「 OpenCV를 활용한 컴퓨터 비전 」

제작 신승호

한국 폴리텍대학교 성남 캠퍼스
하이테크 과정 AI자동화학과

< 목차 >

프로젝트 결과 보고서

1 프로젝트명

OpenCV를 활용한 컴퓨터 비전

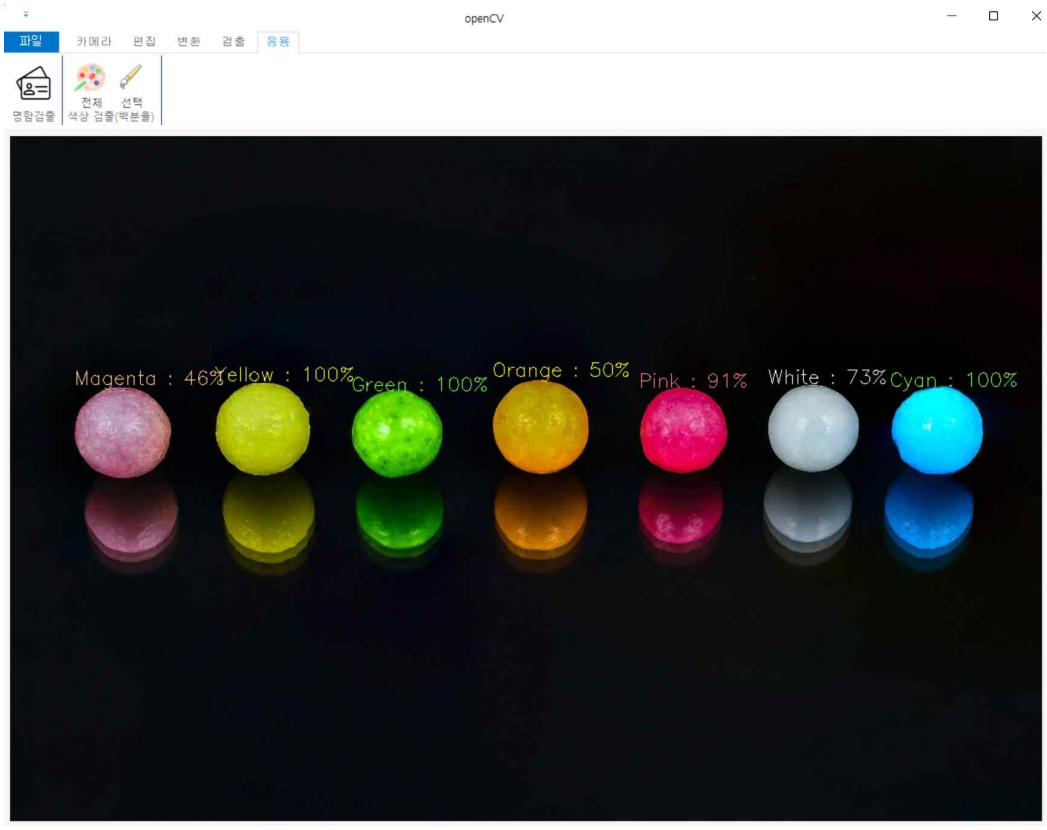
2 프로젝트 개요

2-1 프로젝트 기획 배경

- C# 프로그램 언어 학습
- PC를 통한 Cognex, 웹캠 제어
- 영상, 이미지 편집, 변환, 검출 등 이미지 처리 이해
- 물체 및 패턴 검출 등을 통한 인식 및 정보 추출 시스템의 이해
- 이미지 처리 알고리즘을 이해하여 산업 현장에서 활용

2-2 프로젝트 소개

C#과 OpenCV를 사용해 실시간으로 이미지를 캡처하여 편집, 변환, 검출하고 자동화 현장에서 제품 측정, 검출, 분류 등으로 활용.



2-3 시스템 구성

종류	모델명	용도	이미지
Cognex Vision	n-Sight 2001C-353	실시간 영상, 이미지 추출	
Webcam	USB Camera	실시간 영상, 이미지 추출	
PC 프로그램	Visual Studio C#	추출된 이미지 처리	

2-4 구현 범위

파일	카메라	편집	변환	검출	응용
영상 및 이미지 불러오기, 저장하기	Webcam, Cognex을 사용한 영상 및 이미지 추출	확대축소, 좌우반전, 자르기, 회전 등 기본 이미지 조작	색상, 모폴로지, 블러 효과 등 이미지 변환 알고리즘	코너, 가장자리, 글자, 모형, 색상 검출 등 특징 검출 및 탐지 알고리즘	명함, 색상 검출, 바코드, QR코드, 동전찾기 등의 특정 작업 수행

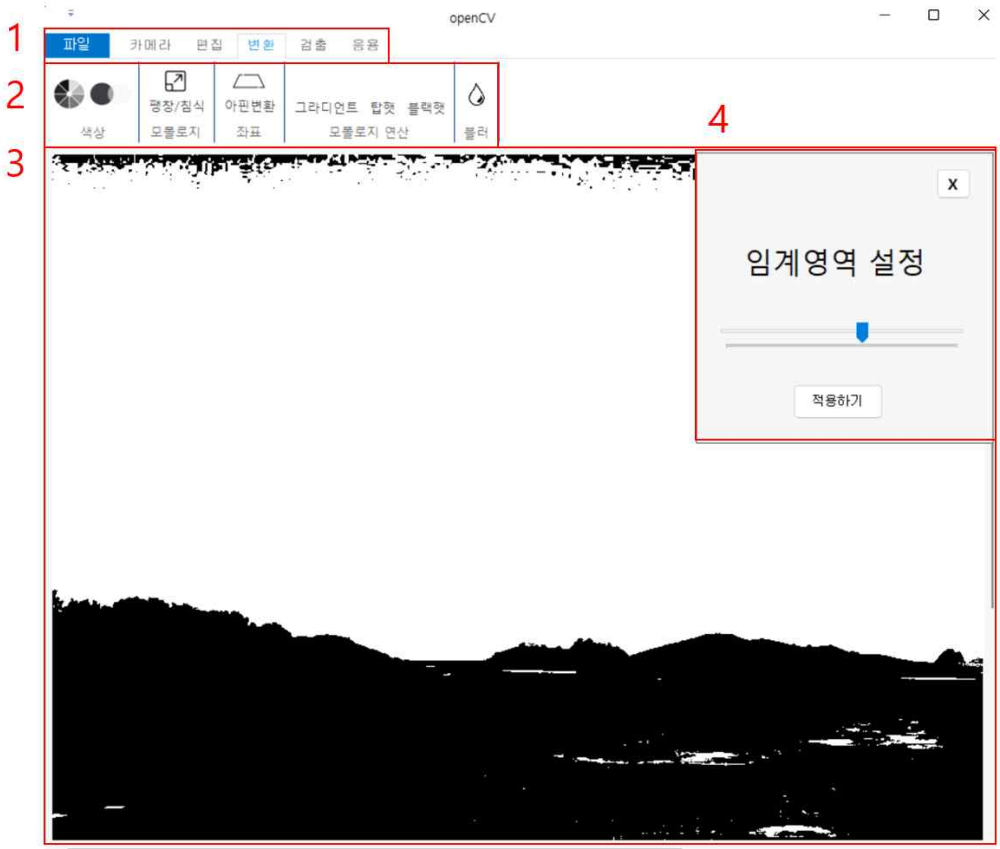
2-5 프로젝트 수행 일정

구분	일정	내용	진행상황
아이디어 구상	2024.8.19	Cognex와 Webcam을 이용한 비전 프로그램 작성	완료
프로그래밍	8.20~8.25	- 전체 디자인 구현 - 파일 저장 및 불러오기 - Cognex, Webcam 연결 - 이미지 기본 조작 알고리즘 구현	완료
	8.26~8.31	- 이미지 변환 알고리즘 구현 - 코너, 가장자리 등 특징 검출 알고리즘 구현 - 글자 판독 및 원형 탐지, 색상 검출 알고리즘 구현	완료
	9.1~9.7	- 명함 글자 검출 프로그램 구현 - 원형 물체 색상 검출 프로그램 구현	완료
	9.8~9.13	- 동전 분류, 바코드 해석 프로그램 구현 - 얼굴 인식 후 데코레이션 프로그램 구현	완료

