high threshold가 작아질수록 탐색되는 edge의 수가 늘어난다.  imshow("image", canny); // canny edge를 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. rice.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CEdge edge; //edge detect를 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("rice.png", IMREAD\_COLOR); // rice.png를 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 canny = edge.GS\_canny\_edge\_Canny(convert, 50, 100); //gray-scale로 convert된 이미지에 canny edge를 추출한다. 이때 첫번째 파라미터는 연산을 수행할 이미지, 두번째 파라미터는 low threshold, 세번째 파라미터는 high threshold이다.  //low threshold가 커질수록 탐색되는 edge의 수가 줄어든다.  //high threshold가 작아질수록 탐색되는 edge의 수가 늘어난다.  imshow("image", canny); // canny edge를 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. Extract Laplacian of Gaussian (sigma value = 2)

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | | **Output** | |
|  |  |  |  |
| code | | | |
| 1. jenny.jpg   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Edge.h"  #include "Filter.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CEdge edge; //edge detect를 위한 클래스를 생성한다.  CFilter filter; //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.jpg를 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 gaussian = filter.GS\_gaussian\_smoothing\_filtering(convert, 2); //gaussian 필터를 이미지에 적용한다.  Mat log = edge.GS\_laplacian\_edge\_Laplacian(gaussian); //gaussian smoothing 처리된 이미지에서 laplacian edge를 추출한다. 이때 첫번째 파라미터는 연산을 수행할 이미지이다.    log = processing.GS\_multiple\_constant(log, 30); //edge를 명확하게 볼 수 있도록 픽셀값들에 값을 곱한다.  imshow("image", log); // LoG를 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. rice.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Edge.h"  #include "Filter.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스를 생성한다.  CEdge edge; //edge detect를 위한 클래스를 생성한다.  CFilter filter; //filtering 작업을 위한 클래스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("rice.png", IMREAD\_COLOR); // rice.png를 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 gaussian = filter.GS\_gaussian\_smoothing\_filtering(convert, 2); //gaussian 필터를 이미지에 적용한다.  Mat log = edge.GS\_laplacian\_edge\_Laplacian(gaussian); //gaussian smoothing 처리된 이미지에서 laplacian edge를 추출한다. 이때 첫번째 파라미터는 연산을 수행할 이미지이다.    log = processing.GS\_multiple\_constant(log, 30); //edge를 명확하게 볼 수 있도록 픽셀값들에 값을 곱한다.  imshow("image", log); // LoG를 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. Discuss on the best edge detector. I.e., what is the best edge detector among Sobel, Laplacian, Canny and Laplacian of Gaussian? and why?

* Sobel edge에서는 edge들의 두께가 균일하지 않고, 중간에 끊어진 부분이 있습니다. Laplacian edge는 Sobel edge보다 edge들의 두께가 균일하고, 중간에 끊어진 부분이 적지만 그래도 일부분에서 끊어진 곳이 있었고, edge의 두께가 너무 얇아 관찰하기 힘든 부분도 있었습니다. 이를 개선하기 위해 고안된 Laplacian of Gaussian에서는 Laplacian edge보다 edge의 두께가 두꺼워져서 관찰하기 쉬워졌지만, 일부 edge가 뭉개진 것을 볼 수 있었습니다. Canny edge는 edge의 두께가 균일하고, 중간에 끊어진 부분도 일부 있었지만 그래도 거의 edge가 연결된 것을 볼 수 있었습니다. 또한 edge가 섬세히 표현된 것을 볼 수 있었고, edge의 밝기가 높은 덕분에 확인이 편했습니다. 그래서 가장 좋은 detector는 canny edge detector가 아닐까 저는 생각합니다.
* Using the given CGeometry class, design a program that can process the following : Use only briefcase.jpg as Input image for all problems

1.

1. (Use the center of given image as center for rotation and scaling)
   1. Rotate the image by 45 degrees using GS\_affineRotate (clockwise)

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |
| code | |
| #include <opencv2/opencv.hpp>  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat rotate = geometry.GS\_affineRotate(image, -45, 1.0); //45 degree로 이미지를 회전시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  imshow("image", rotate); // 회전한 image를 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. Enlarge the image 1.5 times : A image

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

| code |
| --- |
| #include <opencv2/opencv.hpp>  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat rotate = geometry.GS\_affineRotate(image, -45, 1.0); //45 degree로 이미지를 회전시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  Mat scale = geometry.GS\_affineRotate(rotate, 0, 1.5); //이미지를 1.5배 확대시킨다.  imshow("image", scale); // 회전하고 확대한 image를 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. (Use the center of given image as center for rotation and scaling)
   1. Enlarge the image 1.5 times

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |
| code | |
| #include <opencv2/opencv.hpp>  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat scale = geometry.GS\_affineRotate(image, 0, 1.5); //이미지를 1.5배 확대시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  imshow("image", scale); // 확대한 image를 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. Rotate the image by 45 degrees using GS\_affineRotate (The ) : B image

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |
| code | |
| #include <opencv2/opencv.hpp>  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat scale = geometry.GS\_affineRotate(image, 0, 1.5); //이미지를 1.5배 확대시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  Mat rotate = geometry.GS\_affineRotate(scale, -45, 1.0); //확대시킨 이미지를 시계방향으로 45 degree 회전한다.  imshow("image", rotate); // 확대하고 회전시킨 image를 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 B image from A image and check the result

