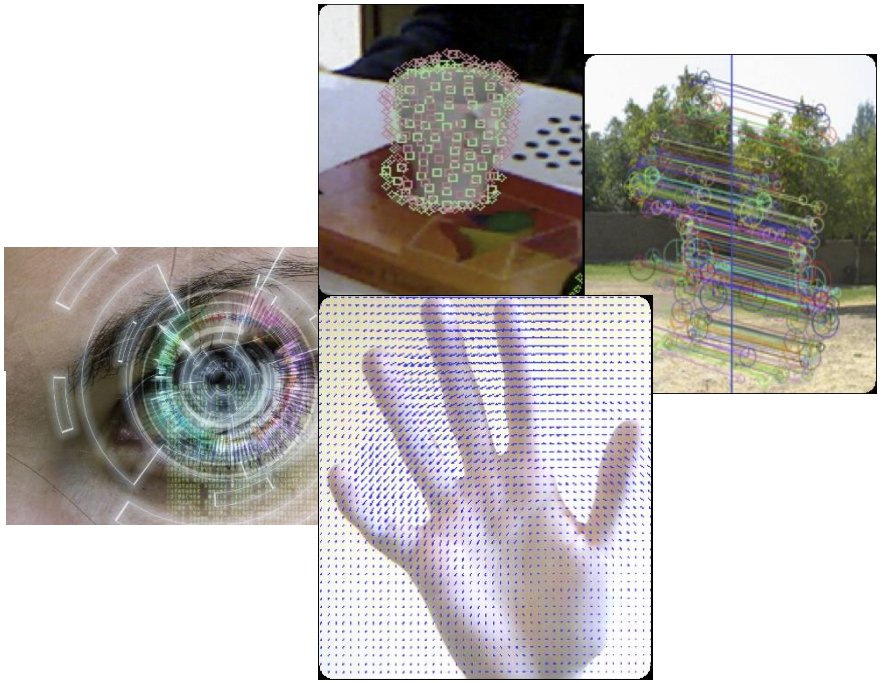
IVPL
Intelligent Vision Processing Lab

2023 Autumn

COMPUTER VISION 비젼 프로그래밍



chap.2 화소 점 처리 (pixel-point processing)-실습

>>화소 점 처리(pixel point processing): 개념

화소 점 처리

- 원 화소의 값이나 위치를 바탕으로 단일 화소 값을 변경하는 기술
- 다른 화소의 영향을 받지 않고 단순히 화소 점의 값만 변경하므로 포인트 처리(Point Processing)라고도 함.
- 산술연산, 논리연산, 반전, 광도 보정, 히스토그램 평활화, 명암 대비 스트레칭 등의 기법이 있음.
- 디지털 영상의 산술연산은 디지털 영상의 각 화소 값에서 임의의 상수 값으로 덧셈, 뺄셈, 곱셈, 나눗셈을 수행하는 것
- 그레이 레벨 영상에서 화소 값이 작으면 영상이 어둡고, 화소의 값이 크면 밝음.

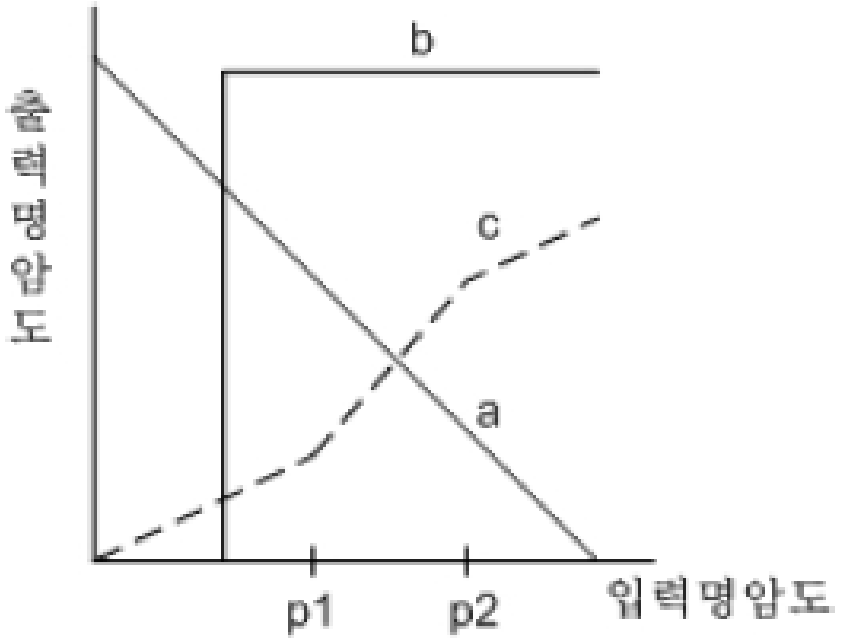
>> 화소 점 처리(pixel point processing): 산술연산/논리연산

- 산술연산
  - 덧셈
  - 뺄셈
  - 나눗셈
  - 곱셈

- 논리연산 (bit operation)
  - OR
  - AND
  - NAND
  - XOR

>>화소 점 처리(pixel point processing): 일반적인 화소 점 처리 설계

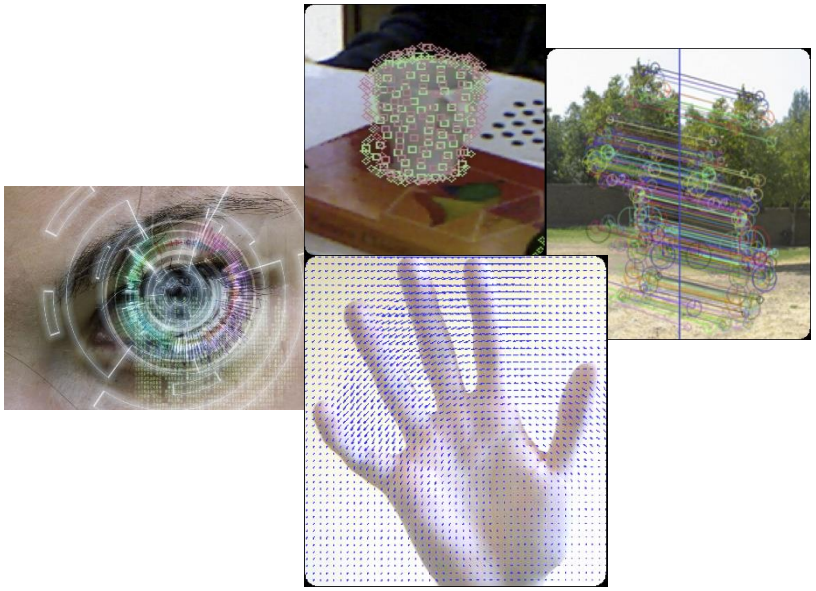
- 화소 점 처리를 일반화된 수식으로 표현
  - Q = T(p), Q=출력 화소값, p=입력 화소값
  - 선형(linear) 혹은 비선형(non-linear) 변환



[다양한 변환 관계식]

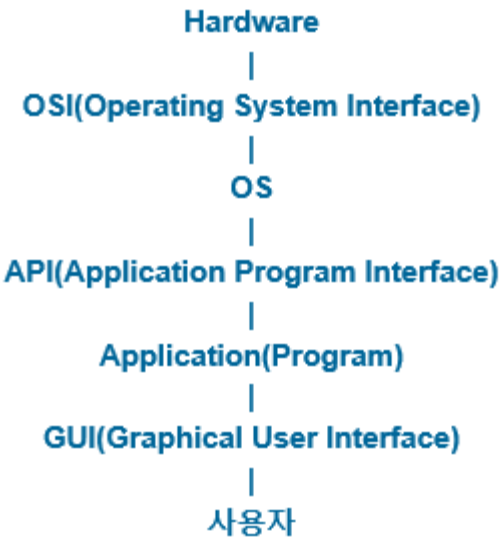
COMPUTER VISION 비젼 프로그래밍

what is API?



Application Programming Interface (API)란?