3 프로젝트 내용

3-1. User Interface

1. 화면구성



순서	내용
1	메뉴 탭
2	작업 창
3	영상 및 이미지 출력 창
4	네비게이션 바

2. 메뉴 구성

이름	내용
파일	영상 및 이미지 열기, 저장
카메라	Cognex, Webcam 연결 및 영상, 이미지 추출
편집	확대/축소, 상하/좌우 반전, 크기조절, 자르기, 회전
변환	그레이스케일, 이진화, 아핀 변환, 모폴로지 팽창/침식, 모폴로지 연산(그라디언트, 탑햇, 블랙햇), 블러(단순 블러, 가우시안 블러)
검출	코너, 가장자리 검출, 글자 판독, 원 탐지, HSV색상 검출
응용	명함 검출, 색상 검출

3-2. 카메라 출력

일반적인 Webcam과 산업용으로 사용되는 Cognex 카메라를 필요에 따라 선택해서 사용

Webcam	Cognex

VideoCapture video;//카메라 출력
Mat camframe;
 	
video = new VideoCapture(0);//첫카메라
camframe = new Mat();
 	
video.Read(camframe);
pictureBox1.Image = camframe.ToBitmap();
- 장치 번호 0번 카메라에서 이미지를 불러와 프레임을 재생
- 카메라에서 실시간으로 프레임을 읽어 Mat 클래스 형식으로 출력
- Mat이미지를 비트맵으로 변환하여 PictureBox1에 이미지 출력
	'using Cognex.InSight;
	using Cognex.InSight.Controls.Display;
	- Cognex사에서 제공하는 라이브러리를 사용하여 이미지를 불러와 프레임을 재생
	
	cvsInSightDisplay1.Enabled = false;
	cvsInSightDisplay1.Visible = false;
	
	cvsInSightDisplay1.InSight.SoftOnline = false;
	cvsInSightDisplay1.InSight.LiveAcquisition = false;
	- 초기값을 False로 설정, 이벤트 발생 전까지 Cognex 카메라 미출력

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

using OpenCvSharp;
using OpenCvSharp.Extensions;
```

- 공통적으로 OpenCV 라이브러리를 사용하여 응용 프로그램을 생성

```
Mat image;
OpenCV_CLASS convert;
OpenCV_filter covertfilter;
OpenCV_Detection detect;
OpenCV_action openCV_Action;
```

- Mat 클래스 형식으로 이미지를 이용.
- 이미지 기본 조작, 필터, 검출, 응용 클래스를 생성하여 사용

```
private void Form1_Load(object sender, EventArgs e)
{
    image = new Mat(640, 480, MatType.CV_8UC3);//초기 이미지 비율
    convert = new OpenCV_CLASS();//클래스 생성
    covertfilter = new OpenCV_filter();
    detect = new OpenCV_Detection();
    openCV_Action = new OpenCV_action();
```

- Mat image의 초기값과 사용할 알고리즘 클래스 생성

1. Webcam 출력


```
//카메라출력
참조 1개
private void camBTN0_Click(object sender, EventArgs e)
{
    stopMotion();
    pictureBox1.Enabled = true;
    pictureBox1.Visible = true;
    try
    {
        camTimer.Enabled = true;
    }
    catch (Exception ex) { }
}
```

- 사용할 PictureBox를 보이게 함
- 이벤트 발생시 Timer를 사용하여 카메라에서 이미지를 받아와 PictureBox에 출력

```
camTimer.Enabled = false;//카메라 타이머 이벤트 초기화
private void camTimer_Tick(object sentrier, EventArgs e)
{
    try
    {
        video = new VideoCapture(0);//첫카메라
        camframe = new Mat();

        video.Read(camframe);
        pictureBox1.Image = camframe.ToBitmap();
    }
    catch (Exception ex) { }
}
```

(ApplicationSettings)	
(Name)	**camTimer**
Enabled	False
GenerateMember	True
Interval	**33**
Modifiers	Private
Tag	

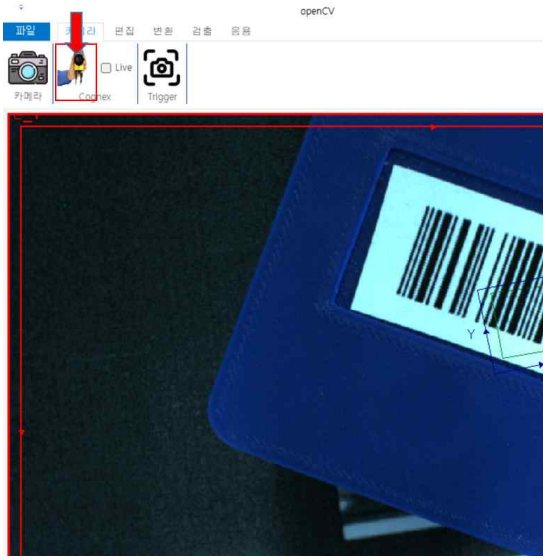
- interval을 33으로 설정 후 Tick 이벤트 마다 영상 및 이미지 출력 창인 PictureBox1에 이미지 삽입
  (fps = 1000/interval이다.
  즉 interval 33은 30fps이다.)

```
private void stopMotion()
{
    if(video != null) video.Dispose();
    videoTimer.Enabled = false;
    if(camframe != null) camframe.Dispose();
    camTimer.Enabled = false;

    pictureBox1.Visible = false;
    pictureBox1.Enabled = false;

    IsConnected1 = false;
    cvsInSightDisplay1.Enabled = false;
    cvsInSightDisplay1.Visible = false;
}
```

- Cognex와 Webcam이 동시 출력되지 않게 종료 후 출력

2. Cognex 출력

1) Cognex 카메라 영상과 메모리에 내장된 검출 프로그램 실행 결과 출력




```
//Cognex카메라
CvsInSight insight = new CvsInSight();
bool IsConnected1 = false;
bool OnLineST1 = false;
```

- Cognex에서 제공하는 라이브러리를 사용하기 위해 클래스 생성
- Load 시 초기값
  : 연결확인 flag와 온라인상태 flag의 초기값을 false로 해줌

