**\*adaptive\_threshold.cpp\***

**1. Code Explanation**

1) overall explanation

이 코드는 moving averages를 이용한 adaptive thresholding 을 구현한다. input으로 받은 image를 thresholding을 거쳐 0 혹은 255의 밝기를 가진 이미지를 출력한다.

(다음 페이지에)

2) code

Mat adaptive\_thres(const Mat input, int n, float bnumber) {

Mat kernel; int row = input.rows; int col = input.cols;

int kernel\_size = (2 \* n + 1);

// Initialiazing Kernel Matrix (uniform mean filter kernel)

kernel = Mat::ones(kernel\_size, kernel\_size, CV\_32F) / (float)(kernel\_size \* kernel\_size);

float kernelvalue = kernel.at<float>(0, 0); // To simplify, as the filter is uniform. All elements of the kernel value are same.

Mat output = Mat::zeros(row, col, input.type());

for (int i = 0; i < row; i++) { //for each pixel in the output

for (int j = 0; j < col; j++) {

// finds the mean intensity using uniform mean filtering with zero paddle border process.

float sum1 = 0.0;

for (int a = -n; a <= n; a++) { // for each kernel window

for (int b = -n; b <= n; b++) {

if ((i + a <= row - 1) && (i + a >= 0) && (j + b <= col - 1) && (j + b >= 0)) { //if the pixel is not a border pixel

sum1 += kernelvalue \* (float)(input.at<G>(i + a, j + b));

}

}

}

float temp = bnumber \* (G)sum1; //threshold for each pixel: bnumber\*m(i,j)

//Fill code that makes output image's pixel intensity to 255 if the intensity of the input image is bigger

// than the temp value else 0.

if ((float)(input.at<G>(i, j) > temp))

output.at<G>(i, j) = 255;

else

output.at<G>(i, j) = 0;

}

}

return output;

}

3) analysis & explanation of the code

O(i, j) = 1 ( if I(i, j) > T(I, j) )

= 0 (otherwise)

T(I, j) = bnumber \* m(i, j) (m은 filtering을 이용해 구한 mean intensity)

Adaptive thresholding using moving averages를 이용한 output intensity는 위와 같다.

1. Uniform mean filter kernel을 초기화한 뒤, 동일한 원소들의 값을 kernelvalue에 저장한다.
2. Uniform mean filtering을 이용해 mean intensity를 구한 뒤 이를 sum1에 저장한다. (boundary pixel들을 filtering 할 때는 zero-padding 을 이용한다. )
3. bnumber에 mean intensity 값을 곱한 뒤 temp에 저장한다. 이것이 (i, j)에 대한 threshold가 된다.
4. (i, j)의 intensity가 thresold보다 크면 output intensity로 255, 그렇지 않으면 0을 저장한다.

**2. Results & analysis and explanation**

텍스트, 서류, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명

5x5, bnumber=0.9

**\*kmeans.cpp\***

**1. Code Explanation**

1) overall explanation

opencv에서 제공하는 kmeans함수를 이용하여 k-means clustering의 결과 이미지를 구하는 코드이다.

2) code

for (int y = 0; y < input.rows; y++)

for (int x = 0; x < input.cols; x++)

{

//Fill code that finds for each pixel of each channel of the output image the intensity of the cluster center.

int cluster\_idx = labels.at<int>(y + x \* input.rows, 0);

new\_image.at<C>(y, x)[0] = centers.at<float>(cluster\_idx, 0);

new\_image.at<C>(y, x)[1] = centers.at<float>(cluster\_idx, 1);

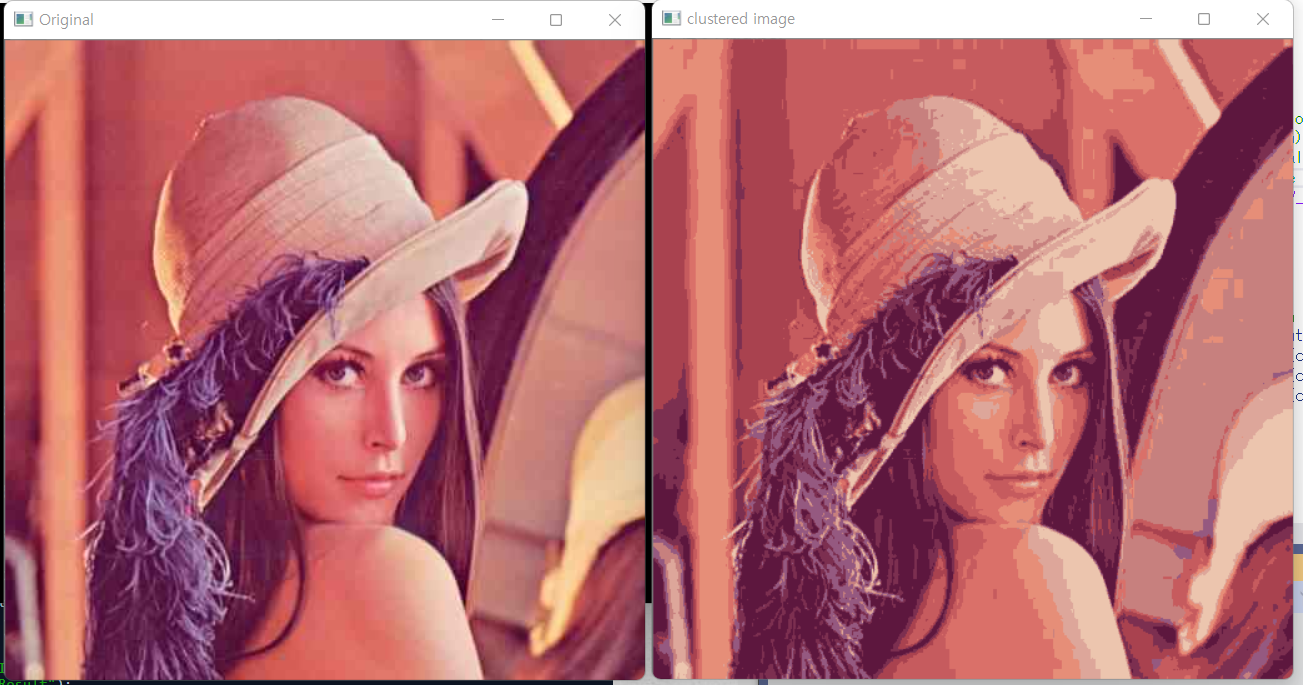
new\_image.at<C>(y, x)[2] = centers.at<float>(cluster\_idx, 2);