- 컴퓨터←→운영체제←???→사용자



- API는 프로그램(사용자)과 운영체제 사이의 중간매체로서 프로그램을 원할히 작동하게 끔 해주는 함수, 루틴과 프로토콜로 정의됨
- SW 구조상 분류로는 **middleware 영역**으로 정의하여 분류함
- 예) fopen(), printf() 함수 등

Application Programming Interface (API)란?

```
void CImageProcessingDoc::OnHistoEqual()
{
   int i, value;
   unsigned char LOW, HIGH, Temp;
   double SUM = 0.0;

   m_Re_height = m_height;
   m_Re_width = m_Width;
   m_Re_size = m_Re_height * m_Re_width;

   LOW = 0;
   HIGH = 255;

   // 초기화
   for(i=0 ; i<256 ; i++)
      m_HIST[i] = LOW;

   // 빈도 수 조사: 히스토그램 생성 //
   for(i=0 ; i<m_size ; i++){
      value = (int)m_InputImage[i];
      m_HIST[value]++;
   }

   // 누적 히스토그램 생성
   for(i=0 ; i<256 ; i++){
      SUM += m_HIST[i];
      m_Sum_Of_HIST[i] = SUM;
   }

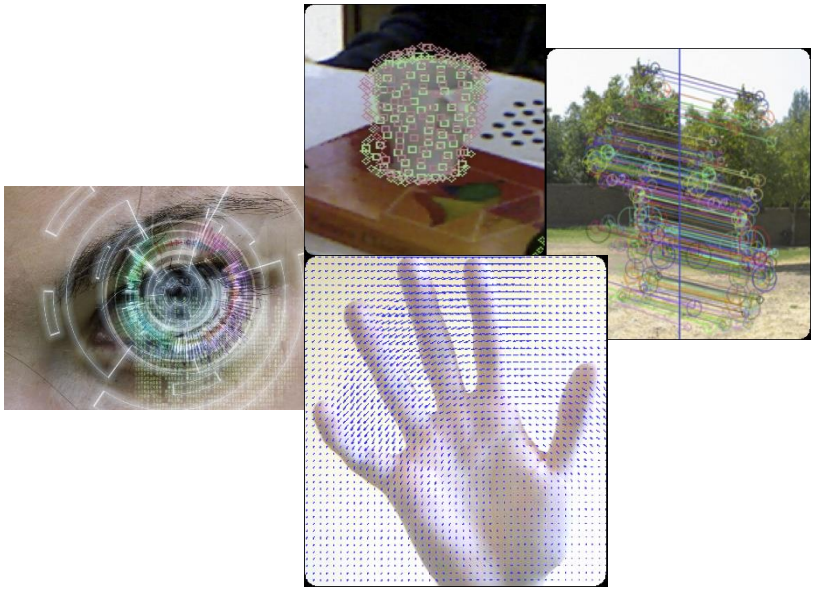
   m_OutputImage = new unsigned char[m_Re_size];

   // 입력 영상을 평활화된 영상으로 출력
   for(i=0 ; i<m_size ; i++){
      Temp = m_InputImage[i];
      m_OutputImage[i]=(unsigned char)(m_Sum_Of_ HIST[Temp]*
                              HIGH/m_size);
   }
}
```

cv::equalizeHist (InputArray src, OutputArray dst)

COMPUTER VISION 비젼 프로그래밍

컴퓨터비젼 프로그래밍 실습 구조



컴퓨터비젼 프로그래밍 실습 구조: SW 구조(1)

- header 추가 + main() 함수 또는
- header 추가+global variables + main() 함수

```
#include <~~~~>
#include "xxxxx.h"

Global 변수 선언

int main(int argc, char** argv)
{
      지역 변수 선언

      처리 알고리즘 구현

      ~~~

      return 0;
}
```

전처리기

전역변수 선언

main() 함수
: 영상 로딩/처리/결과 보여주기 등

컴퓨터비젼 프로그래밍 실습 구조: SW 구조(2)

- header 추가 + **사용자정의 함수**+ main() 함수 또는
- header 추가+(global variables) + **사용자정의 함수** + main() 함수

```
#include <~~~~>
#include "xxxxx.h"

Global 변수 선언

int sum(int a, int b){
            ~~
}

int main(int argc, char** argv)
{
      지역 변수 선언

      처리 알고리즘 구현

      ~~~

      return 0;
}
```

전처리기

전역변수 선언
및 사용자 함수

main() 함수
: 영상 로딩/처리/결과
보여주기 등

컴퓨터비젼 프로그래밍 실습 구조: SW 구조(3)

- header 추가 + **사용자 함수 선언**+ main() 함수+**사용자함수 구현**

```
#include <~~~~>
#include "xxxxx.h"

Global 변수 선언
int sum(int a, int b);

int main(int argc, char** argv)
{
      지역 변수 선언

      처리 알고리즘 구현

      ~~~

      return 0;
}

int sum(int a, int b){
            ~~
}
```

전처리기

전역변수
및 사용자 함수 선언

main() 함수
: 영상 로딩/처리/결과
보여주기 등

사용자 함수 구현

컴퓨터비젼 프로그래밍 실습 구조: SW 구조(4)

- header 추가 + (global variables) + **사용자 클래스 구현** + main() 함수

```
#include <~~~~>
#include "xxxxx.h"

Global 변수 선언

class CheckBox : public Control {
        public: Boolean IsChecked();
        virtual int ChangeState() = 0;
};

int main(int argc, char** argv)
{
    지역 변수 선언

    처리 알고리즘 구현

    ~~~

    return 0;
}
```

전처리기

전역변수
및 클래스 구현

main() 함수
: 영상 로딩/처리/결과
보여주기 등

컴퓨터비전 실제 실습 구조 예제-1

- SW 실습 구조 **(Type-1): main 함수 내에 기능 구현**

```
#include <iostream>
#include <vector>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d/features2d.hpp>

int main()
{
        // Read input images
        cv::Mat image1= cv::imread("../church01.jpg",0);
        cv::Mat image2= cv::imread("../church02.jpg",0);
        if (!image1.data || !image2.data)
        return 0;

                ~~~
                ~~~

          return 0;
}
```

컴퓨터비전 실제 실습 구조 예제-2

- SW 실습 구조 **(Type-2): 사용자 함수 구현 후 사용**

```
#include <~~~~>

///--- Global variables----///
int threshold_value = 0;
int threshold_type = 3;
int const max_value = 255;
int const max_type = 4;
int const max_BINARY_value = 255;

Mat image, src_gray, dst;
char* window_name = "Threshold Demo";

char* trackbar_type = "Type: ₩n 0: Binary ₩n 1: Binary Inverted ₩n 2: Truncate ₩n 3: To Zero ₩n 4: To Zero Inverted";
char* trackbar_value = "Value";

void Threshold_Demo( int, void* )          // 사용자 함수 구현
{
  /* 0: Binary      1: Binary Inverted      2: Threshold Truncated      3: Threshold to Zero  4: Threshold to Zero Inverted  */

  threshold( src_gray, dst, threshold_value, max_BINARY_value,threshold_type );

  imshow( window_name, dst );
}

int main( int argc, char** argv )
{
          /// Load image
           image = imread( "Desert.bmp", 1);   // Read the file

               (계속)
```

컴퓨터비전 실제 실습 구조 예제-2

```
    ~   ~   ~
    if (image.empty()){                              // Check for invalid input
                cout << "Could not open or find the image" << std::endl;
                return -1;
       }

     namedWindow("Display window", WINDOW_AUTOSIZE);  // Create a window for display.
     imshow("Display window", image );

    /// Call the function to initialize
      Threshold_Demo( 0, 0 );

      /// Wait until user finishes program
      while(true)
      {
            int c;
            c = waitKey( 20 );
            if( (char)c == 27 )
                        { break; }

    }
}
```

컴퓨터비전 실제 실습 구조 예제-3

- SW 실습 구조 **(Type-3): 사용자 함수 정의와 구현을 별도로 구현/사용**

```
#include <~~~~>

///--- Global variables----///
int threshold_value = 0;
int threshold_type = 3;
int const max_value = 255;
int const max_type = 4;
int const max_BINARY_value = 255;

Mat image, src_gray, dst;
char* window_name = "Threshold Demo";

char* trackbar_type = "Type: ₩n 0: Binary ₩n 1: Binary Inverted ₩n 2: Truncate ₩n 3: To Zero ₩n 4: To Zero Inverted";
char* trackbar_value = "Value";

void Threshold_Demo( int, void* );          //사용자 함수 선언

int main( int argc, char** argv )
{
            /// Load image
             image = imread( "Desert.bmp", 1);    // Read the file

              if (image.empty()){                                 // Check for invalid input
                        cout << "Could not open or find the image" << std::endl;
                        return -1;
              }
              namedWindow("Display window", WINDOW_AUTOSIZE);  // Create a window for display.
              imshow("Display window", image );

    (계속)
```

컴퓨터비전 실제 실습 구조 예제-3

```
            ~   ~   ~

        /// Call the function to initialize
          Threshold_Demo( 0, 0 );

          /// Wait until user finishes program
          while(true)
          {
                int c;
                c = waitKey( 20 );
                if( (char)c == 27 )
                                { break; }

        }
}
void Threshold_Demo( int, void* )          // 사용자 함수 구현
{
  /* 0: Binary       1: Binary Inverted       2: Threshold Truncated       3: Threshold to Zero   4: Threshold to Zero Inverted   */

  threshold( src_gray, dst, threshold_value, max_BINARY_value,threshold_type );

  imshow( window_name, dst );
}
```