```
private void cognexBTN_Click(object sender, EventArgs e)
{
    stopMotion();
    cvsInSightDisplay1.Enabled = true;
    cvsInSightDisplay1.Visible = true;
    try
    {
        // 카메라가 연결되지 않은 상태일 때
        if (!(IsConnected1))
        {
            cvsInSightDisplay1.Connect("172.31.6.9", "admin", "", false);
            IsConnected1 = true;

            cvsInSightDisplay1.ImageScale = 0.84; // 촬영중인 이미지의 배율 설정
            cvsInSightDisplay1.ShowImage = true; // 카메라가 취득한 이미지를 보여줌
            cvsInSightDisplay1.ShowGraphics = true;

            Online_Check();
        }
        else // 카메라가 연결된 상태일 때
        {
            insight.Disconnect(); // 연결된 카메라와의 접속을 끊음
            IsConnected1 = false;
            cvsInSightDisplay1.ShowImage = false; // 카메라가 취득한 이미지를 가림
            cvsInSightDisplay1.ShowGraphics = false;
        }
    }
    catch { }
}
```

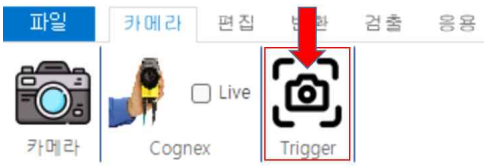
- Cognex와 Webcam이 동시 출력되지 않게 종료 후 출력
- 카메라 이미지를 보이기 위한 화면 출력(cvsInSightDisplay1)
- 카메라가 연결된 상태일 경우 종료

2) Live Mode

- 실시간 이미지 출력
- Online 상태일시 Live가 불가능. Online 상태를 종료 후 Live 시작

```
private void CognexLiveCB_CheckBoxCheckChanged(object sender, EventArgs e)
{
    if (cvsInSightDisplay1.InSight.SoftOnline)
    {
        cvsInSightDisplay1.InSight.SoftOnline = !cvsInSightDisplay1.InSight.SoftOnline;
    }
    cvsInSightDisplay1.InSight.LiveAcquisition = !cvsInSightDisplay1.InSight.LiveAcquisition;
    Online_Check();
}
```

3) Trigger



- Online 상태에선 Triger 비활성화
- Live 종료, 현재 이미지 저장 후 PictureBox에 출력

```
private void Online_Check()
{
    // OnLineST1 에 카메라의 온라인 상태 여부를 할당
    OnLineST1 = cvsInSightDisplay1.SoftOnline;

    // 카메라가 온라인일 때
    if (OnLineST1 == true)
    {
        trigerBTN.Enabled = false; //오프라인 일 때 트리거 비활성화
    }
    // 카메라가 오프라인일 때
    else if (OnLineST1 == false)
    {
        trigerBTN.Enabled = true; //오프라인 일 때 트리거 활성화
    }
}
```

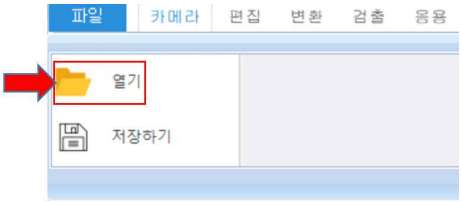
```
private void trigerBTN_Click(object sender, EventArgs e)
{
    if (IsConnected1)
    {
        if (cvsInSightDisplay1.InSight.LiveAcquisition)
        {
            CognexLiveCB.Checked = false;
            cvsInSightDisplay1.InSight.LiveAcquisition = !cvsInSightDisplay1.InSight.LiveAcquisition;
        }
        cvsInSightDisplay1.InSight.ManualAcquire(wait: true);
        pictureBox1.Enabled = true;
        pictureBox1.Visible = true;

        Bitmap bin = cvsInSightDisplay1.GetBitmap();
        image = BitmapConverter.ToMat(bin);
        pictureBox1.Image = cvsInSightDisplay1.GetBitmap();

        cvsInSightDisplay1.Enabled = false;
        cvsInSightDisplay1.Visible = false;
    }
    if (camTimer.Enabled)
    {
        pictureBox1.Visible=true;
        pictureBox1.Enabled=true;
        Cv2.CopyTo(camframe, image);
        pictureBox1.Image = image.ToBitmap();
    }
    camTimer.Enabled = false;
    //메모리 해제
    if (cvsInSightDisplay1 != null) cvsInSightDisplay1.Dispose();
    if (camframe != null) camframe.Dispose();
```

- Webcam도 동일, 현재 이미지 저장 후 PictureBox에 출력
- 카메라 메모리 해제

3-3. 영상 및 이미지 불러오기

기존 저장된 영상 및 이미지를 불러오거나, 현재 이미지를 저장하여 보관

1. 영상 및 이미지 열기



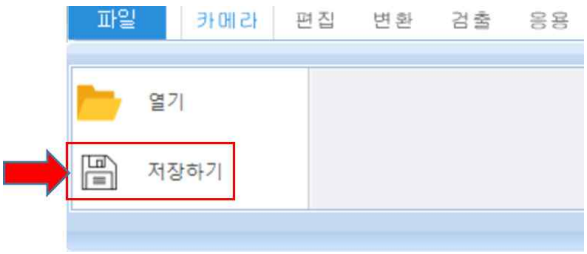

```
private void openfile_Click(object sender, EventArgs e)
{
    try
    {
        stopMotion();
        pictureBox1.Enabled = true;
        pictureBox1.Visible = true;
        OpenFileDialog dlg = new OpenFileDialog();
        if (dlg.ShowDialog() == DialogResult.OK)
        {

            if (dlg.FileName.EndsWith(".mp4") || dlg.FileName.EndsWith(".avi"))
            {
                video = new VideoCapture(dlg.FileName);
                camframe = new Mat();
                videoTimer.Enabled = true;

            }
            else
            {
                image = Cv2.ImRead(dlg.FileName);
                pictureBox1.Size = new System.Drawing.Size(image.Width, image.Height);
                pictureBox1.Image = OpenCvSharp.Extensions.BitmapConverter.ToBitmap(image);
            }

        }
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
}
```

- Cognex와 Webcam이 출력되지 않게 종료
- 이미지를 PictureBox에 출력
- 영상 파일일 경우 Timer를 사용하여 이미지 출력

```
private void videoTimer_Tick(object sender, EventArgs e)
{
    if (video.IsOpened())
    {
        if (video.Read(camframe) && !camframe.Empty())
        {
            pictureBox1.Image?.Dispose(); // Dispose of previous image
            pictureBox1.Image = BitmapConverter.ToBitmap(camframe);
        }
        else
        {
            stopMotion();
        }
    }
}
```

- Interval 33 설정, 30fps로 이전 이미지 삭제 후 현재 이미지 출력

2. 영상 및 이미지 저장하기




```
private void savefileBTN_Click(object sender, EventArgs e)
{
    SaveFileDialog saveFileDialog = new SaveFileDialog
    {
        Filter = "PNG Image|*.png|JPEG Image|*.jpg|Bitmap Image|*.bmp",
        Title = "이미지 저장",
        FileName = "default_image"  // 기본 파일 이름
    };

    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        string filePath = saveFileDialog.FileName;

        // 선택한 경로로 이미지 저장
        SaveImage(image.ToBitmap(), filePath);

        Console.WriteLine("이미지가 저장되었습니다: " + filePath);
    }
    else
    {
        Console.WriteLine("파일 저장이 취소되었습니다.");
    }
}
```

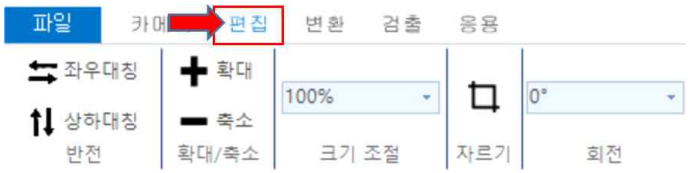
- PNG, JPG, BMP 형식으로 이미지 저장

