

**졸업 프로젝트 보고서**(2018. 1학기)

프로젝트 제목 : 불법차량 조회 시스템

|  |  |  |  |
| --- | --- | --- | --- |
| 학 과 | 컴퓨터공학과 | 교과목명 | 졸업프로젝트 |
| 학 년 | 4 |
| 학 번,  이 름 | 20135141 안현식 20135145 우종훈  20135154 이동현 20135185 황석진 | | |

**목차**

1. 개발 동기
2. 국내 번호판의 종류
3. 검출
4. 인식
5. 조회 프로그램
6. 문제점
7. 소감
8. **개발 동기**

최근 급증하는 불법차량으로 인해 손해를 보는 피해자들도 늘어나고 있습니다. 그래서 국가 치안에 기여하고 싶었고 팀원이 4명이라는 점을 감안하여 팀원들이 자신의 분야를 맡아 할 수 있는 일을 하고 싶었습니다. 그래서 영상처리(번호판 검출), AI(번호인식), 폼구성 및 DB조회 등 다양한 분야를 공부함으로써 혼자서는 쉽게 접하지 못할 것 같은 주제를 선택해 저희 팀원 각각의 역량을 합쳐 하나의 작품을 만들고 싶었습니다.

1. **국내 번호판 종류**



현재 국내에 있는 번호판의 종류는 위 사진에 있는 번호판뿐만 아니라 더 많은 종류의 번호판이 있고 글자는 40개가있습니다.

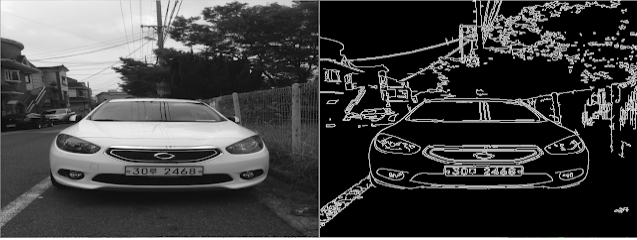
1. **번호판 검출**
2. **동영상에서 사진 캡쳐**

#include <opencv2/core/core.hpp>  
#include <opencv2/highgui/highgui.hpp>  
  
using namespace cv;  
  
int main()  
{  
Mat img; // 동영상 파일 재생을 위한 행렬 선언  
VideoCapture cap(0); // 캠으로부터 영상을 받아온다 (혹은 1번)  
  
char savefile[200]; // 이미지 파일 이름을 200자 이내로 제한하기 위한 char 변수 선언  
  
for(int i=0 ; ; )   
{  
cap >> img; // 동영상 파일을 img에 저장한다.  
imshow("image", img); // 이미지를 출력한다.  
  
sprintf(savefile, "image%d.jpg", i++);  
imwrite(savefile, img); // img를 파일로 저장한다.  
waitKey(100); // 100ms의 딜레이를 발생시킨다. 즉, 1초에 10장의 이미지를 저장하게 된다.  
  