컴퓨터비전 실제 실습 구조 예제-4

- SW 실습 구조 **(Type-4): Class 기반 구현**

```
#include <~~~~>

///--- Global variables----///
int threshold_value = 0;
int threshold_type = 3;
int const max_value = 255;
int const max_type = 4;
int const max_BINARY_value = 255;

class UserFunction{

    Mat image;
    void Threshold_Demo( int, void* )       // 사용자 함수 구현
    {
        /* 0: Binary       1: Binary Inverted       2: Threshold Truncated       3: Threshold to Zero  4: Threshold to Zero Inverted  */
        threshold( src_gray, dst, threshold_value, max_BINARY_value,threshold_type );
        imshow( window_name, dst );
   }
    .....

};

int main( int argc, char** argv )
{
          UserFunction* fun = UserFunction;

          /// Load image
          fun.image = imread( "Desert.bmp", 1);   // Read the file


    (계속)
```

컴퓨터비전 실제 실습 구조 예제-4

```
        ~   ~   ~
    if (fun.image.empty()){                         // Check for invalid input
                cout << "Could not open or find the image" << std::endl;
                return -1;
        }
        namedWindow("Display window", WINDOW_AUTOSIZE);  // Create a window for display.
        imshow("Display window", fun.image );

    /// Call the function to initialize
      fun.Threshold_Demo( 0, 0 );

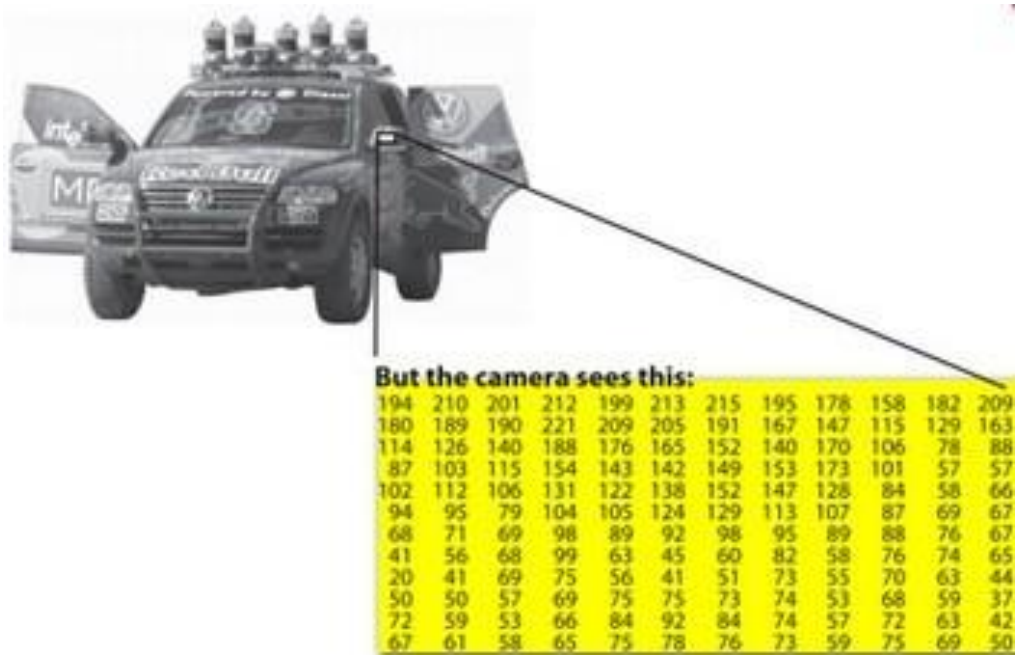
      /// Wait until user finishes program
      while(true)
      {
            int c;
            c = waitKey( 20 );
            if( (char)c == 27 )
                        { break; }

    }
}
```

[1] Mat - The Basic Image Container (1)

Mat Class

- *Mat* is basically a class with two data parts:
  - **the matrix header** (containing information such as the size of the matrix, the method used for storing, at which address is the matrix stored, and so on)
  - a pointer to **the matrix containing the pixel values** (taking any dimensionality depending on the method chosen for storing) .



[1] Mat - The Basic Image Container (2)

- 데이터 구조: Mat class

Mat (int **rows**, int **cols**, int **type**)

Mat (**Size** **size**, int **type**)

Mat (int **rows**, int **cols**, int **type**, const **Scalar** &s)

Mat (**Size** **size**, int **type**, const **Scalar** &s)

```
Example)  Mat frame;                                  // default buffer
          Mat image(324, 240, CV_8UC1) ;  // Size: 320x240, 8bit unsigned char type, 1channel
          Mat image(W, H, CV_8UC3);          // Size: WxH, 8bit unsigned char type, 3channel
          Mat image(324, 240, CV_8UC1, Scaler(120)) ; // Size:320x240, 8bit unsigned char type,
                                                      1channel, Set value for all pixel: 120
```

[1] Mat - The Basic Image Container (3)

- 선언 방법

  Mat 이름(가로크기, 세로 크기, 데이터 타입/채널 수, 초기값);

  ```
  Mat M(2,2, CV_8UC3, Scalar(0,0,255));
  cout << "M = " << endl << " " << M << endl << endl;
  ```

  - 실행 결과:

  ```
  M =
   [0, 0, 255, 0, 0, 255;
    0, 0, 255, 0, 0, 255]
  ```

[1] Mat - The Basic Image Container (3)

데이터 타입

CV_8U : 8-bit unsigned integer: uchar (0..255)
CV_8S : 8-bit signed integer: schar (-128..127)
CV_16U : 16-bit unsigned integer: ushort (0..65535)
CV_16S : 16-bit signed integer: short (-32768..32767)
CV_32S : 32-bit signed integer: int (-2147483648..2147483647)
CV_32F : 32-bit floating-point number: float (-FLT_MAX..FLT_MAX, INF, NAN)
CV_64F : 64-bit floating-point number: double (-DBL_MAX..DBL_MAX, INF, NAN)

다중채널 타입(Multi-channel (n-channel) types)

CV_8UC숫자 : 숫자 채널의 8-bit unsigned integer: uchar (0..255) 채널 3개
CV_8SC숫자 : 숫자 채널의 8-bit signed integer: schar (-128..127)
CV_16UC숫자 : 숫자 채널의 16-bit unsigned integer: ushort (0..65535)
CV_16SC숫자 : 숫자 채널의 16-bit signed integer: short (-32768..32767)
CV_32SC숫자 : 숫자 채널의 32-bit signed integer: int (-2147483648..2147483647)
CV_32FC숫자 : 숫자 채널의 32-bit floating-point number: float (-FLT_MAX..FLT_MAX, INF, NAN)
CV_64FC숫자 : 숫자 채널의 64-bit floating-point number: double
(-DBL_MAX..DBL_MAX, INF, NAN)

[2] 화소 점 처리(pixel point processing)실습:

- OpenCV4.1.1 (over 2.xx)에서 화소(pixel) 접근법
  - 3 channels (RGB 칼러 영상)

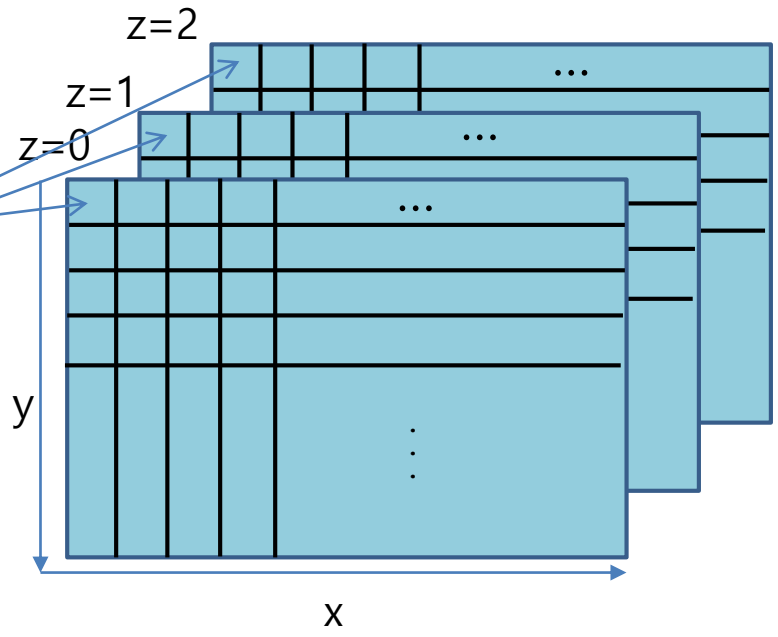
```
Mat img = imread(filename)

Vec3b intensity = img.at<Vec3b>(y, x);
uchar blue = intensity.val[0];
uchar green = intensity.val[1];
uchar red = intensity.val[2];
```

또는

```
Mat img = imread(filename)

uchar blue = img.at<Vec3b>(y, x)[0];
uchar green = img.at<Vec3b>(y, x)[1];
uchar red = img.at<Vec3b>(y, x)[2];
```



cv::Vec3b는 세 개의 unsigned char인 벡터
- image.at<cv::Vec3b>(j ,i)[channel] = value; // channel은 세 개 컬러 채널 중 하나
- Vec3b의 b는 byte를 의미하고, short의 s, int의 i, float의 f, double의 d가 가능함