(If two images are exactly same, the result would be black image)

|  |  |  |
| --- | --- | --- |
| **Input** |  | **Output** |
|  |  |  |
| code | | |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  CPixel pixel; //pixel processing을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat scale\_B = geometry.GS\_affineRotate(image, 0, 1.5); //이미지를 1.5배 확대시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  Mat rotate\_B = geometry.GS\_affineRotate(scale\_B, -45, 1.0); //확대시킨 이미지를 시계방향으로 45 degree 회전한다.    Mat rotate\_A = geometry.GS\_affineRotate(image, -45, 1.0); //확대시킨 이미지를 시계방향으로 45 degree 회전한다.  Mat scale\_A = geometry.GS\_affineRotate(rotate\_A, 0, 1.5); //이미지를 1.5배 확대시킨다.  Mat subtract = pixel.GS\_subtract\_image(scale\_A, rotate\_A); //A 이미지에서 B 이미지를 subtract한다.  imshow("image", subtract); // subtract한 image를 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. (Use the center of given image as center for rotation and scaling)
   1. Rotate the image by 30 degrees (clockwise)

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |
| code | |
| #include <opencv2/opencv.hpp>  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat rotate = geometry.GS\_affineRotate(image, -30, 1.0); //이미지를 시계방향으로 30 degree 회전시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  imshow("image", rotate); // 회전한 image를 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. Repeat rotating the result image again 11 more times : A image

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |
| code | |
| #include <opencv2/opencv.hpp>  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat rotate = geometry.GS\_affineRotate(image, -30, 1.0); //이미지를 시계방향으로 30 degree 회전시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  for (int i = 0; i < 11; i++)  rotate = geometry.GS\_affineRotate(rotate, -30, 1.0); //반복문을 이용하여 시계방향으로 30 degree를 11번 반복하여 회전시킨다.  imshow("image", rotate); // 회전한 image를 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 A image from the original image and check the result

(If two images are exactly same, the result would be black image)

|  |  |  |
| --- | --- | --- |
| **Input** | | **Output** |
|  |  |  |
| code | | |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  CPixel pixel; //pixel processing을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat rotate = geometry.GS\_affineRotate(image, -30, 1.0); //이미지를 시계방향으로 30 degree 회전시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  for (int i = 0; i < 11; i++)  rotate = geometry.GS\_affineRotate(rotate, -30, 1.0); //반복문을 이용하여 시계방향으로 30 degree를 11번 반복하여 회전시킨다.  Mat subtract = pixel.GS\_subtract\_image(image, rotate); // 오리지널 이미지에서 회전시킨 이미지를 subtract 한다.  imshow("image", subtract); // subtract한 image를 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. Repeat the first step (1) of Prob.2 more than five times and then repeat (2) again.

(Input : briefcase.jpg)

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |
| code | |
| #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Geometry.h"  using namespace cv;  int main()  {  CGeometry geometry; //geometry 변환을 위한 클래스를 선언한다.  CPixel pixel; //pixel processing을 위한 클래스를 선언한다.  Mat image = imread("briefcase.jpg", IMREAD\_COLOR); // briefcase.jpg를 불러온다. 이때 IMREAD\_COLOR 파라미터는 컬러 이미지를 불러온다는 것을 알려주기 위한 플래그 역할을 한다.  Mat rotate, subtract;  rotate = geometry.GS\_affineRotate(image, -30, 1.0); //이미지를 시계방향으로 30 degree 회전시킨다. 이때 첫번째 파라미터는 변환 시킬 이미지, 두번째 파라미터는 회전시킬 각도, 세번째 파라미터는 확대시킬 비율이다.  //두 번째 파라미터에서 양의 값을 넣으면 반시계방향, 음의 값을 넣으면 시계방향으로 회전한다.  //세 번째 파라미터에서 배율을 입력하면, 그 배율로 이미지가 확대된다.  for (int i = 0; i < 11; i++)  rotate = geometry.GS\_affineRotate(rotate, -30, 1.0); //반복문을 이용하여 시계방향으로 30 degree를 11번 반복하여 회전시킨다.  subtract = pixel.GS\_subtract\_image(image, rotate); // 오리지널 이미지에서 회전시킨 이미지를 subtract 한다.  for (int i = 0; i < 5; i++)  {  for (int j = 0;j < 12;j++)  subtract = geometry.GS\_affineRotate(subtract, -30, 1.0); //위에서 subtract된 이미지를 다시 회전시킨다.  subtract = pixel.GS\_subtract\_image(image, subtract); //회전시킨 이미지를 오리지널 이미지에서 subtract한다.  }  imshow("image", subtract); // subtract한 image를 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. Explain why the result of the step(3) B blurred.

처음에 이미지를 회전시킨 다음 오리지널 이미지에서subtract 한다면 중간 부분의 밝기값이 0에 가깝게 줄어들 것입니다. 그 상태에서 다시 회전시키고 오리지널 이미지에서subtract를 하면 할수록 중간 부분의 밝기값이 조금씩 줄어들면서 blur효과가 나타나는 것입니다.