if(waitKey(1) >= 0) // 키보드로부터 입력을 받으면 동영상 재생을 멈춘다.  
break;  
}  
return 0;  
}

1. **캡쳐된 사진에서 번호판 검출 후 분할**
2. **그레이 레벨(cvtColor)**



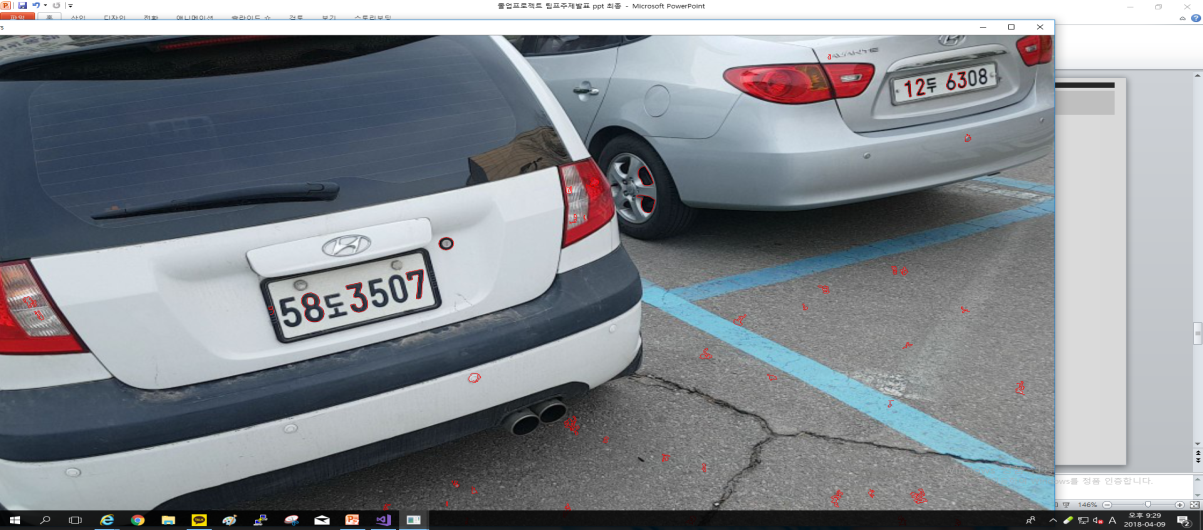
1. **전처리 과정(Canny())**

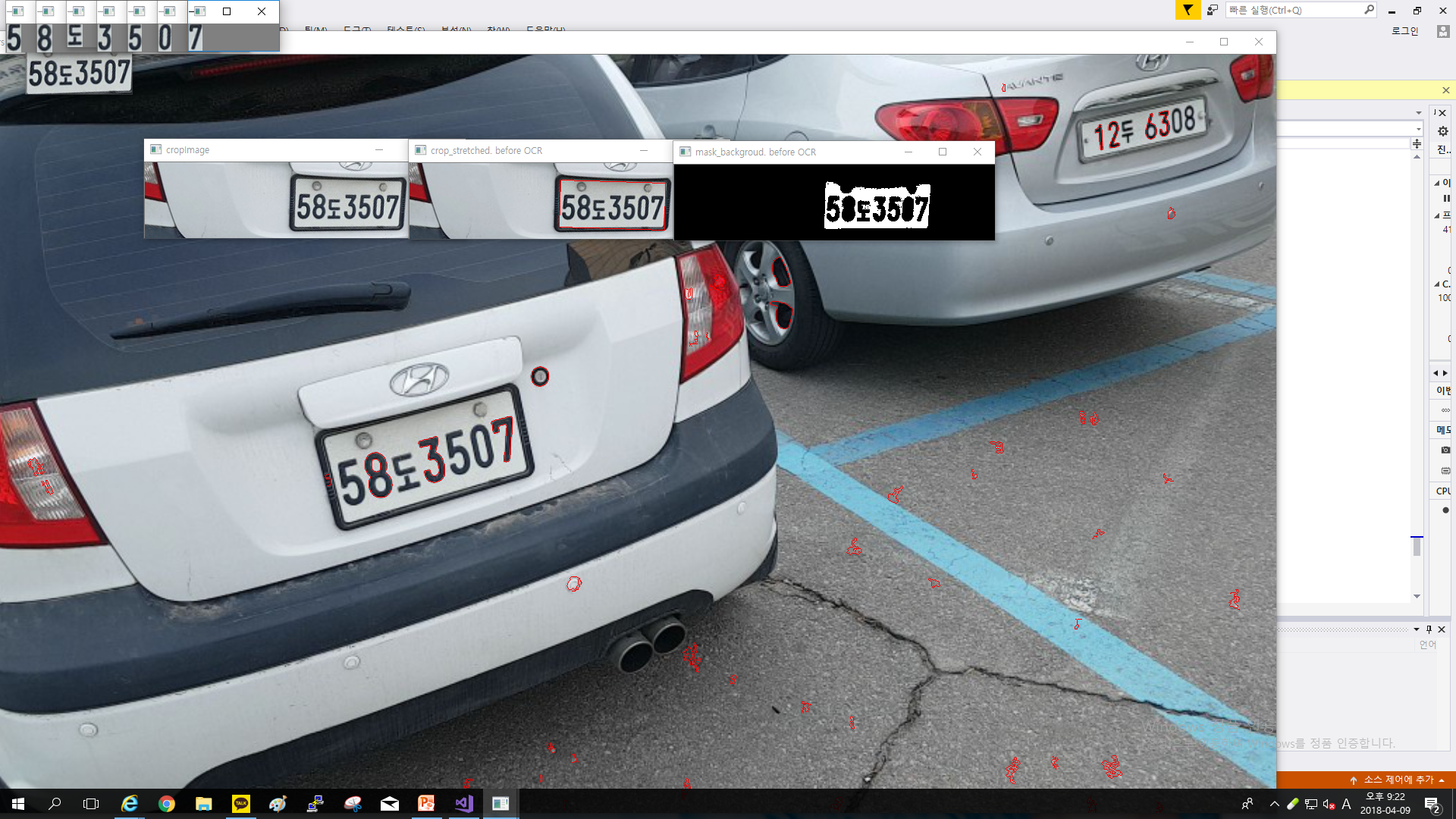


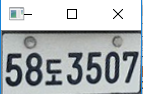
1. **색과 색 사이에 경계선(findContours()) & 사각형 배열**



1. **번호판 검출**







**4.인식**

opencv에서 제공하는 svm을 사용하여 많은 데이터를 트레이닝 한 후 인식률을 높이는 것이 목적입니다.

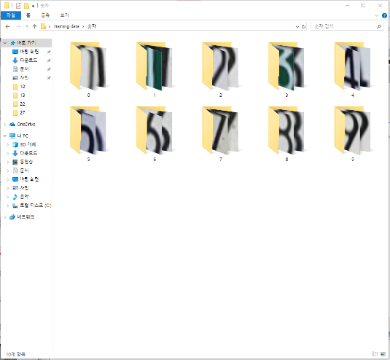
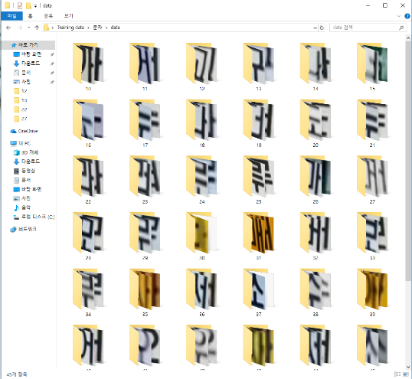
svm소스

const int number\_of\_class = 50;

const int number\_of\_sample = 2400;

const int number\_of\_feature = 32;

#number\_of\_class에 데이터 폴더의 개수를 입력받고 number\_of\_sample에 각 폴더에 있는 데이터의 파일 개수를 입력받습니다.



위에 보시는 사진은 저희 팀원들이 모두 모은 데이터를 합쳐 폴더 안에 넣은 것으로 폴더의 개수갯수는 총50개이기 때문에 number\_of\_class에 50을 넣고 폴더 한 개 안에 있는 데이터의 개수는 모두 2400개이므로 number\_of\_sample에 2400을 넣었습니다.

Ptr<SVM> svm = SVM::create();

svm->setType(SVM::C\_SVC);

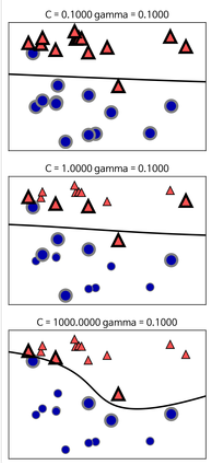
svm->setKernel(SVM::RBF);

svm->setGamma(0.5);

svm->setC(16);

svm->setTermCriteria(TermCriteria(TermCriteria::MAX\_ITER, 100, 1e-6));

#setKernel로 RBF커널 type을 이용하고 C값을 16으로 주어 직선이 아닌 곡선으로 주어 임계영역을 더 넓게 주어 희미하게 찍힌 이미지도 참인 값으로 인식하게 이미지 트레이닝 시켰습니다.



for (size\_t i = 0; i < folders.size(); ++i)

{

;

vector<string> files = list\_file(folders.at(i));

if (files.size() <= 0 || files.size() != number\_of\_sample)

{

printf("test");

return false;

}

string folder\_path = folders.at(i);

cout << "list folder" << folders.at(i) << endl;

string label\_folder = folder\_path.substr(folder\_path.length() - 1);

for (size\_t j = 0; j < files.size(); ++j)

{

src = imread(files.at(j));

if (src.empty())

{

return false;

}

vector<float> feature = calculate\_feature(src);

for (size\_t t = 0; t < feature.size(); ++t)

data.at<float>(index, t) = feature.at(t);

label.at<int>(index, 0) = i;

index++;

