Digital Image Processing

LabWork #CV6

Professor : Jin-Woo Jung

* Using the given CMorph, CEdge, CImageAnalysis classes
* Design a program that can perform the followings

|  |
| --- |
| 1. Binarization and Labeling : (1) + (2) + (3) + (4) + (5) 2. Finding Edge: [ (1) + (2) + (3) + (4) + (6) ] compare to [ (1) + (2) + (3) + (4) + (7) + (8) + (9) ] 3. Hough Transform: (1) + (2) + (3) + (4) + (10) 4. Compare the outputs of Finding Edge and Hough Transform |

<Steps for the implementation>

Use stuff\_color.jpg and rice.png after convert to grayscale image.

1. Erosion using erode function (kernel size 11x11 and MORPH\_ELLIPSE)
2. Dilation using dilate function step1 output
3. subtract the step2 output from the original image (gray)
4. Otzu binarization to make the binarized image for step3 output (threshold value = 0, THRESH\_BINARY | THRESH\_OTSU)
5. Labeling using GS\_findContours function for step4 output
6. Find edges of step4 output using GS\_canny\_edge\_Canny
7. Find edges of step4 output using GS\_sobel\_edge
8. Closing using GS\_closing function for step7 output (MORPH\_RECT)
9. Opening using GS\_opening function for step8 output (MORPH\_RECT)
10. Hough transform of step4 output using GS\_basicHoughTransformGray to find lines
11. Binarization and Labeling : (1) + (2) + (3) + (4) + (5)
    1. Erosion using erode function (kernel size 11x11 and MORPH\_ELLIPSE)

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

* 1. Dilation using dilate function step1 output

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

* 1. subtract the step2 output from the original image (gray)

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

* 1. Otzu binarization to make the binarized image for step3 output (threshold value = 0, THRESH\_BINARY | THRESH\_OTSU)

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

* 1. Labeling using GS\_findContours function for step4 output

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

| code |
| --- |
| 1. stuff\_color.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Morph.h"  #include "ImageAnalysis.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스의 인스턴스를 생성한다.  CMorph morphing; //morphology 연산을 위한 클래스의 인스턴스를 생성한다.  CImageAnalysis analysis; //이미지 분석을 위한 클래스의 인스턴스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("stuff\_color.png", IMREAD\_COLOR); // stuff\_color.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 eroision = morphing.GS\_erode(convert, MORPH\_ELLIPSE, 1); //이미지에 erode 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat dilation = morphing.GS\_dilate(eroision, MORPH\_ELLIPSE, 1); //이미지에 dilate 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.    //erode 한 다음 dilation을 했음으로 opening 연산에 해당한다.  Mat subtraction = processing.GS\_subtract\_image(convert, dilation); //gray-scale 이미지로 변환한 원본 이미지에서 opening 연산한 이미지를 subtract한다.  Mat binarization = processing.GS\_threshold(subtraction, 0, THRESH\_BINARY | THRESH\_OTSU); //subtract한 이미지를 이진화한다.  //이때, threshold값을 0으로 하고, threshold type flag에 THRESH\_BINARY, THRESH\_OTSU를 같이 지정하여 otsu method를 수행한다.  Mat label = analysis.GS\_findContours(binarization); //이진화 한 이미지를 이용하여 labeling을 수행한다.  imshow("image", label); // labeling 한 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"  #include "Morph.h"  #include "ImageAnalysis.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스의 인스턴스를 생성한다.  CMorph morphing; //morphology 연산을 위한 클래스의 인스턴스를 생성한다.  CImageAnalysis analysis; //이미지 분석을 위한 클래스의 인스턴스를 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.    Mat eroision = morphing.GS\_erode(convert, MORPH\_ELLIPSE, 1); //이미지에 erode 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat dilation = morphing.GS\_dilate(eroision, MORPH\_ELLIPSE, 1); //이미지에 dilate 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.    //erode 한 다음 dilation을 했음으로 opening 연산에 해당한다.  Mat subtraction = processing.GS\_subtract\_image(convert, dilation); //gray-scale 이미지로 변환한 원본 이미지에서 opening 연산한 이미지를 subtract한다.    Mat binarization = processing.GS\_threshold(subtraction, 0, THRESH\_BINARY | THRESH\_OTSU); //subtract한 이미지를 이진화한다. .  //이때, threshold값을 0으로 하고, threshold type flag에 THRESH\_BINARY, THRESH\_OTSU를 같이 지정하여 otsu method를 수행한다.  //원본 이미지보다 noise가 많이 제거된 것을 볼 수 있다.  Mat label = analysis.GS\_findContours(binarization); //이진화 한 이미지를 이용하여 labeling을 수행한다.  imshow("image", label); // labeling 한 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. Finding Edge: [ (1) + (2) + (3) + (4) + (6) ] compare to [ (1) + (2) + (3) + (4) + (7) + (8) + (9) ]
   1. Find edges of step4 output using GS\_canny\_edge\_Canny