[2] 화소 점 처리(pixel point processing)실습: 영상 반전

- 반전 화소값→ 255-현재 화소값

```
int main(int argc, char** argv) {
   //--OpenCV 4.x 구현 --//
   Mat image, result;

   image = imread("Desert.bmp", IMREAD_COLOR);   // Read the file
   result = image.clone();

   if (image.empty()){                              // Check for invalid input
       cout << "Could not open or find the image" << std::endl;
       return -1;
   }
   namedWindow("Display window", WINDOW_AUTOSIZE);  // Create a window for display.
   imshow("Display window", image);                 // Show our image inside it.

   //-- 1) pixel by pixel 반전 영상 생성 --//
   for (int i=0; i<image.rows; i++){
            for (int j=0; j<image.cols ; j++){
                     result.at<Vec3b>(i,j)[0] = 255-image.at<Vec3b>(i,j)[0];
                     result.at<Vec3b>(i,j)[1] = 255-image.at<Vec3b>(i,j)[1];
                     result.at<Vec3b>(i,j)[2] = 255-image.at<Vec3b>(i,j)[2];
            }
   }

   namedWindow("Processed image"); // Create a window for display.
   imshow("Processed image", result);// Show our image inside it.

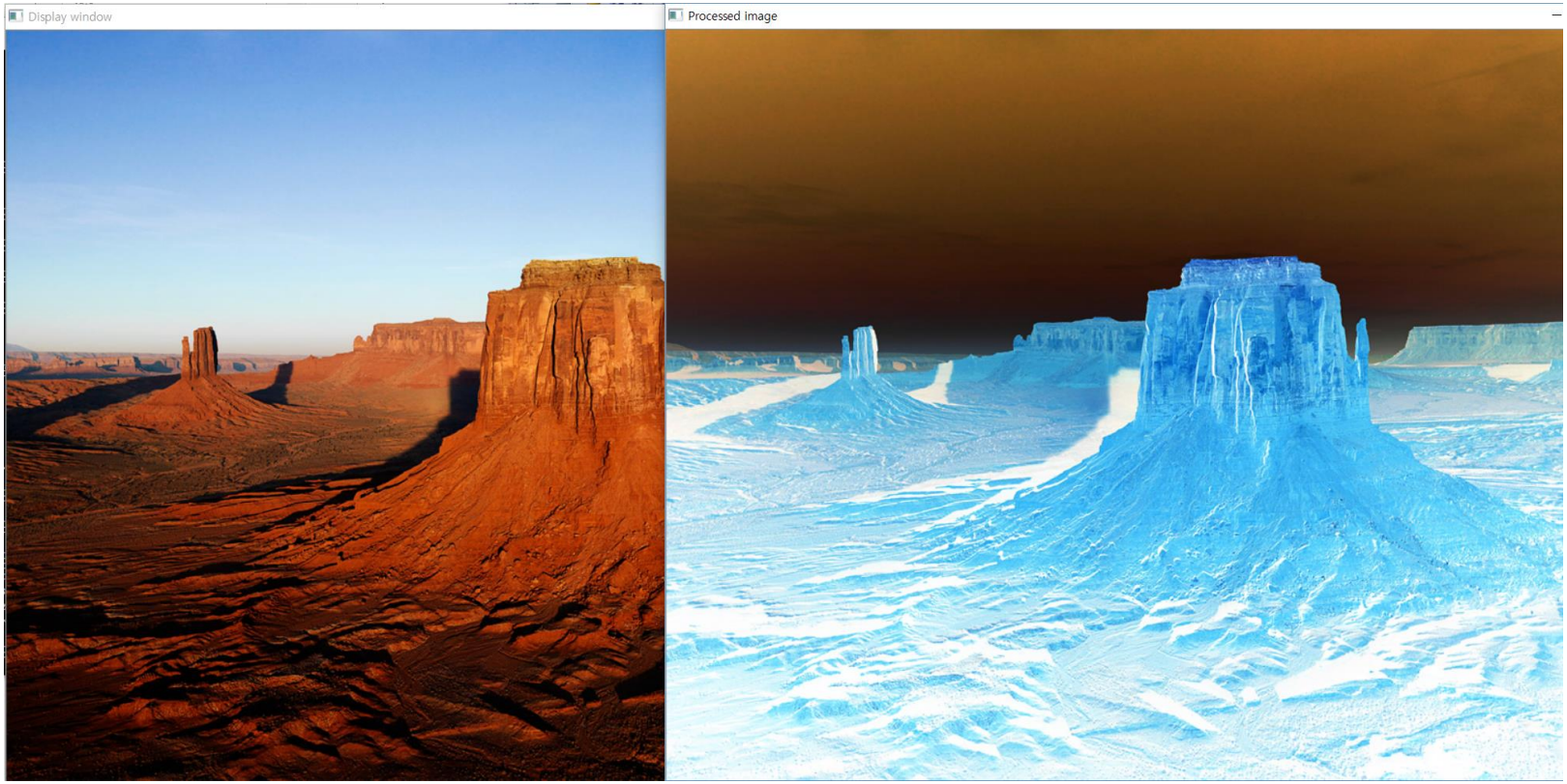
   waitKey(0);                                      // Wait for a keystroke in the window

    destroyAllWindows();

   return 0;
}
```

[2] 화소 점 처리(pixel point processing)실습: 영상 반전

- 수행 결과: 반전된 컬러 영상 출력 함



[3] Open CV APIs: Mat x data structure

- **Mat x = imread("입력파일 이름", color flag);**

y = x.**clone()**: copy image x and make new matrix image y.
x.**empty()** : Returns true if the array has no elements.
Mat y=**ones** (int **rows**, int **cols**, int **type**): Returns an array of all 1's of the specified size and type.
Mat y=**zeros** (int **rows**, int **cols**, int **type**): Returns an array of all 0's of the specified size and type.
x.**depth**() : Returns the depth (color band) of a matrix element.
x.**cols** : the number of columns
x.**rows** : the number of rows
x.**copyTo**(**OutputArray** m): Copies the matrix x to another one.

[3] Open CV APIs: Mat x=imread()

- imread("입력파일 이름", color flag)
  - Creates a window
- 사용법
  - **Mat img = imread("입력파일 이름", color flag)**
  - Mat **cv::imread** (const **String** &filename, int flags=**IMREAD_COLOR**)

Parameters:

•finename – Name of the input image file.
•Color flags – Flags of the color or grayscale. The supported flags are:

- **cv::IMREAD_UNCHANGED** = -1,
  cv::IMREAD_GRAYSCALE = 0,
  cv::IMREAD_COLOR = 1,
  cv::IMREAD_ANYDEPTH = 2,
  cv::IMREAD_ANYCOLOR = 4,
  cv::IMREAD_LOAD_GDAL = 8,
  cv::IMREAD_REDUCED_GRAYSCALE_2 = 16,
  cv::IMREAD_REDUCED_COLOR_2 = 17,
  cv::IMREAD_REDUCED_GRAYSCALE_4 = 32,
  cv::IMREAD_REDUCED_COLOR_4 = 33,
  cv::IMREAD_REDUCED_GRAYSCALE_8 = 64,
  cv::IMREAD_REDUCED_COLOR_8 = 65,
  cv::IMREAD_IGNORE_ORIENTATION = 128.

---

[3] Open CV APIs: namedWindow()

- namedWindow("윈도우 이름", size flag)
  - Creates a window
- 사용법
  - void namedWindow(const string& **winname**, int **flags**=WINDOW_AUTOSIZE)

Parameters:	•name – Name of the window in the window caption that may be used as a window identifier. •flags – Flags of the window. The supported flags are: • WINDOW_NORMAL If this is set, the user can resize the window (no constraint). • WINDOW_AUTOSIZE If this is set, the window size is automatically adjusted to fit the displayed image (see imshow()), and you cannot change the window size manually. • WINDOW_OPENGL If this is set, the window will be created with OpenGL support.

[3] Open CV APIs: imshow()/destroyAllWindows

- imshow("윈도우 이름", 영상데이터 버퍼)
  - Displays an image in the specified window
- 사용법
  - void imshow(const string& **winname**, InputArray **mat**)

Parameters:	• **winname** – Name of the window. • **image** – Image to be shown.

- destroyAllWindows
  - Destroys all of the HighGUI windows.

[3] Open CV APIs: waitkEY() - 입력 대기 관련 함수

- waitkey(n)함수
  - n msec.만큼 임의의 키 입력을 대기함
- 사용법
  - Char ch = waitKey(10); // 10ms동안 입력을 대기 후 키 입력은 ch에 ASCII 코드로 맵핑
  - If(ch == 27) break; // 27 == ESC key
  - If(ch == 32) // 32 == SPACE key
  - waitkey() 또는 waitkey(0): 키 입력까지 무한 대기함

[3] Open CV APIs: Data class <Vec3b>

- <**cv::Vec3**b/s/i/f/d>
  - Set 3-dim vector of each position (x, y) or (x ,y, z) as date type is b=byte, s=short, i=integer, f=float and d=double types.