}

}

# 흑백을 구분한 반환 값을 가지고 레이블 값을 저장시켜준다.

Feature.h 소스

static int count\_pixel(Mat img, bool black\_pixel = true)

{

int black = 0;

int white = 0;

for (int i = 0; i < img.rows; ++i)

for (int j = 0; j < img.cols; ++j)

{

if (img.at<uchar>(i, j) == 0)

black++;

else

white++;

}

if (black\_pixel)

return black;

else

return white;

}

Mat img;

if (src.channels() == 3)

{

cvtColor(src, img, CV\_BGR2GRAY);

threshold(img, img, 100, 255, CV\_THRESH\_BINARY);

}

else

{

threshold(src, img, 100, 255, CV\_THRESH\_BINARY);

}

vector<float> r;

//vector<int> cell\_pixel;

resize(img, img, Size(40, 40));

int h = img.rows / 4;

int w = img.cols / 4;

int S = count\_pixel(img);

int T = img.cols \* img.rows;

for (int i = 0; i < img.rows; i += h)

{

for (int j = 0; j < img.cols; j += w)

{

Mat cell = img(Rect(i, j, h, w));

int s = count\_pixel(cell);

float f = (float)s / S;

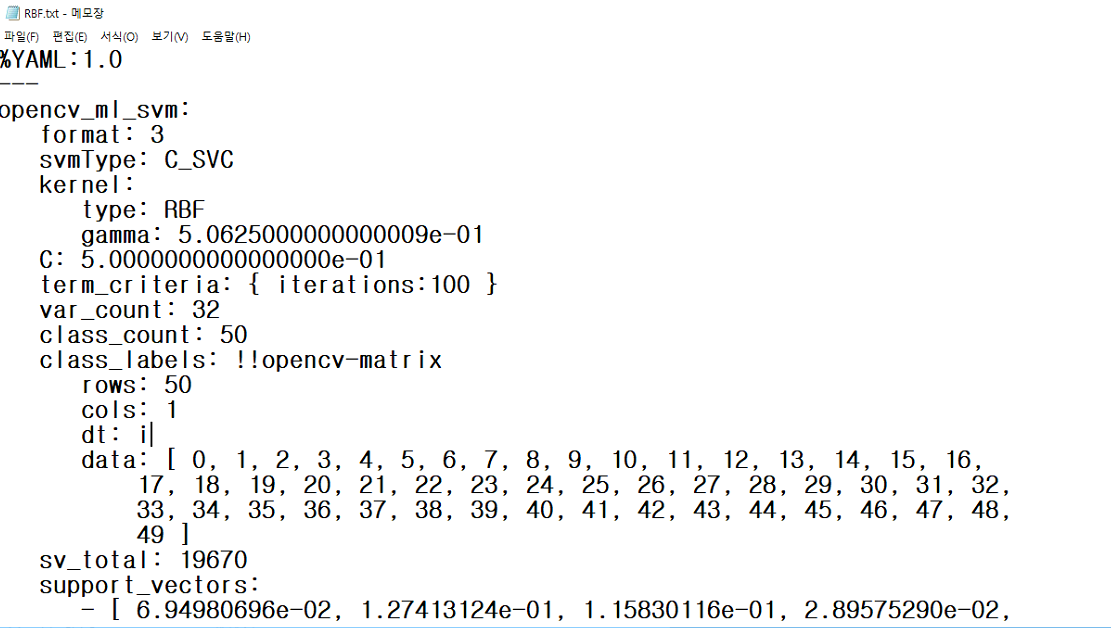
r.push\_back(f);

}

}

해당 이미지를 채널3(컬러)일 경우 흑백으로 이미지 변환 시킨 후 이미지를 이진화를 시킵니다.

이진화 된 이미지를 40x40으로 사이즈를 조정후 이미지 픽셀이 흑인지 백인지 구분하여 값을 따로 저장해줍니다.



**character\_recognition메소드**

// 문자 인식

//Load SVM training file OpenCV 3.1

Ptr<SVM> svmNew = SVM::create();

svmNew = SVM::load("RBF.txt");

char c = '\*';

vector<float> feature = calculate\_feature(img\_character);

// Open CV3.1

Mat m = Mat(1, number\_of\_feature, CV\_32FC1);

for (size\_t i = 0; i < feature.size(); ++i)

{

float temp = feature[i];

m.at<float>(0, i) = temp;

}

int ri = int(svmNew->predict(m)); // Open CV 3.1

# 학습시킨 데이터를 svmNew변수로 읽어들여 해당 데이터를 저장시킵니다. 그 후 번호판에서 번호를 자른 이미지에서 이전과 같은 방법으로 이진화를 시켜 흑과 백을 가린 정보를 feature변수에 저장하여 학습시킨 데이터를 통해 레이블 값을 찾아 ri에 저장시킨다.

//printf("ri : %d\n", ri);

if (ri >= 0 && ri <= 9)

