HW1&2

*Date: 2021. 09. 16*

*Student ID: 21600635*

*Name: Bomoon JUNG*

**1. Homework 1**

1) Please follow the instructions below.

* Read a video “background.mp4”
* Display video for the first 3 seconds
* Find out the proper input parameter for waitKey()
* Print out the number of the current frame and the total number of frames

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트, 바닥이(가) 표시된 사진

자동 생성된 설명

**Figure 1. results of HW 1**

2) Explanation

**Mat and VideoCapture is provided by openCV library. Firstly, read image from mp4 file and check the exception case. Secondly, using while loop can stop at three seconds of play time. Thirdly, store a image from VideoCapture object to Mat object. If Mat object is empty, then stop play the video and break the loop. Fourthly, using get() function can check now frames and total frames. Lastly, update the current play time through get() function.**

3) Source code

*#include* <opencv2/opencv.hpp>

*#include* <iostream>

using *namespace* cv ;

using *namespace* std ;

*int*

main () {

Mat frame ;

VideoCapture cap ;

*// open mp4 file*

*if* ( cap.open("background.mp4") == 0 ) {

cout << "no such file" << endl ;

waitKey(0) ;

}

*int* play\_time = 0 ;

*// play the video for 3sec*

*while* ( play\_time != 3000 ) {

*// exception. (end of video)*

cap >> frame ; *// pixel data -> Mat (Mat can have pixel data)*

*if* ( frame.empty() ) {

cout << "end of video" << endl ;

*break* ;

}

*// show image*

imshow("video", frame) ;

waitKey(33) ;

*// print current frame / total frame*

*int* current\_frame = cap.get(CAP\_PROP\_POS\_FRAMES) ;

*int* total\_frames = cap.get(CAP\_PROP\_FRAME\_COUNT) ;

cout << current\_frame << " / "<< total\_frames << endl ;

*// update play time*

play\_time = cap.get(CAP\_PROP\_POS\_MSEC) ;

}

*return* 0 ;

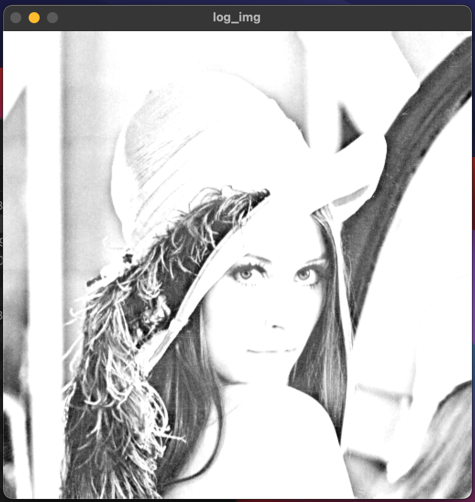
}

**2. Homework 2**

1) Please follow the instructions below.

* Read an image “Lena.png”
* Perform the following operations
* Negative transformation (Result mat: negative\_img)
* Log transformation (Result mat: log\_img)
* Use log(mat a) function to perform log operation
* To use log function, pixel type of input should be floating point
* Also use normalize(img, img, 0, 255, NORM\_MINMAX)
* Normalize img to (0 ~ 255)
* Gamma transformation with gamma as 0.5 (Result mat gamma\_img)
* Make sure you normalize pixel values from 0 to 1.0

텍스트, 사람, 실내이(가) 표시된 사진

자동 생성된 설명텍스트, 사람이(가) 표시된 사진

자동 생성된 설명

**Figure 2. results of HW 2**

2) Explanation

* Negative transformation

Equation of negative transformation is this one.

Therefore, pixel value is changed through this equation. For quick operation, I used pointer access to pixel values.

* Log transformation

Equation of log transformation is this one. The reason ‘+1’ is used is for log condition.

After log transformation, to protect pixel values that out of 0~255, normalization is needed. And then convert float type to uchar type.

* Gamma transformation

Equation of log transformation is this one.

I used lookup table to reduce the amount of computation. Therefore, original value of pixel is matched to lookup table values.

3) Source code

*#include* <opencv2/opencv.hpp>

*#include* <iostream>

*#include* <math.h>

*#include* <typeinfo>

using *namespace* cv ;

using *namespace* std ;

*int*

main () {

*// Loading image.*

Mat image = imread("Lena.png", 0) ;

*// Channels of image = 3. Because image is color.*

*int* channels = image.channels() ;

*//Exception.*

*if* (image.empty()) {

cout << "no such file" << endl ;

*return* 0 ;

}

*// 1. Negative transformation.*

Mat negative\_img = image.clone() ;

*// Position of pixcel.*

*int* pixel\_size = image.rows \* image.cols \* channels ;

*for* ( *int* i = 0 ; i < pixel\_size ; i++ )

negative\_img.data[ i ] = 255 - 1 - negative\_img.data[ i ] ;

*// Result of negative transformation.*

imshow("negative\_img", negative\_img) ;

*// 2. Log transformation.*

Mat log\_img ;

*// Convert uchar to float type.*

image.convertTo(log\_img, CV\_32F) ;

log(log\_img + 1, log\_img) ;

normalize(log\_img, log\_img, 0, 255, NORM\_MINMAX) ;

convertScaleAbs(log\_img, log\_img, 1.5) ; *// float -> uchar*

*// Result of log transformation.*

imshow("log\_img", log\_img) ;

*// 3. Gamma transformation.*

Mat gamma\_img = image.clone() ;

*unsigned* *char* look\_up[256] ;

*int* arr\_size = sizeof(look\_up) / sizeof(*unsigned* *char*) ;

*float* gamma = 0.5 ;

*// Look\_up table.*

*for* ( *int* i = 0 ; i < arr\_size ; i++ ) {

look\_up[i] = saturate\_cast<uchar>( pow( i / 255.0, gamma ) \* 255.0 ) ;

}

*for* ( *int* i = 0 ; i < pixel\_size ; i++ ) {

gamma\_img.data[ i ] = look\_up[ gamma\_img.data[ i ] ] ;

}

*// Result of gamma transformation.*

imshow("gamma\_img", gamma\_img) ;

waitKey(0) ;

*return* 0 ;

}