- **at**() api (cv::Mat::at<data type>(pointer of location))
  - Returns a reference (value) to the specified array element.

```
Mat H(100, 100, CV_64F);
for(int i = 0; i < H.rows; i++)
    for(int j = 0; j < H.cols; j++)
        H.at<double>(i,j)=1./(i+j+1);
```

•If matrix is of type CV_8U then use Mat.at<uchar>(y,x).
•If matrix is of type CV_8S then use Mat.at<schar>(y,x).
•If matrix is of type CV_16U then use Mat.at<ushort>(y,x).
•If matrix is of type CV_16S then use Mat.at<short>(y,x).
•If matrix is of type CV_32S then use Mat.at<int>(y,x).
•If matrix is of type CV_32F then use Mat.at<float>(y,x).
•If matrix is of type CV_64F then use Mat.at<double>(y,x).

[4] 화소 점 처리(pixel point processing)실습: 색상 변화

- 색상 변화

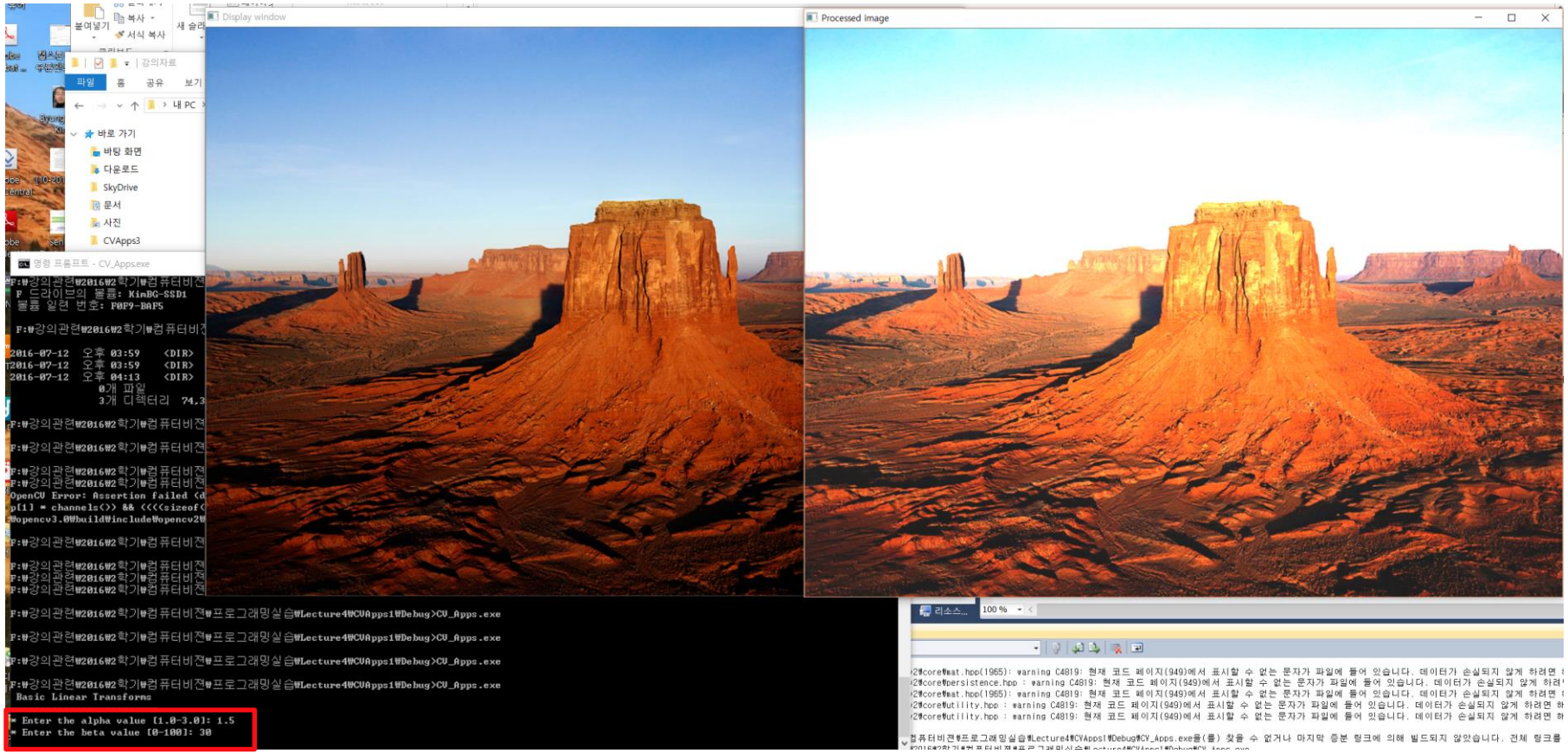
```
double alpha; /**< Simple contrast control */
int beta; /**< Simple brightness control */
int main( int argc, char** argv ) {
        /// Read image given by user
        Mat image = imread( argv[1] );
        Mat new_image = Mat::zeros( image.size(), image.type() );
        /// Initialize values
        std::cout<<" Basic Linear Transforms "<<std::endl;
        std::cout<<"-------------------------"<<std::endl;
        std::cout<<"* Enter the alpha value [1.0-3.0]: ";
        std::cin>>alpha;
        std::cout<<"* Enter the beta value [0-100]: ";
        std::cin>>beta;

        /// Do the operation new_image(i,j) = alpha*image(i,j) + beta
        for( int y = 0; y < image.rows; y++ ) {
                for( int x = 0; x < image.cols; x++ ) {
                        for( int c = 0; c < 3; c++ ) {
                                new_image.at<Vec3b>(y,x)[c] = saturate_cast<uchar>( alpha*( image.at<Vec3b>(y,x)[c] ) + beta );
                        }
                }
        }
        /// Create Windows
        namedWindow("Original Image", 1);
        namedWindow("New Image", 1);
        /// Show stuff
        imshow("Original Image", image);
        imshow("New Image", new_image);

        /// Wait until user press some key
        waitKey();
        return 0;
}
```

[4] 화소 점 처리(pixel point processing)실습: 색상 변환

- 수행결과
  - 입력한 값에 의해 색상 변환이 된 영상 출력함



[5] 화소 점 처리(pixel point processing)실습: 영상 이진화

- 이진화
  - **기본적으**로 영상을 2개의 영역으로 분할하는 것

~~~~

```
 //-- 3)영상 이진화     ---//
std::cout<<"* Enter threshold value [0~255]: ";
std::cin>>Thres;