c =(char)ri+48; //ma ascii 0 = 48

/\*if (ri == 10) c = (char)(ri + 44022); // 가

if (ri == 11) c = (char)(ri + 44133); // 거

if (ri == 12) c = (char)(ri + 44244); // 고

if (ri == 13) c = (char)(ri + 44383); // 구

if (ri == 14) c = (char)(ri + 45306); // 너

if (ri == 45) c = (char)(ri + 45163); // 나

if (ri == 15) c = (char)(ri + 45417); // 노

if (ri == 16) c = (char)(ri + 45556); // 누

if (ri == 17) c = (char)(ri + 45779); // 다

if (ri == 18) c = (char)(ri + 45890); // 더

if (ri == 19) c = (char)(ri + 46001); // 도

if (ri == 20) c = (char)(ri + 46140); // 두

if (ri == 21) c = (char)(ri + 46951); // 라

if (ri == 22) c = (char)(ri + 47062); // 러

if (ri == 23) c = (char)(ri + 47173); // 로

if (ri == 24) c = (char)(ri + 47312); // 루

if (ri == 25) c = (char)(ri + 47535); // 마

if (ri == 26) c = (char)(ri + 47646); // 머

if (ri == 27) c = (char)(ri + 47757); // 모

if (ri == 28) c = (char)(ri + 47896); // 무

if (ri == 44) c = (char)(ri + 48104); // 바

if (ri == 29) c = (char)(ri + 48231); // 버

if (ri == 30) c = (char)(ri + 48342); // 보

if (ri == 31) c = (char)(ri + 48481); // 부

// 사

if (ri == 32) c = (char)(ri + 49404); // 서

if (ri == 33) c = (char)(ri + 49515); // 소

if (ri == 34) c = (char)(ri + 49654); // 수

// 아

if (ri == 35) c = (char)(ri + 50577); // 어

if (ri == 36) c = (char)(ri + 50688); // 오

if (ri == 37) c = (char)(ri + 50827); // 우

if (ri == 38) c = (char)(ri + 51050); // 자

if (ri == 39) c = (char)(ri + 51161); // 저

if (ri == 40) c = (char)(ri + 51272); // 조

if (ri == 41) c = (char)(ri + 51411); // 주

if (ri == 42) c = (char)(ri + 54574); // 하

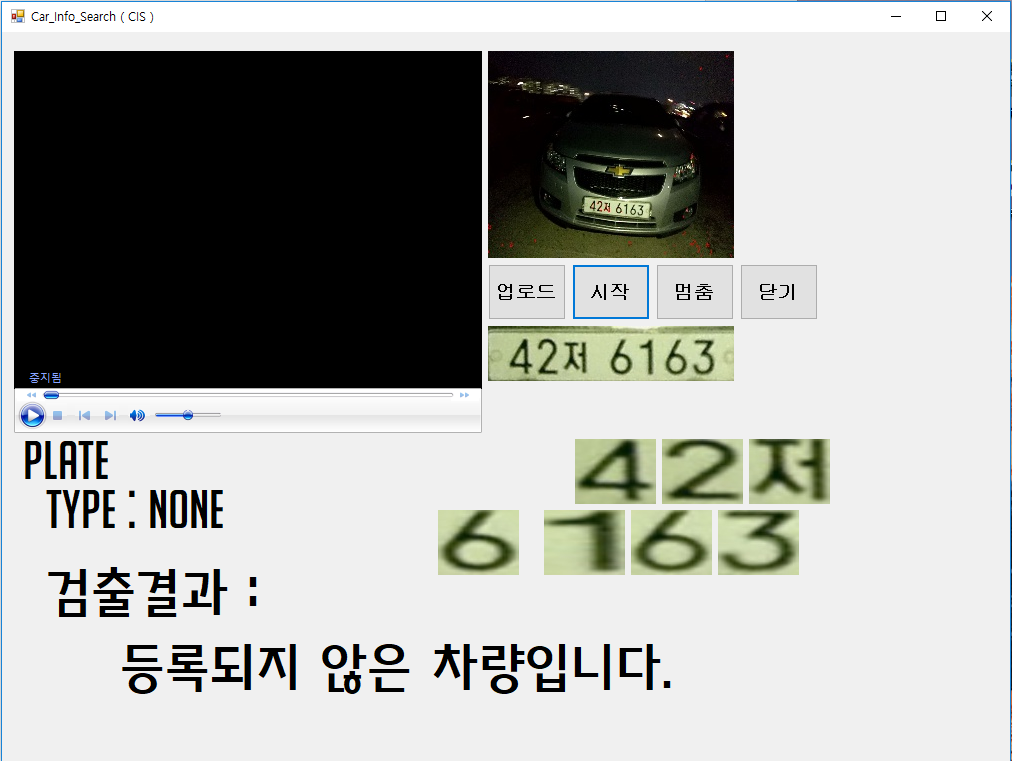
// 허

if (ri == 43) c = (char)(ri + 54797); // 호

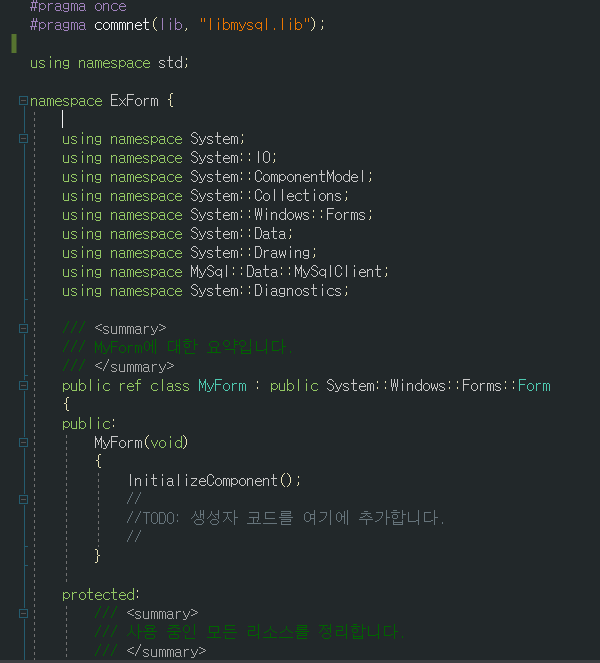
# ri에 레이블 별로 해당 문자를 인식하면 ri 값에 아스키 코드 값을 더하여 레이블 별 인식한 문자를 출력할 수 있게 한다.