```
static void SaveImage(Bitmap bitmap, string filePath)
{
    try
    {
        // Bitmap을 파일로 저장 (파일 확장자에 따라 포맷 자동 결정)
        string extension = Path.GetExtension(filePath).ToLower();
        ImageFormat format;

        switch (extension)
        {
            case ".jpg":
                format = ImageFormat.Jpeg;
                break;
            case ".bmp":
                format = ImageFormat.Bmp;
                break;
            default:
                format = ImageFormat.Png;
                break;
        }

        bitmap.Save(filePath, format);
    }
    catch (Exception ex)
    {
        Console.WriteLine("이미지 저장 중 오류 발생: " + ex.Message);
    }
}
```

- 선택한 위치에 선택한 확장자로 파일 저장

3-4. 영상 및 이미지 편집

이미지를 반전, 확대/축소, 자르기, 크기 조절, 회전하여 원하는 형태로 편집



1. 이미지 반전

- Cv2.Flip(원본 이미지, 결과 이미지, 대칭 축)로 색상 공간을 변환

대칭 축 종류

속성	의미
FlipMode.X	X축 대칭 (상하 대칭)
FlipMode.Y	Y축 대칭 (좌우 대칭)
FlipMode.XY	XY축 대칭 (상하좌우 대칭)

좌우대칭 (Y축)	상하대칭 (X축)

public Mat SymmetryY(Mat src)
{
symn = new Mat(src.Size(), MatType.CV_8UC3);
Cv2.Flip(src, symn, FlipMode.Y);
return symn;
}
	public Mat SymmetryX(Mat src)
	{
	symn = new Mat(src.Size(), MatType.CV_8UC3);
	Cv2.Flip(src, symn, FlipMode.X);
	return symn;
	}

2. 확대/축소

- 입력 이미지의 크기를 단계적으로 변화시키는 이미지 피라미드 사용
- Cv2.Pyr*(원본 이미지, 결과 이미지, 결과 이미지 크기, 테두리 외삽법)으로 이미지 크기를 변환
- 테두리 외삽법은 이미지 밖의 픽셀을 외삽하는데 사용되는 테두리 모드
  (확대, 축소할 경우, 영역 밖의 픽셀은 추정해서 값을 할당)
- 결과 이미지 크기는 매개변수에 직접 인수를 할당해서 (업/다운)샘플링을 수행 가능
- 피라미드 방식은 2의 배수로만 확대/축소 가능

확대	축소

public Mat ZoomIn(Mat src)
{
zoomin = new Mat();
Cv2.PyrUp(src, zoomin);
return zoomin;
}
	public Mat ZoomOut(Mat src)
	{
	zoomout = new Mat();
	Cv2.PyrDown(src, zoomout)
	return zoomout;
	}

3. 크기 조절

- Cv2.Resize*(원본 이미지, 결과 이미지, 절대 크기, 상대 크기(X), 상대 크기(Y), 보간법)으로 이미지 크기를 변환
- 절대 크기 또는 상대 크기를 사용해 이미지 크기를 조절하며 절대 크기는 Size 구조체로 크기를 설정
  (절대 크기는 필수, 상대 크기는 선택, 절대 크기 인수를 0으로 설정하면 상대 크기 사용 가능)
- 쌍선형 보간법 사용

크기 조절

```
public Mat ResizeImage(Mat src, int size)
{
    resize = new Mat();
    Cv2.Resize(src, resize, new Size(src.Width* size / 100, src.Height* size / 100));
    return resize;
}
```

- 매개 변수 size를 통해 %로 값을 받았기에 100으로 나누어줌

4. 자르기

- SubMat()는 Range 구조체, Rect 구조체, int 할당 등을 통해 생성
- 특정 영역에 대해서 작업하기 때문에 이미지를 자르는 것을 관심 영역 지정이라 부름

자르기

```
public Mat Cutting(Mat src, int pointX, int pointY, int pointW, int pointH)
{
    cut = src.SubMat(new Rect(pointX,pointY,pointW,pointH));
    return cut;
}
```

- 마우스 이벤트를 활용하여 매개변수를 넘겨받는다.

5. 회전

- Cv2.GetRotationMatrix2D(중심점의 좌표, 회전 각도, 비율)로 회전 행렬을 생성
- 중심점의 좌표를 기준으로 회전 각도만큼 회전하며, 비율만큼 크기를 변경

회전

```
public Mat Rotation(Mat src, int angle)
{
    rotation = new Mat();
    Mat matrix = Cv2.GetRotationMatrix2D(new Point2f(src.Width / 2, src.Height / 2), angle, 1.0);
    Cv2.WarpAffine(src, rotation, matrix, new Size(src.Width, src.Height));
    return rotation;
}
```

3-5. 영상 및 이미지 변환

이미지를 그레이스케일, 이진화 모폴로지 팽창/침식, 연산, 블러 효과를 주어 노이즈 제거 및 원하는 결과값을 얻기 수월한 형태로 변환

1. 그레이스케일

- 영상이나 이미지의 색상을 흑백 색상으로 변환하기 위해서 사용(3채널에서 단일 채널로 변경)
- Cv2.CvtColor(원본 이미지, 결과 이미지, ColorConversionCodes.BGR2GRAY)

그레이 스케일

```
public Mat GrayScale(Mat src)
{
    gray = new Mat();
    //이미 채널이 1이면(그레이스케일이 되어있으면) RGB2GRAY에서 오류가 생김
    if (src.Channels() != 1)
    {
        Cv2.CvtColor(src, gray, ColorConversionCodes.RGB2GRAY);
        return gray;
    }
    else
        return src;
}
```

2. 이진화

- Cv2.Threshold(원본 이미지, 결과 이미지, 임곗값, 최댓값, 임곗값 형식)로 이진화를 적용
- 임곗값 형식에 따라 이진화 방식을 설정
  (임곗값보다 낮은 픽셀값은 0이나 원본 픽셀값, 높은 픽셀값은 최댓값으로 변경)

임곗값 형식 종류

속성	의미
ThresholdTypes.Binary	임곗값을 초과할 경우 최댓값, 아닐 경우 0
ThresholdTypes.BinaryInv	임곗값을 초과할 경우 0, 아닐 경우 최댓값
ThresholdTypes.Mask	검은색 이미지로 변경(마스크용)

이진화

```
public Mat Binary(Mat src, int TH)
{
    bin = new Mat();
    Cv2.Threshold(src, bin, TH, 255, ThresholdTypes.Binary);
    return bin;
}
```

- 임계값을 조절하기 위해 매개변수로 받음

3. 모폴로지 팽창/침식

- Cv2.GetPerspectiveTransform(커널의 형태, 커널의 크기, 중심점)로 구조 요소을 생성
- 커널의 형태 : 직사각형(Rect), 십자가(Cross), 타원(Ellipse)
- 커널의 크기 : 구조 요소의 크기(크기가 작으면 커널의 형태는 영향을 받지 않음)
- 고정점 : 커널의 중심 위치(미설정 시 사용하는 함수 값에 따라 결정)

- 영상이나 이미지의 화소값을 대체하기 위해 사용
- Dilate와 Erode를 이용해 이미지의 정확도를 높일 수 있음
- 팽창(Dilate)은 구조 요소를 사용하여 이웃한 화소를 최대 픽셀로 대체
  (노이즈 제거 후 줄어든 크기를 복구 시 사용)
- 침식(Erode)은 구조 요소를 사용하여 이웃한 화소를 최소 픽셀로 대체
  (노이즈 제거에 주로 사용)
- 응용하여 노이즈 제거, 요소 결합 및 분리, 강도 피크 검출 등에 이용

팽창 연산