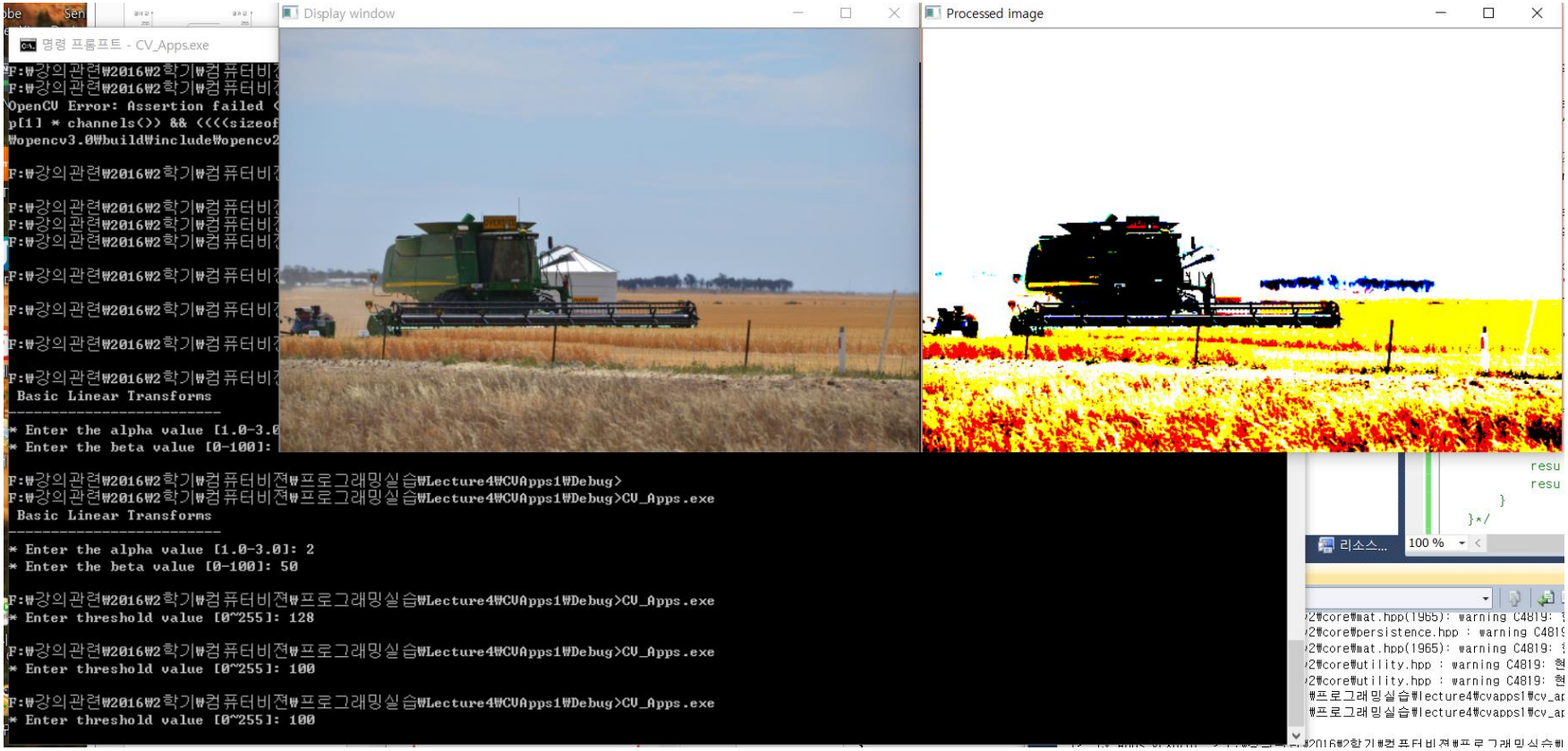
for (int i=0; i<image.rows; i++){
        for (int j=0; j<image.cols ; j++){
                if (image.at<Vec3b>(i,j)[0]>Thres){
                        result.at<Vec3b>(i,j)[0] =255;
                }else{
                        result.at<Vec3b>(i,j)[0] =0;
                }
                if (image.at<Vec3b>(i,j)[1]>Thres){
                        result.at<Vec3b>(i,j)[1] =255;
                }else{
                        result.at<Vec3b>(i,j)[1] =0;
                }
                if (image.at<Vec3b>(i,j)[2]>Thres){
                        result.at<Vec3b>(i,j)[2] =255;
                }else{
                        result.at<Vec3b>(i,j)[2] =0;
            }
        }
}

namedWindow("Processed image"); // Create a window for display.
imshow("Processed image", result);// Show our image inside it.
```

~~~

[5] 화소 점 처리(pixel point processing)실습: 영상 이진화

- 수행 결과
  - 컬러 이진화 된 것을 볼 수 있음.



[6] 화소 점 처리(pixel point processing)실습: 감마 보정

- 감마보정

```
int main(int argc, char** argv) {

    /// Read image given by user
    Mat src = imread(argv[1]);

    Mat dst = Mat(src.rows, src.cols, CV_8UC1);
    cvtColor(src, src, COLOR_BGR2GRAY);
    dst = Scalar(0);

     cout << " Basic Linear Transforms: Gamma correction " << std::endl;
    std::cout << "-------------------------" << std::endl;
    std::cout << "* Enter the gamma value [1.0-3.0]: ";
    std::cin >> gamma;

    for (int x = 0; x < src.rows; x++) {
        for (int y = 0; y < src.cols; y++) {
            int pixelValue = (int)src.at<uchar>(x, y);

            dst.at<uchar>(x, y) = pow(pixelValue, gamma);

        }
    }

    namedWindow("Input", 1);
    namedWindow("Output", 1);

    imshow("Input", src);
    imshow("Output", dst);

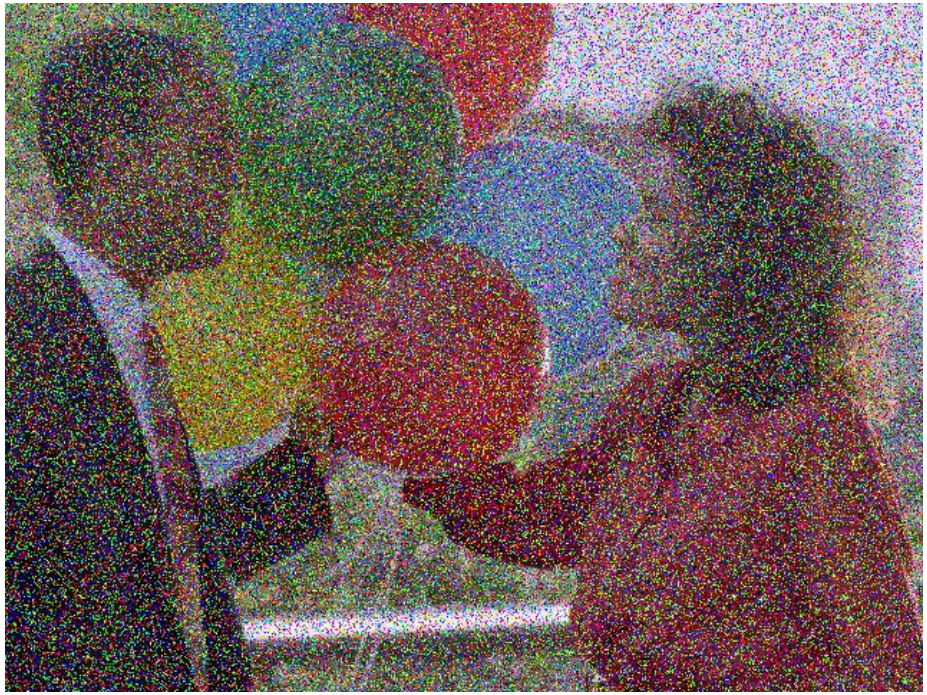
    waitKey(0);
    return 0;
}
```

[6] 화소 점 처리(pixel point processing)실습: 감마 보정

- 수행 결과

[7] 화소 점 처리(pixel point processing)실습: Salt-Pepper잡음 생성

- Salt-Pepper 잡음이란?
  - 영상에 넓게 퍼진 형태로 무작위적인 희고 검은 점이 발생하는 잡음의 한 종류
    - 위치: 무작위, 값: 0 또는 255



[7] 화소 점 처리(pixel point processing)실습: Salt-Pepper잡음 생성

예제 코드의 이해

```
~~
void salt(Mat &img, int n); // 잡음 추가 함수 선언

int main(int argc, char** argv)
{

//--OpenCV 2.x 구쐼?현o --//
Mat image, result;

   image = imread("test.jpg", IMREAD_COLOR);   // Read the file

   if (image.empty()){                              // Check for invalid input
       cout << "Could not open or find the image" << std::endl;
       return -1;
   }
   namedWindow("Display window", WINDOW_AUTOSIZE);  // Create a window for display.
   imshow("Display window", image );                 // Show our image inside it.

   result = image.clone();
   salt(result, 3000);

   namedWindow("Processed image"); // Create a window for display.
    imshow("Processed image", result);// Show our image inside it.

   waitKey(0);                                          // Wait for a keystroke in the window

    destroyAllWindows();

   return 0;
}
   ~~~
```

[7] 화소 점 처리(pixel point processing)실습: Salt-Pepper잡음 생성

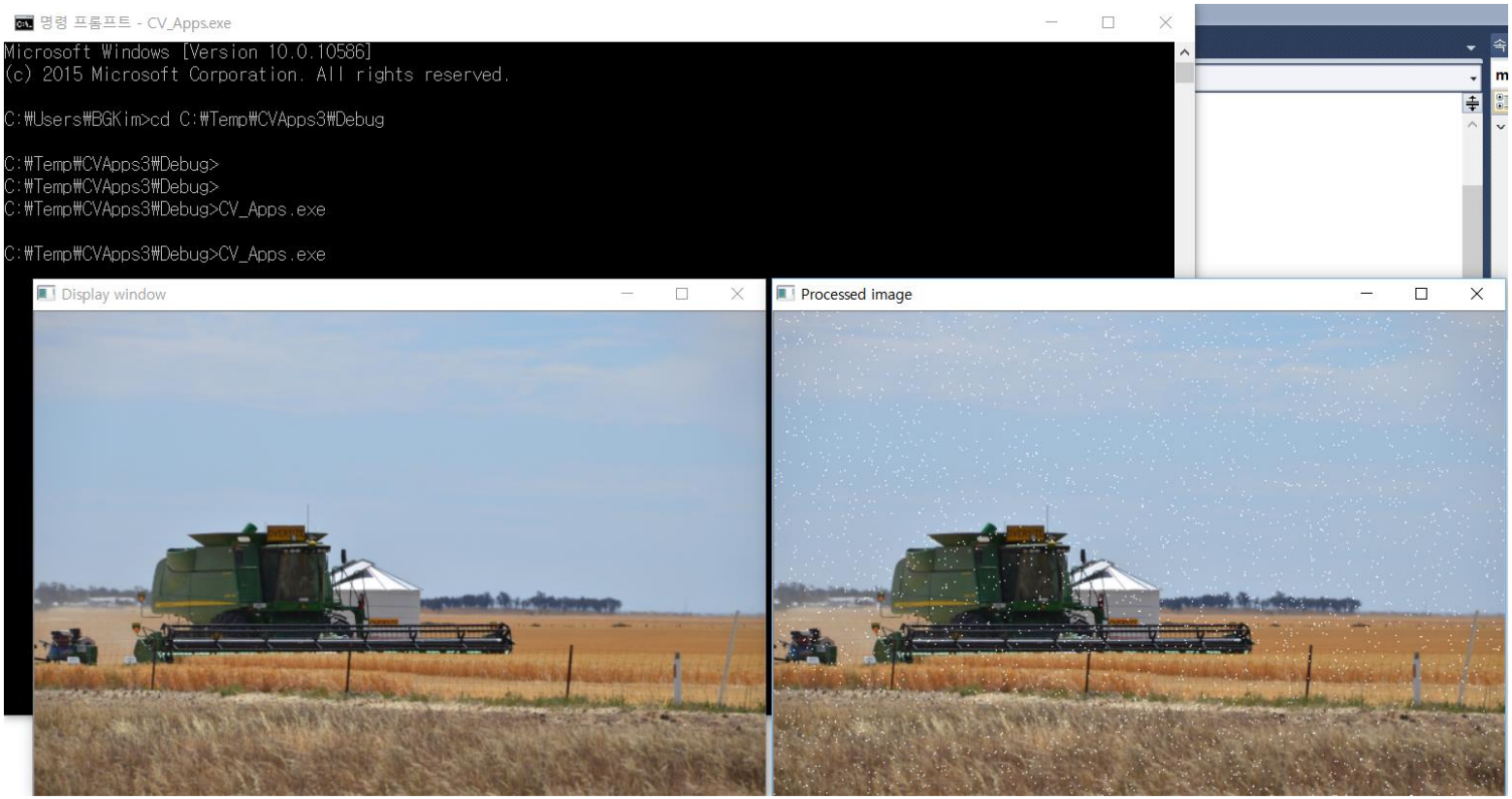
- 예제 코드의 이해