**5. 조회 프로그램**

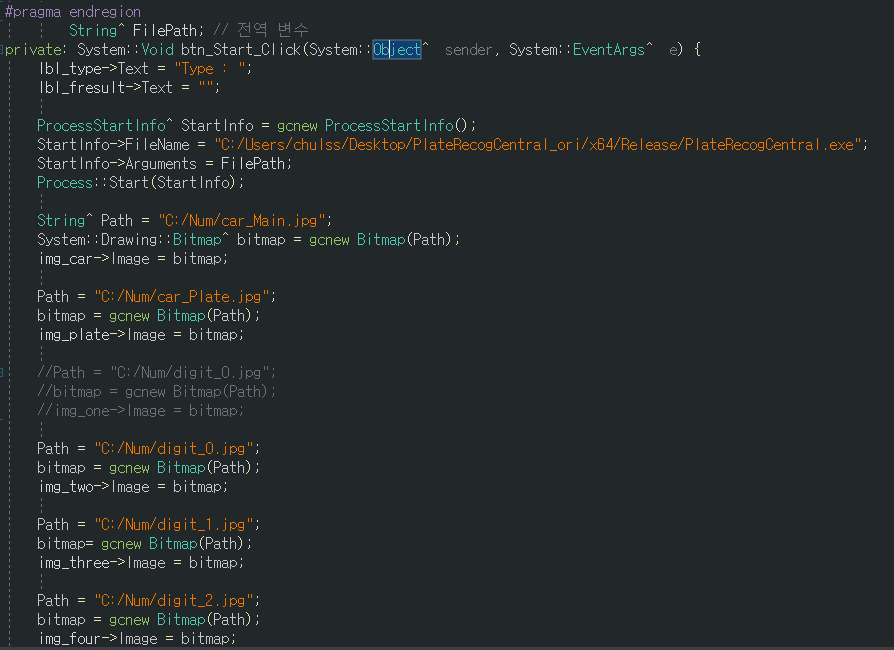
**[ 프로그램 시연 사진 ]**

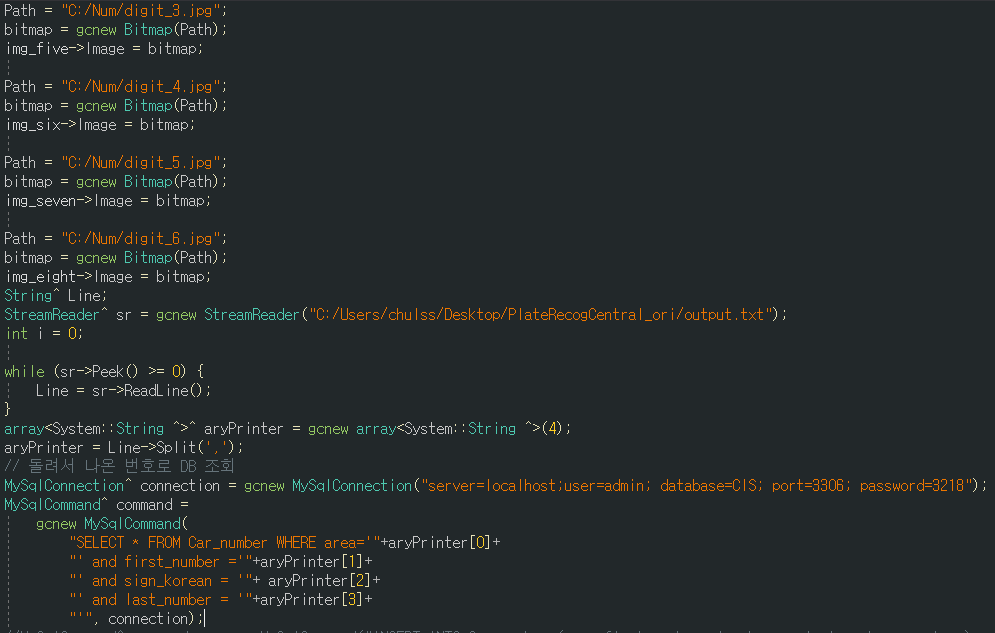


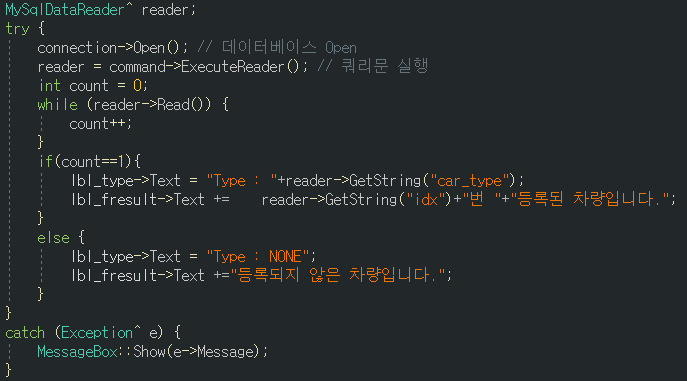
**[ 프로그램 소스 요약 사진 ]**

****

* **사용된 namespace**

****

****

****

* **btn\_Start 버튼**

버튼 클릭 시, 업로드 버튼을 눌러 설정된 매개변수로 검출/인식 프로세스 실행

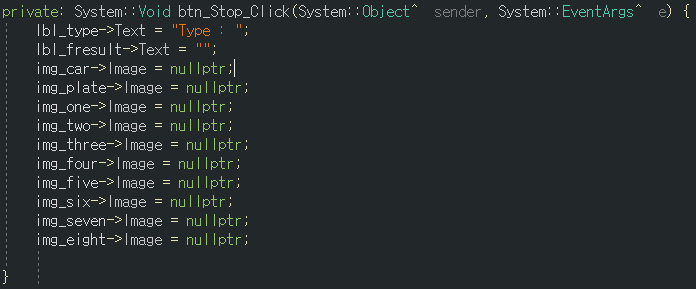
실행 되면서 C:\ 밑 Num 폴더에 폼에 쓰일 이미지를 저장, 인식된 번호를 ‘,’를 붙여 txt파일로 저장

우선 저장된 이미지들을 form image컴포넌트에 설정

번호 txt파일을 ‘,’로 다시 잘라서 문자열 배열에 넣어 줌

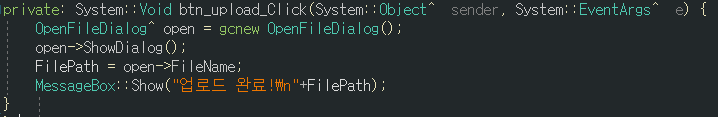
DB에 연동 한 후 쿼리문을 사용하여 조회

데이터베이스가 있을 때 까지 반복하며 내용 있으면 count 변수 증가, 증가되었으면 타입과 검출결과를 나타내줌

****

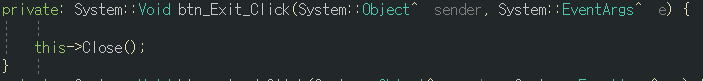
* **btn\_Stop 버튼**

모든 컴포넌트 다시 초기화

****

* **btn\_upload 버튼**

프로세스를 실행할 때 사용할 매개변수 설정

****

* **btn\_Exit 버튼**

종료

**프로그램 전체 소스 텍스트**

#pragma once

#pragma commnet(lib, "libmysql.lib");

//#include <opencv2/core/core.hpp>

//#include <opencv2/highgui/highgui.hpp>

//#include <iostream>

using namespace std;

//using namespace cv;

namespace ExForm {

//using namespace cv;

using namespace System;

using namespace System::IO;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace MySql::Data::MySqlClient;

using namespace System::Diagnostics;

/// <summary>

/// MyForm에 대한 요약입니다.

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

//

//TODO: 생성자 코드를 여기에 추가합니다.