```
public Mat Dil(Mat src, int count)
{
    dil = new Mat();
    Mat element = Cv2.GetStructuringElement(MorphShapes.Cross, new Size(5,5));
    Cv2.Dilate(src, dil, element, new Point(-1, -1), count);
    return dil;
}
```

- Cv2.Dilate(원본 배열, 결과 배열, 구조 요소, 고정점, 반복 횟수, 테두리 외삽법, 테두리 색상)

침식 연산

```
public Mat Ero(Mat src, int count)
{
    ero = new Mat();
    Mat element = Cv2.GetStructuringElement(MorphShapes.Cross, new Size(5, 5));
    Cv2.Erode(src, ero, element, new Point(-1, -1), count);
    return ero;
}
```

- Cv2.Erode(원본 배열, 결과 배열, 구조 요소, 고정점, 반복 횟수, 테두리 외삽법, 테두리 색상)

- 커널(구조 요소)의 형태는 십자형
- 고정점을 (-1, -1)로 할당할 경우, 커널의 중심부에 고정점이 위치
- 반복횟수를 조절하기 위해 매개변수로 받음
- 구조 요소란 연산을 실행할 때 행렬의 크기, 형태 등을 의미

4. 모폴로지 연산

- Cv.MorphologyEx(원본, 결과, 임시, 요소, 연산 방법, 반복횟수)
  (임시는 크기가 동일한 이미지를 입력. 연산 중 이미지를 잠시동안 교체하기 위한 용도)
- 팽창(Dilate)과 침식(Erode)을 이용하여 고급 형태학을 적용 가능

- `MorphologyOperation.*`
  - `MorphologyOperation.Open` : 열기 연산
  - `MorphologyOperation.Close` : 닫기 연산
  - `MorphologyOperation.Gradient` : 그라디언트 연산
  - `MorphologyOperation.TopHat` : 탑햇 연산
  - `MorphologyOperation.BlackHat` : 블랙햇 연산

그라디언트 연산

```
public Mat Gradient(Mat src)
{
    gradient = new Mat();
    Mat element = Cv2.GetStructuringElement(MorphShapes.Ellipse, new Size(5, 5));
    Cv2.MorphologyEx(src, gradient, MorphTypes.Gradient, element, new Point(-1, -1), 3);
    return gradient;
}
```

- 팽창(Dilate)에서 침식(Erode)을 제외
- Gradient = Dilate(src) - Erode(src)

탑햇

```
public Mat Tophat(Mat src)
{
    tophat = new Mat();
    Mat element = Cv2.GetStructuringElement(MorphShapes.Ellipse, new Size(5, 5));
    Cv2.MorphologyEx(src, tophat, MorphTypes.TopHat, element, new Point(-1, -1), 3);
    return tophat;
}
```

- 원본에서 열기 연산을 제외
- TopHat = src – Open
  (Open = Dilate(Erode(src)), 침식(Erode) 후, 팽창(Dilate) 적용)

블랙햇

```
public Mat Blackhat(Mat src)
{
    blackhat = new Mat();
    Mat element = Cv2.GetStructuringElement(MorphShapes.Ellipse, new Size(5, 5));
    Cv2.MorphologyEx(src, blackhat, MorphTypes.BlackHat, element, new Point(-1, -1), 3);
    return blackhat;
}
```

- 닫기 연산에서 원본을 제외
- BlackHat = Close – src
  (Close = Erode(Dilate(src)), 팽창(Dilate) 후, 침식(Erode) 적용)

5. 기하학적 변환

영상이나 이미지 위에 기하학적으로 변환하기 위해 사용

- WarpPerspective는 4개의 점을 매핑
- Cv2.GetPerspectiveTransform()가 List<Point2f> 형식을 사용함
  (float형 2D형식으로 값을 받기 때문에 float로 선언)

- Cv2.GetPerspectiveTransform(변환 전 픽셀 좌표, 변환 후 픽셀 좌표)
- Cv2.WarpPerspective(원본 배열, 결과 배열, 행렬, 결과 배열의 크기,
  보간법, 테두리 외삽법, 테두리 색상)

- 보간법, 테두리 외삽법, 테두리 색상 생략가능
  (생략할 경우 기본 값은 각각 선형 보간, 경계 외삽법, 0)

아핀 변환

```
public Mat Affine(Mat src, float[] xy)
{
    affine = new Mat();
    List<Point2f> src_pts = new List<Point2f>()
    {
        new Point2f(0.0f, 0.0f),
        new Point2f(0.0f, src.Height),
        new Point2f(src.Width, 0.0f),
        new Point2f(src.Width, src.Height)
    };
    List<Point2f> affine_pts = new List<Point2f>()
    {
        new Point2f(xy[0], xy[1]),
        new Point2f(xy[2], xy[3]),
        new Point2f(xy[4], xy[5]),
        new Point2f(xy[6], xy[7])
    };
    Mat matrix = Cv2.GetPerspectiveTransform(affine_pts, src_pts);
    Cv2.WarpPerspective(src, affine, matrix, new OpenCvSharp.Size(src.Width, src.Height));
    return affine;
}
```

- 변환되어 이미지를 출력할 4개의 지점을 선택
- 마우스 이벤트로 입력된 값을 배열로 받음
- 순서는 좌상, 좌하, 우상, 우하

- WarpAffine는 3개의 점을 매핑
- 아핀 변환은 사각형을 평행사변형으로 변환(길이의 비와 평행성이 보존)

6. 블러

흐림 효과, 블러링(Blurring), 스무딩(Smoothing)라고도 부름
노이즈를 줄이거나 외부 영향을 최소화하는 데 사용
노이즈를 제거하여 연산 시 계산을 빠르고 정확하게 수행하기 위한 전처리

- 해당 픽셀의 주변값들과 비교하고 계산하여 픽셀들의 색상 값을 재조정
- 커널(Kernel)은 이미지에서 (x, y)의 픽셀과 (x, y) 픽셀 주변을 포함한 작은 크기의 공간
- 고정점(Anchor Point)은 커널을 통한 픽셀 재조정의 기준점
- 테두리 외삽법(Border Extrapolation)은 이미지 가장자리 부분의 처리 방식
  (이미지 가장자리 부분 계산 불가능, 테두리 이미지 바깥쪽에 가상의 픽셀을 만들어 처리)

- 심플 블러와 가우시안 블러를 주로 사용

심플 블러

```
public Mat Blur(Mat src, int size)
{
    blur = new Mat();
    Cv2.Blur(src, blur, new Size(size, size), new Point(-1, -1), BorderTypes.Default);
    return blur;
}
```

- Cv2.Blur(원본 배열, 결과 배열, 커널, 고정점, 테두리 외삽법)
- 단순 흐림 효과 함수(Cv2.Blur)는 각 픽셀에 대해 커널을 적용해 모든 픽셀의 단순 평균을 구함
- 고정점 위치를 (-1, -1)할당 경우, 중심부가 고정점

가우시안 블러