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

* 1. Find edges of step4 output using GS\_sobel\_edge

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

* 1. Closing using GS\_closing function for step7 output (MORPH\_RECT)

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

* 1. Opening using GS\_opening function for step8 output (MORPH\_RECT)

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

| code |
| --- |
| 1. stuff\_color.png 2. canny edge   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Morph.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스의 인스턴스를 생성한다.  CMorph morphing; //morphology 연산을 위한 클래스의 인스턴스를 생성한다.  CEdge edge; //edge detecting을 위한 클래스의 인스턴스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("stuff\_color.png", IMREAD\_COLOR); // stuff\_color.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 eroision = morphing.GS\_erode(convert, MORPH\_ELLIPSE, 1); //이미지에 erode 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat dilation = morphing.GS\_dilate(eroision, MORPH\_ELLIPSE, 1); //이미지에 dilate 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  //erode 한 다음 dilation을 했음으로 opening 연산에 해당한다.  Mat subtraction = processing.GS\_subtract\_image(convert, dilation); //gray-scale 이미지로 변환한 원본 이미지에서 opening 연산한 이미지를 subtract한다.  Mat binarization = processing.GS\_threshold(subtraction, 0, THRESH\_BINARY | THRESH\_OTSU); //subtract한 이미지를 이진화한다. .  //이때, threshold값을 0으로 하고, threshold type flag에 THRESH\_BINARY, THRESH\_OTSU를 같이 지정하여 otsu method를 수행한다.  //원본 이미지보다 noise가 많이 제거된 것을 볼 수 있다.  Mat canny = edge.GS\_canny\_edge\_Canny(binarization); //이진화한 이미지에서 canny edge detecting을 수행한다.  //파라미터로는 연산을 수행할 이미지, low threshold, high threshold가 온다.  //이 함수는 low threshold가 default로 50, high threshold가 default로 100이 설정되어있다.  imshow("image", canny); // 탐색한 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. closing and opening of sobel edge   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Morph.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스의 인스턴스를 생성한다.  CMorph morphing; //morphology 연산을 위한 클래스의 인스턴스를 생성한다.  CEdge edge; //edge detecting을 위한 클래스의 인스턴스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("stuff\_color.png", IMREAD\_COLOR); // stuff\_color.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 eroision = morphing.GS\_erode(convert, MORPH\_ELLIPSE, 1); //이미지에 erode 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat dilation = morphing.GS\_dilate(eroision, MORPH\_ELLIPSE, 1); //이미지에 dilate 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  //erode 한 다음 dilation을 했음으로 opening 연산에 해당한다.  Mat subtraction = processing.GS\_subtract\_image(convert, dilation); //gray-scale 이미지로 변환한 원본 이미지에서 opening 연산한 이미지를 subtract한다.  Mat binarization = processing.GS\_threshold(subtraction, 0, THRESH\_BINARY | THRESH\_OTSU); //subtract한 이미지를 이진화한다. .  //이때, threshold값을 0으로 하고, threshold type flag에 THRESH\_BINARY, THRESH\_OTSU를 같이 지정하여 otsu method를 수행한다.  //원본 이미지보다 noise가 많이 제거된 것을 볼 수 있다.  Mat sobel = edge.GS\_sobel\_edge(binarization); //이진화한 이미지에서 sobel edge detecting을 수행한다.  //파라미터로는 연산을 수행할 이미지, detecting 방향을 정하는 값이 온다.  //만약 방향을 정하는 값이 0이면 수직 edge detecting, 1이면 수평 edge detecting, 2이면 수직/수평 edge detecting을 수행한다.  //방향을 정하는 값은 default로 2로 설정되어있다.  