//

}

protected:

/// <summary>

/// 사용 중인 모든 리소스를 정리합니다.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: AxWMPLib::AxWindowsMediaPlayer^ vdo\_Main;

protected:

private: System::Windows::Forms::Button^ btn\_Start;

private: System::Windows::Forms::Button^ btn\_Stop;

private: System::Windows::Forms::Button^ btn\_Exit;

private: System::Windows::Forms::PictureBox^ img\_plate;

private: System::Windows::Forms::PictureBox^ img\_one;

private: System::Windows::Forms::PictureBox^ img\_two;

private: System::Windows::Forms::PictureBox^ img\_three;

private: System::Windows::Forms::PictureBox^ img\_six;

private: System::Windows::Forms::PictureBox^ img\_seven;

private: System::Windows::Forms::PictureBox^ img\_eight;

private: System::Windows::Forms::PictureBox^ img\_nine;

private: System::Windows::Forms::Label^ lbl\_plate;

private: System::Windows::Forms::Label^ lbl\_type;

private: System::Windows::Forms::PictureBox^ img\_car;

private: System::Windows::Forms::Label^ lbl\_result;

private: System::Windows::Forms::PictureBox^ img\_four;

private: System::Windows::Forms::PictureBox^ img\_five;

private: System::Windows::Forms::Label^ lbl\_fresult;

private: System::Windows::Forms::Button^ btn\_upload;

private: System::ComponentModel::IContainer^ components;

protected:

private:

/// <summary>

/// 필수 디자이너 변수입니다.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// 디자이너 지원에 필요한 메서드입니다.

/// 이 메서드의 내용을 코드 편집기로 수정하지 마세요.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

this->vdo\_Main = (gcnew AxWMPLib::AxWindowsMediaPlayer());

this->btn\_Start = (gcnew System::Windows::Forms::Button());

this->btn\_Stop = (gcnew System::Windows::Forms::Button());

this->btn\_Exit = (gcnew System::Windows::Forms::Button());

this->img\_plate = (gcnew System::Windows::Forms::PictureBox());

this->img\_one = (gcnew System::Windows::Forms::PictureBox());

this->img\_two = (gcnew System::Windows::Forms::PictureBox());

this->img\_three = (gcnew System::Windows::Forms::PictureBox());

this->img\_six = (gcnew System::Windows::Forms::PictureBox());

this->img\_seven = (gcnew System::Windows::Forms::PictureBox());

this->img\_eight = (gcnew System::Windows::Forms::PictureBox());

this->img\_nine = (gcnew System::Windows::Forms::PictureBox());

this->lbl\_plate = (gcnew System::Windows::Forms::Label());

this->lbl\_type = (gcnew System::Windows::Forms::Label());

this->img\_car = (gcnew System::Windows::Forms::PictureBox());

this->lbl\_result = (gcnew System::Windows::Forms::Label());

this->img\_four = (gcnew System::Windows::Forms::PictureBox());

this->img\_five = (gcnew System::Windows::Forms::PictureBox());

this->lbl\_fresult = (gcnew System::Windows::Forms::Label());

this->btn\_upload = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->vdo\_Main))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_plate))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_one))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_two))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_three))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_six))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_seven))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_eight))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_nine))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_car))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_four))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_five))->BeginInit();

this->SuspendLayout();

//

// vdo\_Main

//

this->vdo\_Main->Enabled = true;

this->vdo\_Main->Location = System::Drawing::Point(12, 19);

this->vdo\_Main->Name = L"vdo\_Main";

this->vdo\_Main->OcxState = (cli::safe\_cast<System::Windows::Forms::AxHost::State^>(resources->GetObject(L"vdo\_Main.OcxState")));

this->vdo\_Main->Size = System::Drawing::Size(468, 382);

this->vdo\_Main->TabIndex = 0;

//

// btn\_Start

//

this->btn\_Start->Font = (gcnew System::Drawing::Font(L"굴림", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(129)));

this->btn\_Start->Location = System::Drawing::Point(570, 232);

this->btn\_Start->Name = L"btn\_Start";

this->btn\_Start->Size = System::Drawing::Size(78, 56);

this->btn\_Start->TabIndex = 1;

this->btn\_Start->Text = L"시작";

this->btn\_Start->UseVisualStyleBackColor = true;

this->btn\_Start->Click += gcnew System::EventHandler(this, &MyForm::btn\_Start\_Click);

//

// btn\_Stop

//

this->btn\_Stop->Font = (gcnew System::Drawing::Font(L"굴림", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(129)));

this->btn\_Stop->Location = System::Drawing::Point(654, 232);

this->btn\_Stop->Name = L"btn\_Stop";

this->btn\_Stop->Size = System::Drawing::Size(78, 56);

this->btn\_Stop->TabIndex = 2;

this->btn\_Stop->Text = L"멈춤";

this->btn\_Stop->UseVisualStyleBackColor = true;

this->btn\_Stop->Click += gcnew System::EventHandler(this, &MyForm::btn\_Stop\_Click);

//

// btn\_Exit

//

this->btn\_Exit->Font = (gcnew System::Drawing::Font(L"굴림", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(129)));

this->btn\_Exit->Location = System::Drawing::Point(738, 232);

this->btn\_Exit->Name = L"btn\_Exit";

this->btn\_Exit->Size = System::Drawing::Size(78, 56);

this->btn\_Exit->TabIndex = 3;

this->btn\_Exit->Text = L"닫기";

this->btn\_Exit->UseVisualStyleBackColor = true;

this->btn\_Exit->Click += gcnew System::EventHandler(this, &MyForm::btn\_Exit\_Click);

//

// img\_plate

//

this->img\_plate->Location = System::Drawing::Point(486, 294);

this->img\_plate->Name = L"img\_plate";

this->img\_plate->Size = System::Drawing::Size(246, 107);

this->img\_plate->TabIndex = 4;

this->img\_plate->TabStop = false;

//

// img\_one

//

this->img\_one->Location = System::Drawing::Point(486, 407);

this->img\_one->Name = L"img\_one";

this->img\_one->Size = System::Drawing::Size(81, 65);

this->img\_one->TabIndex = 5;

this->img\_one->TabStop = false;

//

// img\_two

//

this->img\_two->Location = System::Drawing::Point(573, 407);

this->img\_two->Name = L"img\_two";

this->img\_two->Size = System::Drawing::Size(81, 65);

this->img\_two->TabIndex = 6;

this->img\_two->TabStop = false;