```
public Mat Gaussian(Mat src, int size)
{
    gaussian = new Mat();
    Cv2.GaussianBlur(src, gaussian, new Size(size, size), 0,0, BorderTypes.Default);
    return gaussian;
}
```

- Cv2.GaussianBlur(원본 배열, 결과 배열, 커널, X 방향 표준 편차, Y 방향 표준 편차,
  테두리 외삽법)
- 가우시안 흐림 효과 함수(Cv2.GaussianBlur)는 이미지의 각 지점에 가우시안 커널을 적용해 합산한 후에 출력 이미지를 반환
- X 방향 표준 편차와 Y 방향 표준 편차는 가우스 커널의 표준 편차를 의미
- X 방향 표준 편차가 0인 경우, Y 방향 표준 편차의 값은 X 방향 표준 편차의 값과 같아짐
- 모두 0으로 설정한다면 커널 크기를 고려해 자동 설정

3-6. 영상 및 이미지 검출

노이즈가 제거된 영상 및 이미지에서 필요한 특징이나 정보를 검출

1. 코너 검출

- 영상이나 이미지의 모서리(코너)를 검출하기 위해 사용
- Cv2.GoodFeaturesToTrack(그레이스케일, 최대 코너 점의 수, 코너 품질, 최소 거리,
  마스크(null일 경우 전체), 검출 블록 크기,
  Harris 코너 검출기 사용, 코너 강도)

코너 검출

```
public Mat Corner(Mat src, int size)
{
    corner = new Mat();
    gray = new Mat();
    Cv2.CopyTo(src, corner);

    //이미 채널이 1이면(그레이스케일이 되어있으면) RGB2GRAY에서 오류가 생김
    if (src.Channels() != 1)  Cv2.CvtColor(src, gray, ColorConversionCodes.RGB2GRAY);
    else Cv2.CopyTo(src, gray);

    Point2f[] corners;
    corners = Cv2.GoodFeaturesToTrack(gray, 100, 0.03, size, null, 3, false, 0);

    for (int i = 0; i < corners.Length; i++)
    {
        OpenCvSharp.Point pt = new OpenCvSharp.Point((int)corners[i].X, (int)corners[i].Y);
        Cv2.Circle(corner, pt, 5, Scalar.Red, 2, LineTypes.AntiAlias);
    }
    return corner;
}
```

- GoodFeaturesToTrack 함수는 이미지의 강도 변화에 따라 코너를 검출
  (그레이스케일 이미지는 단일 강도 값을 사용하므로 계산이 간단하고 정확)
  (특징 점 검출 알고리즘은 이미지의 강도 변화(밝기 변화)에 민감)

- 검출한 코너를 보여주기 위해 Cv2.Circle을 사용해 그려준 뒤 return 값을 반환
- Cv2.Circle(대상 이미지, 중심 좌표, 반지름, 색상, 두께, 선타입, 반지름 정확도(기본값 0))

2. 가장자리 검출

- 밝기가 큰 폭으로 변하는 지점을 가장자리로 간주
- 픽셀의 밝기 변화율(Rate of change)이 높은 부분을 검출
  (주로 1차 미분이나 2차 미분을 사용)

- 미분을 할 경우 노이즈에 큰 영향을 받기에 블러를 한 뒤 가장자리를 검출하는 것이 좋음
- 이미지는 샘플링과 양자화가 처리된 데이터, 밝기의 평균변화율이 아닌 순간변화율을 계산

- Cv2.Canny(원본, 결과, 하위 임곗값, 상위 임곗값, 소벨 연산자 크기, L2 그레이디언트)
  (픽셀이 상위 임곗값보다 큰 기울기를 가지면 가장자리로 판단)
  (L2 그레이디언트(L2gradient)는 좀 더 정확하게 계산할지 선택)

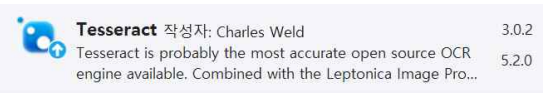
케니 엣지

```
public Mat Canny(Mat src)
{
    canny = new Mat();
    Cv2.Canny(src, canny, 100, 200, 3, true);
    return canny;
}
```

3. 글자 검출

- Tesseract OCR를 이용하여 Bitmap으로된 이미지에서 문자를 인식하여 string형식으로 반환
- Tesseract 언어 데이터 파일 필요 kor.traineddata
- TesseractEngine(언어 데이터 파일 경로, 언어, 엔진모드)

- EngineMode.*
  - EngineMode.Default : 기본값으로 판독
  - EngineMode.CubeOnly : 큐브 방식으로 정확도는 높아지지만, 속도가 느림
  - EngineMode.TesseractOnly : Tesseract 방식만 실행하며, 속도가 가장 빠름
  - EngineMode.TesseractAndCube : 큐브와 Tesseract 방식의 결합, 가장 높은 정확도



글자 검출

```
using Tesseract;

public string Tesseract(Mat src)
{
    tesser = src.ToBitmap();
    var ocr = new TesseractEngine("./tessdata", "kor+eng", EngineMode.Default);
    var texts = ocr.Process(tesser);
    return texts.GetText();
}
```

- ocr 변수에 TesseractEngine()을 이용하여 언어 데이터 파일을 사용하여 판독
- 영어 : eng, 한국어 : kor, 영한 : kor+eng
- ocr에서 셋팅된 방법으로 img를 이용해 판독할 문자들을 저장
- GetText()를 이용하여 string형태로 저장

4. 모형 검출

영상이나 이미지에서 원, 사각형 등의 형태를 찾기 위해 사용

- HoughCircles를 이용해 원 검출 가능
- Cv2.HoughCircles(그레이스케일, 검출된 원 정보 저장 배열, Hough 변환 방법, 해상도 배율, 최소거리, 하위 임계값, 중심 임계값, 최소반지름, 최대반지름)

원 검출

```
public Mat Circle(Mat src)
{
    circle = new Mat();
    Mat dst = new Mat();
    Cv2.CopyTo(src, dst);

    Mat kernel = Cv2.GetStructuringElement(MorphShapes.Rect, new OpenCvSharp.Size(3, 3));
    if (src.Channels() != 1) Cv2.CvtColor(src, circle, ColorConversionCodes.BGR2GRAY);
    else Cv2.CopyTo(src, circle);

    //좀 더 명확한 원 검출을 하기 위해서는 블러처리를 할 필요가 있음
    //블러 처리된 이미지를 사용하거나 중간에 블러처리 실행

    CircleSegment[] circles = Cv2.HoughCircles(circle, HoughModes.Gradient, 1, 100, 100, 35, 0, 0);
    for (int i = 0; i < circles.Length; i++)
    {
        OpenCvSharp.Point center = new OpenCvSharp.Point(circles[i].Center.X, circles[i].Center.Y);

        Cv2.Circle(dst, center, (int)circles[i].Radius, Scalar.White, 3);
        Cv2.Circle(dst, center, 5, Scalar.AntiqueWhite, Cv2.FILLED);
    }
    return dst;
}
```

- 정확성을 높이기 위해 그레이스케일(원의 경계 강도 변화 명확), 블러(노이즈 제거)를 사용
- 검출된 원을 Cv2.Circle을 이용해 표시한 뒤 반환
-

사각형 검출