}

3) analysis & explanation of the code

1. kmeans() 함수를 실행시키면 labels에는 각 pixel이 속해 있는 cluster의 index가, centers에는 각 cluster의 intensity가 저장된다.
2. input의 모든 픽셀들에 대해 for 루프를 돌면서 해당 픽셀이 속해 있는 cluster의 index를 구해 cluster\_idx 변수에 저장한다.
3. 해당 cluster의 intensity를 각각 3개의 채널에 대해 구한 뒤, new\_image의 각 채널에 이를 저장한다.

**2. Results & analysis and explanation**



Color image segmentation

**\*mean\_shift\_convert2gray.cpp\***

**1. Code Explanation**

1) overall explanation

: converting “lena.jpg” into grayscale image and save it using imwrite() function

2) code

#include <iostream>

#include <opencv2/opencv.hpp>

using namespace cv;

int main() {

Mat input = imread("C:\\Users\\MIN\\Desktop\\권민서\\학교\\2022-1\\오소프\\Lab06\\lena.jpg", CV\_LOAD\_IMAGE\_COLOR);

Mat input\_gray;

cvtColor(input, input\_gray, CV\_RGB2GRAY); // Converting image to gray

if (!input.data)

{

std::cout << "Could not open" << std::endl;

return -1;

}

imwrite("C:\\Users\\MIN\\Desktop\\권민서\\학교\\2022-1\\오소프\\Lab06\\lena\_gray.jpg", input\_gray);

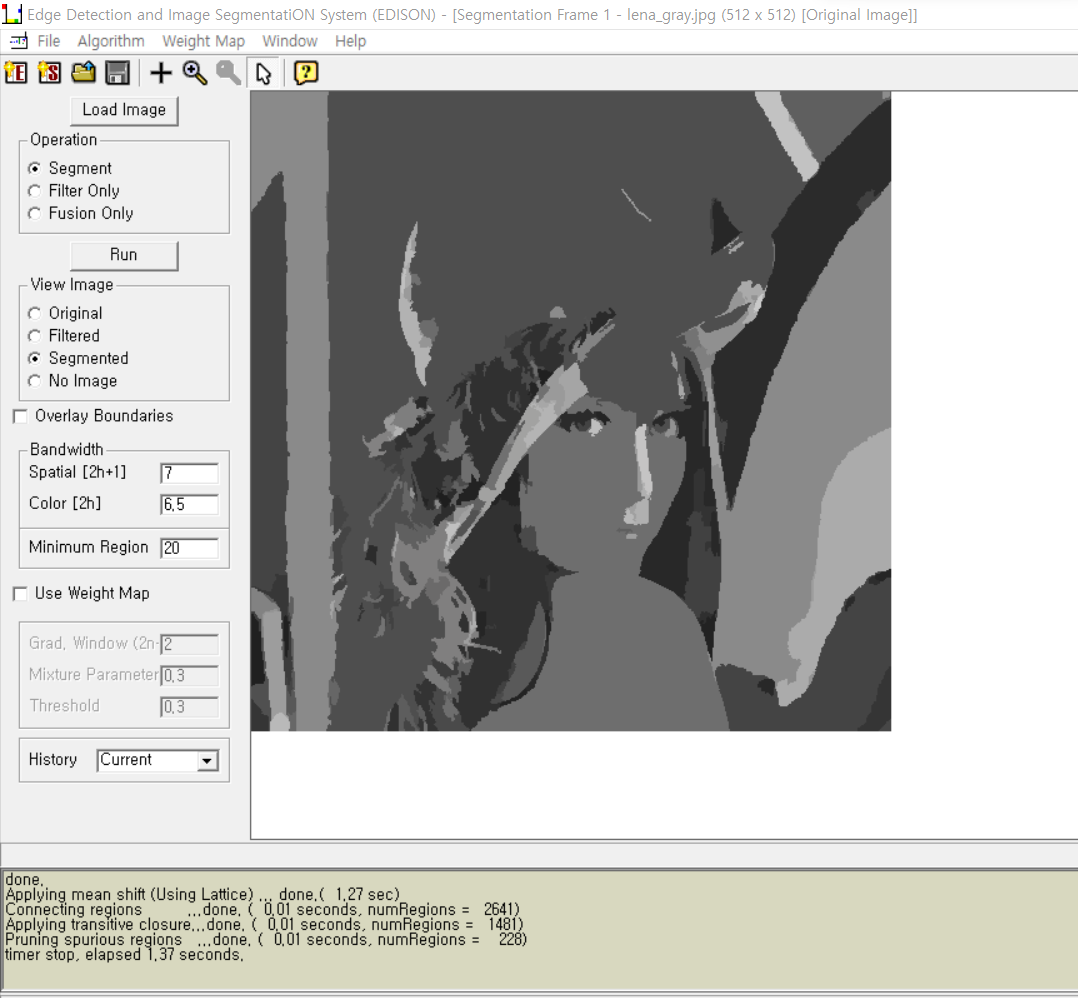
return 0;