//

// img\_three

//

this->img\_three->Location = System::Drawing::Point(660, 407);

this->img\_three->Name = L"img\_three";

this->img\_three->Size = System::Drawing::Size(81, 65);

this->img\_three->TabIndex = 7;

this->img\_three->TabStop = false;

//

// img\_six

//

this->img\_six->Location = System::Drawing::Point(542, 478);

this->img\_six->Name = L"img\_six";

this->img\_six->Size = System::Drawing::Size(81, 65);

this->img\_six->TabIndex = 8;

this->img\_six->TabStop = false;

//

// img\_seven

//

this->img\_seven->Location = System::Drawing::Point(629, 478);

this->img\_seven->Name = L"img\_seven";

this->img\_seven->Size = System::Drawing::Size(81, 65);

this->img\_seven->TabIndex = 9;

this->img\_seven->TabStop = false;

//

// img\_eight

//

this->img\_eight->Location = System::Drawing::Point(716, 478);

this->img\_eight->Name = L"img\_eight";

this->img\_eight->Size = System::Drawing::Size(81, 65);

this->img\_eight->TabIndex = 10;

this->img\_eight->TabStop = false;

//

// img\_nine

//

this->img\_nine->Location = System::Drawing::Point(803, 478);

this->img\_nine->Name = L"img\_nine";

this->img\_nine->Size = System::Drawing::Size(81, 65);

this->img\_nine->TabIndex = 11;

this->img\_nine->TabStop = false;

//

// lbl\_plate

//

this->lbl\_plate->AutoSize = true;

this->lbl\_plate->Font = (gcnew System::Drawing::Font(L"Koverwatch", 39.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(129)));

this->lbl\_plate->Location = System::Drawing::Point(12, 404);

this->lbl\_plate->Name = L"lbl\_plate";

this->lbl\_plate->Size = System::Drawing::Size(108, 49);

this->lbl\_plate->TabIndex = 12;

this->lbl\_plate->Text = L"Plate";

//

// lbl\_type

//

this->lbl\_type->AutoSize = true;

this->lbl\_type->Font = (gcnew System::Drawing::Font(L"Koverwatch", 39.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(129)));

this->lbl\_type->Location = System::Drawing::Point(35, 453);

this->lbl\_type->Name = L"lbl\_type";

this->lbl\_type->Size = System::Drawing::Size(124, 49);

this->lbl\_type->TabIndex = 13;

this->lbl\_type->Text = L"Type : ";

//

// img\_car

//

this->img\_car->Location = System::Drawing::Point(486, 19);

this->img\_car->Name = L"img\_car";

this->img\_car->Size = System::Drawing::Size(246, 207);

this->img\_car->TabIndex = 14;

this->img\_car->TabStop = false;

//

// lbl\_result

//

this->lbl\_result->AutoSize = true;

this->lbl\_result->Font = (gcnew System::Drawing::Font(L"KBIZ한마음고딕 B", 39.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->lbl\_result->Location = System::Drawing::Point(31, 519);

this->lbl\_result->Name = L"lbl\_result";

this->lbl\_result->Size = System::Drawing::Size(264, 75);

this->lbl\_result->TabIndex = 15;

this->lbl\_result->Text = L"검출결과 : ";

//

// img\_four

//

this->img\_four->Location = System::Drawing::Point(747, 407);

this->img\_four->Name = L"img\_four";

this->img\_four->Size = System::Drawing::Size(81, 65);

this->img\_four->TabIndex = 16;

this->img\_four->TabStop = false;

//

// img\_five

//

this->img\_five->Location = System::Drawing::Point(436, 478);

this->img\_five->Name = L"img\_five";

this->img\_five->Size = System::Drawing::Size(81, 65);

this->img\_five->TabIndex = 17;

this->img\_five->TabStop = false;

//

// lbl\_fresult

//

this->lbl\_fresult->AutoSize = true;

this->lbl\_fresult->Font = (gcnew System::Drawing::Font(L"KBIZ한마음고딕 B", 39.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->lbl\_fresult->Location = System::Drawing::Point(105, 594);

this->lbl\_fresult->Name = L"lbl\_fresult";

this->lbl\_fresult->Size = System::Drawing::Size(0, 75);

this->lbl\_fresult->TabIndex = 18;

//

// btn\_upload

//

this->btn\_upload->Font = (gcnew System::Drawing::Font(L"굴림", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(129)));

this->btn\_upload->Location = System::Drawing::Point(486, 232);

this->btn\_upload->Name = L"btn\_upload";

this->btn\_upload->Size = System::Drawing::Size(78, 56);

this->btn\_upload->TabIndex = 19;

this->btn\_upload->Text = L"업로드";

this->btn\_upload->UseVisualStyleBackColor = true;

this->btn\_upload->Click += gcnew System::EventHandler(this, &MyForm::btn\_upload\_Click);

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(7, 12);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(1008, 729);

this->Controls->Add(this->btn\_upload);

this->Controls->Add(this->lbl\_fresult);

this->Controls->Add(this->img\_five);

this->Controls->Add(this->img\_four);

this->Controls->Add(this->lbl\_result);

this->Controls->Add(this->img\_car);

this->Controls->Add(this->lbl\_type);

this->Controls->Add(this->lbl\_plate);

this->Controls->Add(this->img\_nine);

this->Controls->Add(this->img\_eight);

this->Controls->Add(this->img\_seven);

this->Controls->Add(this->img\_six);

this->Controls->Add(this->img\_three);

this->Controls->Add(this->img\_two);

this->Controls->Add(this->img\_one);

this->Controls->Add(this->img\_plate);

this->Controls->Add(this->btn\_Exit);

this->Controls->Add(this->btn\_Stop);

this->Controls->Add(this->btn\_Start);

this->Controls->Add(this->vdo\_Main);

this->Name = L"MyForm";

this->Text = L"Car\_Info\_Search ( CIS )";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->vdo\_Main))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_plate))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_one))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_two))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_three))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_six))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_seven))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_eight))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_nine))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_car))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_four))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->img\_five))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

String^ FilePath = ""; // 전역 변수

private: System::Void btn\_Start\_Click(System::Object^ sender, System::EventArgs^ e) {

img\_car->Image = nullptr;

img\_plate->Image = nullptr;

img\_one->Image = nullptr;

img\_two->Image = nullptr;

img\_three->Image = nullptr;

img\_four->Image = nullptr;

img\_five->Image = nullptr;

img\_six->Image = nullptr;

img\_seven->Image = nullptr;

img\_eight->Image = nullptr;

lbl\_type->Text = "Type : ";

lbl\_fresult->Text = "";