```
public Mat Square(Mat src)
{
    OpenCvSharp.Point[] square = FindSquare(src);
    Mat dst = DrawSquare(src, square);
    return dst;
}
```

- 사각형 검출

```
public static double CalcAngle(OpenCvSharp.Point pt1, OpenCvSharp.Point pt0, OpenCvSharp.Point pt2)
{
    //cos각도 수식
    double u1 = pt1.X - pt0.X, u2 = pt1.Y - pt0.Y;
    double v1 = pt2.X - pt0.X, v2 = pt2.Y - pt0.Y;

    double numerator = u1 * v1 + u2 * v2;
    double dewnominator = Math.Sqrt(u1*u1+u2*u2)*Math.Sqrt(v1*v1+v2*v2);
    return numerator / dewnominator;
}
```

- 각도를 계산 후 반환(사각형인지 검증용)

사각형 검출

```
public OpenCvSharp.Point[] FindSquare(Mat src)
{
    Mat[] split = Cv2.Split(src);//bgr채널 나눈뒤 이진화(정확성을 위해)
    Mat blur = new Mat();//정확성을 위한 블러와 이진화
    Mat binary = new Mat();
    OpenCvSharp.Point[] square = new OpenCvSharp.Point[4];//반환할 포인트

    int N = 10; //이진화 종류(정확성을 위해 임계값을 다르게 이진화를 한다.)
    double cos = 1; //사각형 각도
    double max = src.Size().Width * src.Size().Height * 0.9;//입력 이미지의 90%까지의 사각형 이하만 명함으로 인정
    double min = src.Size().Width * src.Size().Height * 0.1;//10%이상만

    for(int channel =0; channel < 3; channel++)//각채널
    {
        Cv2.GaussianBlur(split[channel], blur, new OpenCvSharp.Size(5, 5), 1);//블러처리
        for(int i = 0; i < N; i++)//이진화10개
        {
            Cv2.Threshold(blur, binary, i * 255 / N, 255, ThresholdTypes.Binary);
            //윤곽선검출
            OpenCvSharp.Point[][] contours;//찾은윤곽선 저장 리스트(윤곽선은 점들의 리스트로 표현)
            HierarchyIndex[] hierarchy;//윤곽선 간 계층 구조(중첩이 어떻게 되어있는지)
            Cv2.FindContours(binary, out contours, out hierarchy, RetrievalModes.External,
                ContourApproximationModes.ApproxTC89KCOS);
            //RetrievalModes.External 윤곽선 검색 종류 : 외곽 윤곽선 검색
            //ContourApproximationModes.ApproxTC89KCOS 윤곽선 근사화 방법 : Teh - chin 체인 코드 알고리즘으로
            for (int j=0; j<contours.Length; j++)//찾은 윤곽선 수만큼
            {
                //윤곽선 길이 계산 (윤곽선 구성 점 배열, 폐곡선 여부)
                double perimeter = Cv2.ArcLength(contours[j], true);
                //윤곽선을 다각형으로 근사화(윤곽선, 길이에 대한 허용 오차, 폐곡선 여부)
                OpenCvSharp.Point[] result = Cv2.ApproxPolyDP(contours[j], perimeter * 0.02, true);
                //윤곽선 면적 계산
                double area = Cv2.ContourArea(result);
                //윤곽선이 볼록한지 확인
                bool convex = Cv2.IsContourConvex(result);

                //사각형 검증
                if(result.Length == 4&&area> min&& area < max && convex)
                {
                    double[] angles = new double[4];
                    for(int k =1; k < 5; k++)
                    {
                        double angle = Math.Abs(CalcAngle(result[(k - 1) % 4], result[k % 4], result[(k + 1) % 4]));
                        angles[k-1] = angle;
                    }
                    if (angles.Max() < cos && angles.Max() < 0.15)
                    {
                        cos=angles.Max();
                        square = result;
                    }
                }
            }
        }
    }
    return square;
}
```

- 사각형 검출 메소드

```
public Mat DrawSquare(Mat src, OpenCvSharp.Point[] square)
{
    Mat drawSquare = src.Clone();
    OpenCvSharp.Point[][] pts = new OpenCvSharp.Point[][] { square};
    Cv2.Polylines(drawSquare, pts, true, Scalar.Yellow, 3, LineTypes.AntiAlias, 0);
    return drawSquare;
}
```

- 검출한 사각형에 사각형 표시

5. 색상 검출

이미지에서 색상을 검출하기 위해 사용

- HSV(색상 Hue, 채도 Saturation, 명도 Value)사용, (BGR이나 RGB 패턴으로는 색상 구별 어려움)
- Cv2.InRange(입력 이미지, 색상 범위 하한, 상한, 결과 이미지)
- 입력 이미지에서 지정한 색상 범위에 해당하는 픽셀을 검출, 해당 부분만 추출

HSV 검출

```
public Mat Color(Mat src, string srcColor)
{
    color = new Mat();
    Mat hsv = new Mat();

    Cv2.CvtColor(src, hsv, ColorConversionCodes.BGR2HSV);
    Mat[] HSV = Cv2.Split(hsv);
    Mat H_color = new Mat();
    switch (srcColor)
    {
        case "red":
            //0~179, 0~255, 0~255
            Mat H_color_upRed = new Mat();
            Mat H_color_downRed = new Mat();
            Cv2.InRange(HSV[0], new Scalar(0, 100, 100), new Scalar(10, 255, 255), H_color_upRed);
            Cv2.InRange(HSV[0], new Scalar(178, 100, 100), new Scalar(179, 255, 255), H_color_downRed);
            Cv2.AddWeighted(H_color_upRed, 1.0, H_color_downRed, 1.0, 0.0, H_color);
            break;
        case "orange":
            Cv2.InRange(HSV[0], new Scalar(11), new Scalar(25), H_color);
            break;
        case "yellow":
            Cv2.InRange(HSV[0], new Scalar(26), new Scalar(40), H_color);
            break;
        case "green":
            Cv2.InRange(HSV[0], new Scalar(41), new Scalar(84), H_color);
            break;
        case "cyan":
            Cv2.InRange(HSV[0], new Scalar(85), new Scalar(110), H_color);
            break;
        case "blue":
            Cv2.InRange(HSV[0], new Scalar(111), new Scalar(140), H_color);
            break;
        case "magenta":
            Cv2.InRange(HSV[0], new Scalar(141), new Scalar(165), H_color);
            break;
        case "pink":
            Cv2.InRange(HSV[0], new Scalar(166), new Scalar(177), H_color);
            break;
        default:
            MessageBox.Show("색상 범위 오류");
            break;
    }
    Cv2.BitwiseAnd(hsv, hsv, color, H_color);
    Cv2.CvtColor(color, color, ColorConversionCodes.HSV2BGR);

    return color;
}
```

- 원하는 색상을 선택하기 위해 매개변수로 넘겨받아 사용
- Cv2.Split() 함수를 사용해 HSV 색상 공간에서의 각 색상 채널을 개별적으로 분리하여 처리 (색상, 채도, 밝기 순으로 배열에 삽입)
- Red의 경우 HSV 색상표에서 처음과 끝으로 나뉘기에 Cv2.AddWeighted() 함수를 이용해 합침
- Cv2.AddWeighted() 함수는 두 개의 이미지를 가중합하여 새로운 이미지를 생성
- Cv2.AddWeighted(첫 이미지, 가중치 계수, 두 번째 이미지, 가중치 계수, 추가 값, 결과 이미지)

3-7. 영상 및 이미지 응용

전처리 과정을 거친 이미지를 응용하여 검출, 분류, 변환, 사용