```
void salt(Mat &img, int n)
{
	for (int k=0; k<n ; k++){
		int i= rand()%img.cols;
		int j= rand()%img.rows;

		if (img.channels()==1){		// Gray scale image
			img.at<uchar>(j,i) = 255;
		}else if (img.channels()==3){	// Color image
			img.at<Vec3b>(j,i)[0] = 255;
			img.at<Vec3b>(j,i)[1] = 255;
			img.at<Vec3b>(j,i)[2] = 255;
		}
	}
}
```

[7] 화소 점 처리(pixel point processing)실습: Salt-Pepper잡음 생성

- 실행 결과: 입력한 n개의 salt-and-pepper 잡음이 추가 됨



[8] 화소 점 처리(pixel point processing)실습: 컬러 reduction

- 화소(Pixel) 처리를 위한 메모리 데이터처리 기본 방식
  - Pointer 사용

```
uchar *data = image.ptr<uchar>(j);
```

  - Iterator 사용

```
Mat_<Vec3b>::iterator it=image.begin<Vec3b>();
Mat_<Vec3b>::iterator itend=image.end<Vec3b>();
```

  - at() 함수 사용

```
int B= image.at<Vec3b>(j,i)[0];
int G = image.at<Vec3b>(j,i)[1];
int R = image.at<Vec3b>(j,i)[2];
```

[8] 화소 점 처리(pixel point processing)실습: 컬러 reduction

- Color Posterizing (quantization)
  - 256개의 컬러를 임의의 N개 컬러영역 영상으로 변환

```
  ~~~
//--OpenCV 2.x 구쐼?현o --//
Mat image, result;

   image = imread("test.jpg", IMREAD_COLOR);   // Read the file

   if (image.empty()){                              // Check for invalid input
       cout << "Could not open or find the image" << std::endl;
       return -1;
   }
   namedWindow("Display window", WINDOW_AUTOSIZE);  // Create a window for display.
   imshow("Display window", image );                   // Show our image inside it.

   result = image.clone();
   colorReduce(result, 64);

   namedWindow("Processed image"); // Create a window for display.
    imshow("Processed image", result);// Show our image inside it.

   waitKey(0);
~~~
```

[8] 화소 점 처리(pixel point processing)실습: 컬러 reduction

- 컬러 수 감소 함수 구현: **pointer로 구현 시**

```
void colorReduce(Mat &image, int div){
        int nl = image.rows;//행개수
        int nc = image.cols*image.channels();//각행의 데이터 개수

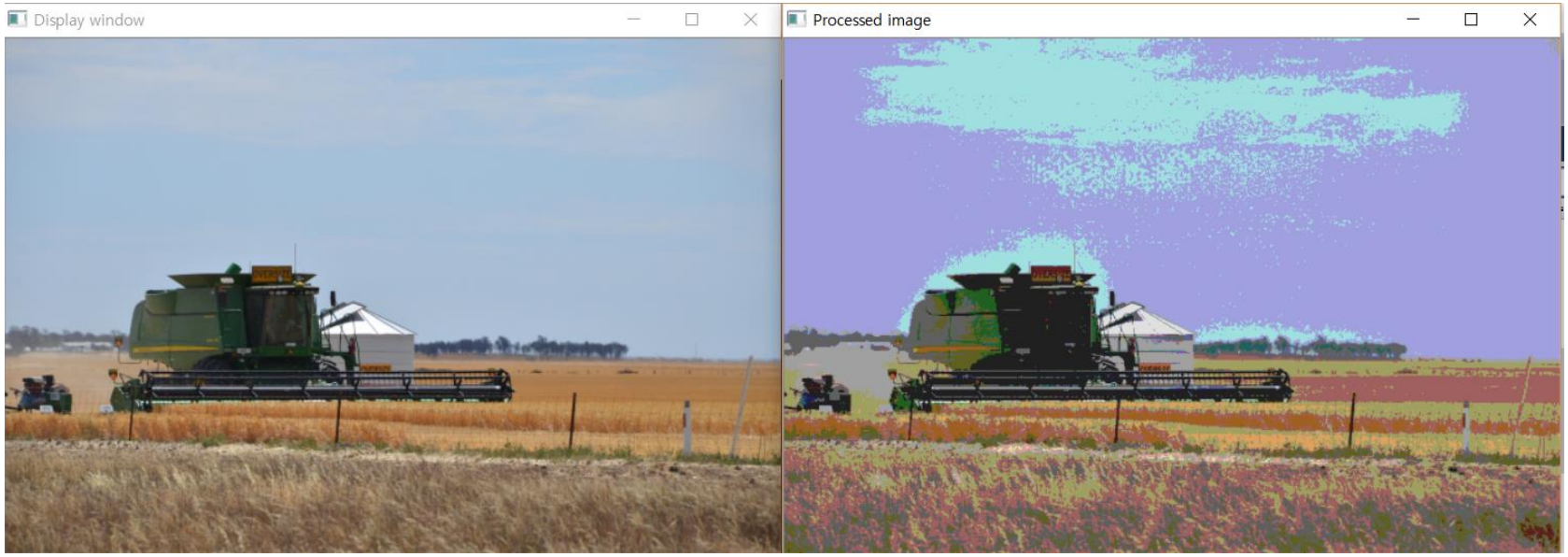
        for (int j=0; j<nl ; j++){

                //-- j열의 주소 (nc 개만큼) 가져오기 --//
                uchar *data = image.ptr<uchar>(j);
                for (int i=0; i<nc ; i++){

                        //---각 화소값 분할--//
                        data[i] = data[i]/div*div + div/2;
                }
        }
}
```

[8] 화소 점 처리(pixel point processing)실습: 컬러 reduction

- Color Posterizing (quantization)
  - 256개의 컬러를 임의의 N개 컬러 영상으로 변환



[8] 화소 점 처리(pixel point processing)실습: 컬러 reduction

- 컬러 수 감소 함수 구현: **iterator로 구현 시**

```
void colorReduce1(Mat &image, int div){

        Mat_<Vec3b>::iterator it=image.begin<Vec3b>();
        Mat_<Vec3b>::iterator itend=image.end<Vec3b>();

        //모든 화소 조회 //
        for (; it!=itend ; ++it){
                //--- 개별화소 처리---//
                (*it)[0] = (*it)[0]/div*div + div/2;
                (*it)[1] = (*it)[1]/div*div + div/2;
                (*it)[2] = (*it)[2]/div*div + div/2;
        }

}
```

[8] 화소 점 처리(pixel point processing)실습: 컬러 reduction

- 컬러 수 감소 함수 구현: **at 메소드로 구현 시**