ProcessStartInfo^ StartInfo = gcnew ProcessStartInfo();

StartInfo->FileName = "C:/Users/chulss/Desktop/PlateRecogCentral\_ori/x64/Release/PlateRecogCentral.exe";

StartInfo->Arguments = FilePath;

Process::Start(StartInfo);

System::Threading::Thread::Sleep(10000);

String^ Path = "C:/Num/car\_Main.jpg";

System::Drawing::Bitmap^ bitmap = gcnew Bitmap(Path);

img\_car->Image = bitmap;

Path = "C:/Num/car\_Plate.jpg";

bitmap = gcnew Bitmap(Path);

img\_plate->Image = bitmap;

//Path = "C:/Num/digit\_0.jpg";

//bitmap = gcnew Bitmap(Path);

//img\_one->Image = bitmap;

Path = "C:/Num/digit\_0.jpg";

bitmap = gcnew Bitmap(Path);

img\_two->Image = bitmap;

Path = "C:/Num/digit\_1.jpg";

bitmap = gcnew Bitmap(Path);

img\_three->Image = bitmap;

Path = "C:/Num/digit\_2.jpg";

bitmap = gcnew Bitmap(Path);

img\_four->Image = bitmap;

Path = "C:/Num/digit\_3.jpg";

bitmap = gcnew Bitmap(Path);

img\_five->Image = bitmap;

Path = "C:/Num/digit\_4.jpg";

bitmap = gcnew Bitmap(Path);

img\_six->Image = bitmap;

Path = "C:/Num/digit\_5.jpg";

bitmap = gcnew Bitmap(Path);

img\_seven->Image = bitmap;

Path = "C:/Num/digit\_6.jpg";

bitmap = gcnew Bitmap(Path);

img\_eight->Image = bitmap;

String^ Line;

StreamReader^ sr = gcnew StreamReader("C:/Users/chulss/Desktop/PlateRecogCentral\_ori/output.txt");

int i = 0;

while (sr->Peek() >= 0) {

Line = sr->ReadLine();

}

array<System::String ^>^ aryPrinter = gcnew array<System::String ^>(4);

aryPrinter = Line->Split(',');

// 돌려서 나온 번호로 DB 조회

MySqlConnection^ connection = gcnew MySqlConnection("server=localhost;user=admin; database=CIS; port=3306; password=3218");

MySqlCommand^ command = gcnew MySqlCommand("SELECT \* FROM car\_number WHERE area = '" + aryPrinter[0] + "' and first\_number = '" + aryPrinter[1] + "' and sign\_korean = '" + aryPrinter[2] + "' and last\_number = '" + aryPrinter[3] + "'", connection);

//MySqlCommand^ command = gcnew MySqlCommand("SELECT \* FROM car\_number WHERE area = '" + Line + "'", connection);

//MySqlCommand^ command = gcnew MySqlCommand("INSERT INTO Car\_number (area,first\_number,sign\_korean,last\_number,car\_type) values ('22','na','9929','1')");

MySqlDataReader^ reader;

try {

connection->Open(); // 데이터베이스 Open

reader = command->ExecuteReader(); // 쿼리문 실행

int count = 0;

//lbl\_fresult->Text += aryPrinter[0] + " " + aryPrinter[1] + " " + aryPrinter[2] + " " + aryPrinter[3] + "\n";

while (reader->Read()) {

count++;

}

if (count == 1) {

lbl\_type->Text = "Type : " + reader->GetString("car\_type");

lbl\_fresult->Text += reader->GetString("idx") + "번 " + "등록된 차량입니다.";

}

else {

lbl\_type->Text = "Type : NONE";

lbl\_fresult->Text += "등록되지 않은 차량입니다.";

}

connection->Close();

}

catch (Exception^ e) {

MessageBox::Show(e->Message);

connection->Close();

}

}

private: System::Void btn\_Stop\_Click(System::Object^ sender, System::EventArgs^ e) {

lbl\_type->Text = "Type : ";

lbl\_fresult->Text = "";

img\_car->Image = nullptr;

img\_plate->Image = nullptr;

img\_one->Image = nullptr;

img\_two->Image = nullptr;

img\_three->Image = nullptr;

img\_four->Image = nullptr;

img\_five->Image = nullptr;

img\_six->Image = nullptr;

img\_seven->Image = nullptr;

img\_eight->Image = nullptr;

}

private: System::Void btn\_Exit\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show("종료합니다.\n");

this->Close();

}

private: System::Void btn\_upload\_Click(System::Object^ sender, System::EventArgs^ e) {

OpenFileDialog^ open = gcnew OpenFileDialog();

open->ShowDialog();

FilePath = open->FileName;

vdo\_Main->URL = FilePath;

vdo\_Main->Ctlcontrols->play();

/\*

cv::VideoCapture cap(0); // 캠으로부터 영상을 받아온다 (혹은 1번)

cv::Mat img; // 동영상 파일 재생을 위한 행렬 선언

char savefile[200]; // 이미지 파일 이름을 200자 이내로 제한하기 위한 char 변수 선언

for (int i = 0; ; )

{

cv::cap >> img; // 동영상 파일을 img에 저장한다.

cv::imshow("image", img); // 이미지를 출력한다.

sprintf(savefile, "image%d.jpg", i++);

cv::imwrite(savefile, img); // img를 파일로 저장한다.

cv::waitKey(100); // 100ms의 딜레이를 발생시킨다. 즉, 1초에 10장의 이미지를 저장하게 된다.

}\*/

MessageBox::Show("업로드 완료!\n" + FilePath);

}

};

}

**6. 문제점**

1. 번호판의 종류와 환경적 요인의 영향을 많이 받아 인식률이 저조합다.

2. 영상을 불러와서 캡쳐를 하지못하고 캡쳐된 사진으로만 프로그램이 돌아갑니다.

**7.소감**

팀원 각자 맡은 분야에 대해서는 물론, 자신의 분야가 아닌 부분도 서로 도와가면서 했기 때문에 함께 모인 시간이 많으며, 모여 밤 새 고민도 해보고 차주들에게 좋지않은 소리도 많이 들으며 데이터수집도 하고, 문제를 해결하며 성취감도 느낀 좋은 기회였습니다. 끝으로 각자 맡은 분야는 다르지만 팀으로 활동한 시간이 많아 서로 고생의 정도는 같다고 생각합니다.