Mat closing = morphing.GS\_closing(sobel, MORPH\_RECT, 1); //이미지에 close 연산을 취한다. (dilation -> erode)  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat opening = morphing.GS\_opening(closing, MORPH\_RECT, 1); //이미지에 opening 연산을 취한다. (erode -> dilation)  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  imshow("image", opening); // 연산을 수행한 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 2. canny edge   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Morph.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스의 인스턴스를 생성한다.  CMorph morphing; //morphology 연산을 위한 클래스의 인스턴스를 생성한다.  CEdge edge; //edge detecting을 위한 클래스의 인스턴스를 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.  Mat eroision = morphing.GS\_erode(convert, MORPH\_ELLIPSE, 1); //이미지에 erode 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat dilation = morphing.GS\_dilate(eroision, MORPH\_ELLIPSE, 1); //이미지에 dilate 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  //erode 한 다음 dilation을 했음으로 opening 연산에 해당한다.  Mat subtraction = processing.GS\_subtract\_image(convert, dilation); //gray-scale 이미지로 변환한 원본 이미지에서 opening 연산한 이미지를 subtract한다.  Mat binarization = processing.GS\_threshold(subtraction, 0, THRESH\_BINARY | THRESH\_OTSU); //subtract한 이미지를 이진화한다. .  //이때, threshold값을 0으로 하고, threshold type flag에 THRESH\_BINARY, THRESH\_OTSU를 같이 지정하여 otsu method를 수행한다.  //원본 이미지보다 noise가 많이 제거된 것을 볼 수 있다.  Mat canny = edge.GS\_canny\_edge\_Canny(binarization); //이진화한 이미지에서 canny edge detecting을 수행한다.  //파라미터로는 연산을 수행할 이미지, low threshold, high threshold가 온다.  //이 함수는 low threshold가 default로 50, high threshold가 default로 100이 설정되어있다.  imshow("image", canny); // 탐색한 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. closing and opening of sobel edge   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Morph.h"  #include "Edge.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스의 인스턴스를 생성한다.  CMorph morphing; //morphology 연산을 위한 클래스의 인스턴스를 생성한다.  CEdge edge; //edge detecting을 위한 클래스의 인스턴스를 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.  Mat eroision = morphing.GS\_erode(convert, MORPH\_ELLIPSE, 1); //이미지에 erode 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat dilation = morphing.GS\_dilate(eroision, MORPH\_ELLIPSE, 1); //이미지에 dilate 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  //erode 한 다음 dilation을 했음으로 opening 연산에 해당한다.  Mat subtraction = processing.GS\_subtract\_image(convert, dilation); //gray-scale 이미지로 변환한 원본 이미지에서 opening 연산한 이미지를 subtract한다.  Mat binarization = processing.GS\_threshold(subtraction, 0, THRESH\_BINARY | THRESH\_OTSU); //subtract한 이미지를 이진화한다. .  //이때, threshold값을 0으로 하고, threshold type flag에 THRESH\_BINARY, THRESH\_OTSU를 같이 지정하여 otsu method를 수행한다.  //원본 이미지보다 noise가 많이 제거된 것을 볼 수 있다.  Mat sobel = edge.GS\_sobel\_edge(binarization); //이진화한 이미지에서 sobel edge detecting을 수행한다.  //파라미터로는 연산을 수행할 이미지, detecting 방향을 정하는 값이 온다.  //만약 방향을 정하는 값이 0이면 수직 edge detecting, 1이면 수평 edge detecting, 2이면 수직/수평 edge detecting을 수행한다.  //방향을 정하는 값은 default로 2로 설정되어있다.  Mat closing = morphing.GS\_closing(sobel, MORPH\_RECT, 1); //이미지에 close 연산을 취한다. (dilation -> erode)  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat opening = morphing.GS\_opening(closing, MORPH\_RECT, 1); //이미지에 opening 연산을 취한다. (erode -> dilation)  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  imshow("image", opening); // 연산을 수행한 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;  } |