```
void colorReduce2(Mat &image, int div){

        int nl = image.rows;//행 개수
        int nc = image.cols; //열 개수

        //모든 화소 조회¸ //
        for (int j=0; j<nl ; j++){
                for (int i=0; i<nc ; i++){
                        //---각화소 처리--//
                        image.at<Vec3b>(j,i)[0] = image.at<Vec3b>(j,i)[0]/div*div + div/2;
                        image.at<Vec3b>(j,i)[1] = image.at<Vec3b>(j,i)[1]/div*div + div/2;
                        image.at<Vec3b>(j,i)[2] = image.at<Vec3b>(j,i)[2]/div*div + div/2;
                }
        }

}
```

[8] OpenCV 4.1에서 실행 시간 측정: cv::getTickCount() 함수

- **getTickCount()**
  - Returns the number of ticks (clocks)
  - **Int64 getTickCount()**

- **getTickFrequency()**
  - Returns the number of ticks per second.
  - **double getTickFrequency()**

- 기본 사용법

```
double t = (double)getTickCount();
// do something ...
t = ((double)getTickCount() - t)/getTickFrequency();
```

[8] OpenCV 4.1에서 실행 시간 측정: cv::getThickCount() 함수

- 실제 실습 코드 이해 (main 함수 내에)

```
  ~~~~
result = image.clone();
result1 = image.clone();
result2 = image.clone();
//salt(result, 3000);
t = getTickCount();
colorReduce(result, 64);
t = ((double)getTickCount() - t)/getTickFrequency();

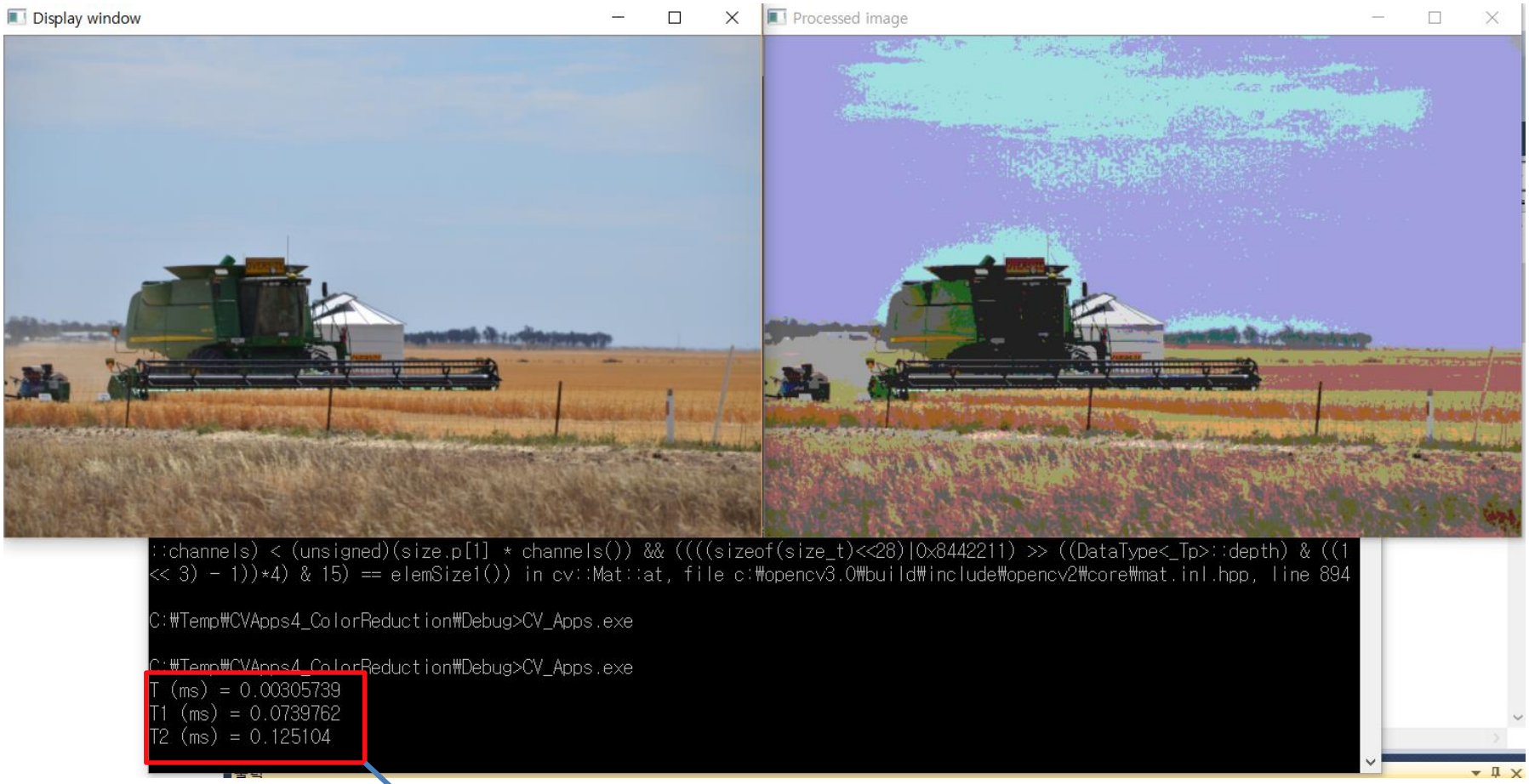
t1 = getTickCount();
colorReduce1(result, 64);
t1 = ((double)getTickCount() - t1)/getTickFrequency();

t2 = getTickCount();
colorReduce2(result, 64);
t2 = ((double)getTickCount() - t2)/getTickFrequency();

cout << "T (ms) = " << t << std::endl;
cout << "T1 (ms) = " << t1 << std::endl;
cout << "T2 (ms) = " << t2 << std::endl;
~~~~
```

[8] OpenCV 3.0에서 실행 시간 측정: cv::getThickCount() 함수

- 실제 수행 결과: **for 문 구조에 따른 처리 속도 비교**



포인터 사용: T (ms) = 0.00305739
Iterator사용: T1 (ms) = 0.0739762
3 채널 분리 사용: T2 (ms) = 0.125104

OpenCV1.x 영상데이터 <--> OpenCV4.x 영상데이터

- IplImage * → cv::Mat 변환
  - Mat Mat_img = cvarrToMat(IplImage_img);
  - Mat Mat_img(IplImage_img);

- Cv::Mat → IplImage * 변환
  - IplImage *IplImage_img = new IplImage(Mat_img);

과제#1 –
화소 데이터 처리 3가지 방법에 의한 화소 처리 시간 측정 실험 및 분석

내용

- 강의자료 pp. 41-51 의 내용을 구현하여 실제 시간을 측정하기
- 개별 처리 시간 비교 및 속도 차이가 나는 이유 고민해 보기
- 소스코드 및 결과 화면 캡쳐하여 반드시 첨부할 것
- 보고서 표지도 작성할 것 (과제#xx ~~~~~)

작성방법

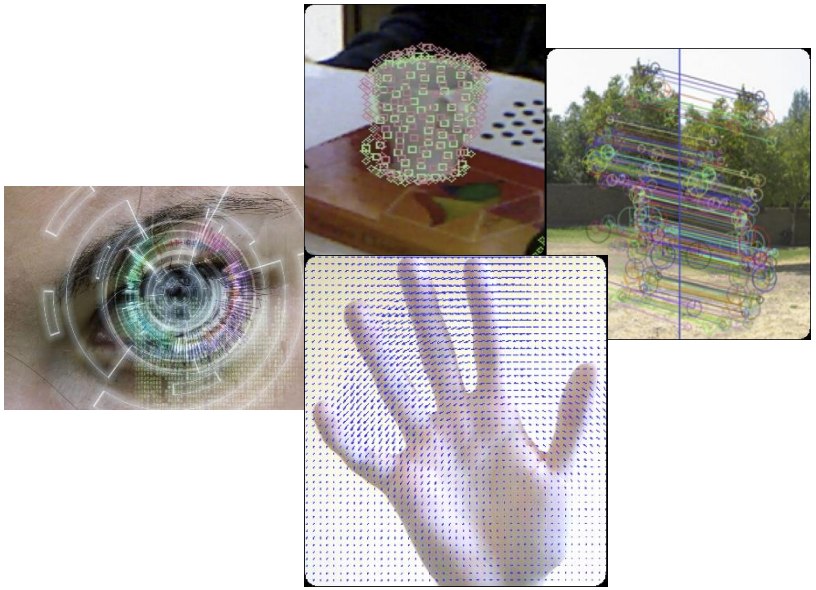
- 속도 등에 의한 차이는 수기로 그 이유나 원인을 분석해서 작성할 것

제출 기한

- 9월 18일 (월) 수업 시작 전까지~~~

COMPUTER VISION 비젼 프로그래밍

Thank you and Question?



IVPL
Intelligent Vision Processing Lab