}

3) analysis & explanation of the code

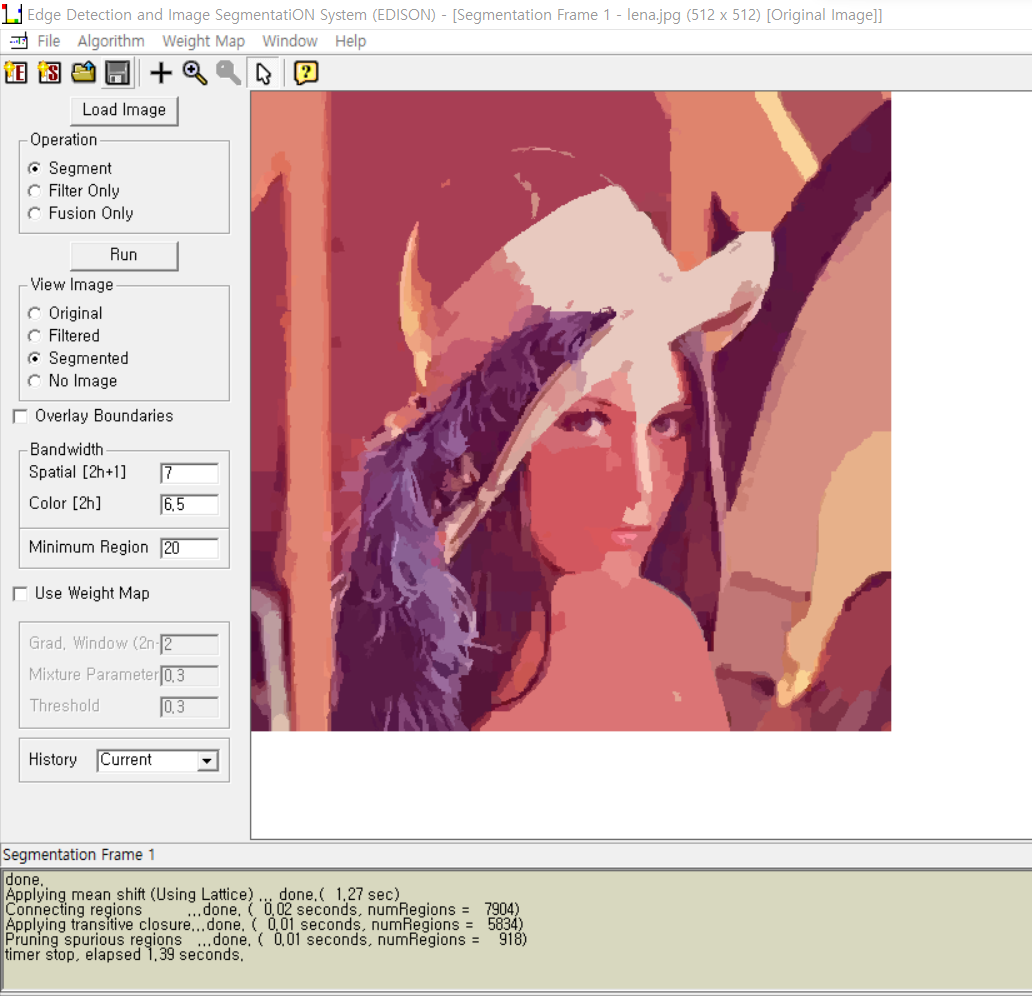
This code converts “lena.jpg” into grayscale image using cvtColor() function, and save the image in C:\\Users\\MIN\\Desktop\\권민서\\학교\\2022-1\\오소프[\\Lab06](file:///\\Lab06) . This is performed to load grayscale image as input in EDISON to see the result of mean shift segmentation.

**2. Results using EDISON**



Grayscale image

(다음 페이지에 계속)



Color image