* 비교내용 작성

Canny edge detecting한 결과와 Sobel edge detecting 한 후 closing, opening한 결과는 놀랍게도 큰 차이를 보여줬습니다. Canny edge detecting에서는 예상대로 두께가 균일하고, 끊어짐이 없는 edge를 출력했습니다. 그런데 sobel edge detecting한 상태에서 closing을 하면 edge의 두께가 두꺼워진 것을 제외하면 edge의 형태는 그대로였습니다. 그러나 여기서 opening을 할 경우 edge의 중간중간에 끊어짐이 발생하면서 원래의 edge모양과는 동떨어진 결과물을 출력하게 되었습니다. 이러한 현상이 발생하는 이유는 structure element의 크기가 edge의 두께보다 크기 때문에 opening 작업 중 erode 연산에 의해 edge의 중간 부분이 많이 지워지지 않았을까 생각합니다.

1. Hough Transform: (1) + (2) + (3) + (4) + (10)
   1. Hough transform of step4 output using GS\_basicHoughTransformGray to find lines

|  |  |
| --- | --- |
| **Input** | **Output** |
|  |  |
|  |  |
| code | |
| 1. stuff\_color.png   #include <opencv2/opencv.hpp>  #include "Pixel.h"  #include "Morph.h"  #include "ImageAnalysis.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스의 인스턴스를 생성한다.  CMorph morphing; //morphology 연산을 위한 클래스의 인스턴스를 생성한다.  CImageAnalysis analysis; //이미지 분석을 위한 클래스의 인스턴스를 생성한다.  Mat convert; //input image를 8bit gray-scale로 변환하고 저장하기 위한 변수를 선언한다.  Mat image = imread("stuff\_color.png", IMREAD\_COLOR); // stuff\_color.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 eroision = morphing.GS\_erode(convert, MORPH\_ELLIPSE, 1); //이미지에 erode 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat dilation = morphing.GS\_dilate(eroision, MORPH\_ELLIPSE, 1); //이미지에 dilate 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  //erode 한 다음 dilation을 했음으로 opening 연산에 해당한다.  Mat subtraction = processing.GS\_subtract\_image(convert, dilation); //gray-scale 이미지로 변환한 원본 이미지에서 opening 연산한 이미지를 subtract한다.  Mat binarization = processing.GS\_threshold(subtraction, 0, THRESH\_BINARY | THRESH\_OTSU); //subtract한 이미지를 이진화한다. .  //이때, threshold값을 0으로 하고, threshold type flag에 THRESH\_BINARY, THRESH\_OTSU를 같이 지정하여 otsu method를 수행한다.  //원본 이미지보다 noise가 많이 제거된 것을 볼 수 있다.  Mat hough = analysis.GS\_basicHoughTransformGray(binarization);  imshow("image", hough); // 연산을 수행한 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"  #include "Morph.h"  #include "ImageAnalysis.h"  using namespace cv;  int main()  {  CPixel processing; //pixel processing을 위한 클래스의 인스턴스를 생성한다.  CMorph morphing; //morphology 연산을 위한 클래스의 인스턴스를 생성한다.  CImageAnalysis analysis; //이미지 분석을 위한 클래스의 인스턴스를 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.  Mat eroision = morphing.GS\_erode(convert, MORPH\_ELLIPSE, 1); //이미지에 erode 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태, 이진화를 수행한 후에 연산을 취할지 아니면 그대로 연산을 취할 것인지 정하는 flag로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  Mat dilation = morphing.GS\_dilate(eroision, MORPH\_ELLIPSE, 1); //이미지에 dilate 연산을 취한다.  //파라미터는 연산을 수행할 이미지, structure element의 형태로 구성되어 있다.  //structure element의 형태에는 MORPH\_CROSS(십자), MORPH\_RECT(사각형), MORPH\_ELLIPSE(타원)이 있다.  //structure element의 형태에 따라 동일한 이미지에 대한 morphology 연산의 결과는 다르게 나온다.  //erode 한 다음 dilation을 했음으로 opening 연산에 해당한다.  Mat subtraction = processing.GS\_subtract\_image(convert, dilation); //gray-scale 이미지로 변환한 원본 이미지에서 opening 연산한 이미지를 subtract한다.  Mat binarization = processing.GS\_threshold(subtraction, 0, THRESH\_BINARY | THRESH\_OTSU); //subtract한 이미지를 이진화한다. .  //이때, threshold값을 0으로 하고, threshold type flag에 THRESH\_BINARY, THRESH\_OTSU를 같이 지정하여 otsu method를 수행한다.  //원본 이미지보다 noise가 많이 제거된 것을 볼 수 있다.  Mat hough = analysis.GS\_basicHoughTransformGray(binarization);  imshow("image", hough); // 연산을 수행한 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. Compare the outputs of Finding Edge and Hough Transform

stuff\_color.png에서 hough transform 결과를 보면, 배경 때문에 잘 보이지 않지만 곳곳에 파랗게 표현된 것을 볼 수 있습니다. 그런데 edge 부분 뿐만 아니라 일부 픽셀이 모여 있는 부분에서도 hough transform이 표현되어 있음을 발견할 수 있습니다. 이를 통해 line으로 표현될 수 없는 부분에서도 hough transform을 통해 line으로 오인하는 경우가 있음을 볼 수 있었습니다. rice.png 또한 hough transform 결과를 살펴보면 stuff\_color.png보다 선명하게 line을 발견할 수 있었습니다. 그러나 stuff\_color.png와 마찬가지로 line으로 판별될 수 없는 부분이 hough transform을 통해 line으로 판별되어 출력된 것을 볼 수 있었습니다. 이런 현상이 발생하는 이유가 이미지 line이 표현되지 않아도 픽셀들이 line을 이루는 것처럼 분포하게 되면서 hough transform에서 line일 가능성이 높다고 판단하고 line임을 출력하기 때문이라고 생각합니다.