```
OpenCV_filter openCV_Filter = new OpenCV_filter();
OpenCV_Detection openCV_Detection = new OpenCV_Detection();
```

- 이미지를 전처리하기 위해 위에서 만든 변환, 검출 클래스 사용

1. 명함 검출

- Tesseract 라이브러리를 이용하여 명함에서 문자를 검출

명함 검출

```
public Mat CardDetect(Mat src)
{
    card = new Mat();
    OpenCvSharp.Point[] square = openCV_Detection.FindSquare(src);
    List<Point2f> src_pts = new List<Point2f>()
    {
        new Point2f(0.0f, 0.0f),
        new Point2f(0.0f, src.Height),
        new Point2f(src.Width, 0.0f),
        new Point2f(src.Width, src.Height)
    };

    square = sortPoint(square);
    List<Point2f> affine_pts = new List<Point2f>()
    {
        new Point2f(square[0].X, square[0].Y),
        new Point2f(square[1].X, square[1].Y),
        new Point2f(square[2].X, square[2].Y),
        new Point2f(square[3].X, square[3].Y)
    };
    Mat matrix = Cv2.GetPerspectiveTransform(affine_pts, src_pts);
    Cv2.WarpPerspective(src, card, matrix,
        new OpenCvSharp.Size(src.Width, src.Height));

    Text = openCV_Detection.Tesseract(card);

    return card;
}
```

- OCR 하기 쉬운 형태로 이미지를 전처리 후 문자 검출
- 사각형 검출 뒤 검출한 좌표를 정렬
- 정렬한 좌표로 기하학적 변환
- 만든 검출 클래스를 사용하여 문자 검출

```
private OpenCvSharp.Point[] sortPoint(OpenCvSharp.Point[] square)
{
    OpenCvSharp.Point[] sort = new OpenCvSharp.Point[4];
    int min1 = int.MaxValue; int min2 = int.MaxValue;
    int minX1 = 0; int minX2 = 0;
    for (int i =0; i < 4; i++)
    {
        if (square[i].X < min1)
        {
            min1 = square[i].X;
            minX1 = i;
        }
        else if (square[i].X <= min2 && i != minX1)
        {
            min2 = square[i].X;
            minX2 = i;
        }
    }
    if (square[minX1].Y < square[minX2].Y)//왼쪽 좌표 위아래 정렬
    {
        sort[0] = square[minX1];
        sort[1] = square[minX2];
    }
    else
    {
        sort[1] = square[minX1];
        sort[0] = square[minX2];
    }
    int[] maxX = new int[2]; int count = 0;
    for (int i =0; i < 4; i++)//안뽑힌 두수 뽑기
    {
        if (i != minX1 && i != minX2 && count ==0)
        {
            maxX[0] = i;
            count++;
        }
        else if(i != minX1 && i != minX2 && maxX[0] != i)
        {
            maxX[1] = i;
        }
    }
    //오른쪽 위아래 정렬
    if(square[maxX[0]].Y < square[maxX[1]].Y)
    {
        sort[2] = square[maxX[0]];
        sort[3] = square[maxX[1]];
    }
    else
    {
        sort[2] = square[maxX[1]];
        sort[3] = square[maxX[0]];
    }
    return sort;
```

2. 색상 검출

- 관심 영역 설정 후 난수를 생성하여 정확도를 높이고 색상 농도 검출

색상 검출

```
public void whatIsColor(Mat src)
{
    //난수설정
    Random wRandom = new Random();
    Random hRandom = new Random();
    Mat hsvImage = new Mat();
    Cv2.CvtColor(src, hsvImage,
        ColorConversionCodes.BGR2HSV)
    //랜덤 지점 설정
    int wNum;
    int hNum;
    //검출된 색상 정보
    Vec3b hsvColor;
    byte hue;//색상
    byte saturation;//채도
    byte value;//명도
    //색상 카운트
    int redCount = 0;
    int orangeCount = 0;
    int yellowCount = 0;
    int greenCount = 0;
    int cyanCount = 0;
    int blueCount = 0;
    int magentaCount = 0;
    int pinkCount = 0;
    int allCount = 0;
    //무채색 카운트
    int blackCount = 0;
    int whiteCount = 0;
    //창에 쓸 문구
    colorStr = "";
```

- 변수 설정

```
while (allCount<100)//난수 발생
{
    wNum = wRandom.Next(hsvImage.Cols);
    hNum = hRandom.Next(hsvImage.Rows);
    hsvColor = hsvImage.At<Vec3b>(hNum, wNum);//색상검출
    hue = hsvColor.Item0;//색상
    saturation = hsvColor.Item1;//채도
    value = hsvColor.Item2;//명도

    // 채도와 명도 범위 (0~255)
    int minSaturation = 50;
    int maxSaturation = 255;
    int minValue = 50;
    int maxValue = 255;

    // 흑색과 백색 정의
    int minBlackValue = 0;
    int maxBlackValue = 50;
    int minWhiteValue = 200;
    int maxWhiteValue = 255;
    int maxBlackSaturation = 50; // 흑색은 채도가 매우 낮음
    int maxWhiteSaturation = 30;  // 백색은 채도가 낮음
    if (saturation >= minSaturation && saturation <= maxSaturation &&
        value >= minValue && value <= maxValue)
    {
        if ((hue >= 0 && hue <= 10) || (hue >= 170 && hue <= 180))
            redCount++;
        else if (hue >= 11 && hue <= 25)
            orangeCount++;
        else if (hue >= 26 && hue <= 35)
            yellowCount++;
        else if (hue >= 36 && hue <= 70)
            greenCount++;
        else if (hue >= 71 && hue <= 100)
            cyanCount++;
        else if (hue >= 101 && hue <= 130)
            blueCount++;
        else if (hue >= 131 && hue <= 160)
            magentaCount++;
        else if (hue >= 161 && hue <= 170)
            pinkCount++;
    }
```

- HSV로 색상을 검출한 뒤 채도와 명도가 일정이상으로 낮으면 무채색으로 간주

```
    else if (saturation <= maxBlackSaturation && value >= minBlackValue && value <= maxBlackValue)
    {
        blackCount++;
    }
    else if (saturation <= maxWhiteSaturation && value >= minWhiteValue && value <= maxWhiteValue)
    {
        whiteCount++;
    }
    //색상 100개 검출 완료시 끝
    allCount = redCount + orangeCount + yellowCount + greenCount
        + cyanCount + blueCount + magentaCount + pinkCount+ blackCount+ whiteCount;
}
```

- 무채색을 포함하여 난수 100개 카운트
- 채도는 OpenCV에서 범위가 0~179

색상 검출

```
    //색상 농도 보여줄 텍스트
    if (redCount != 0) colorStr = "Red : " + redCount + "%" + "₩n₩r";
    if (orangeCount != 0) colorStr = colorStr + "Orange : " + orangeCount + "%" + "₩n₩r";
    if (yellowCount != 0) colorStr = colorStr + "Yellow : " + yellowCount + "%" + "₩n₩r";
    if (greenCount != 0) colorStr = colorStr + "Green : " + greenCount + "%" + "₩n₩r";
    if (cyanCount != 0) colorStr = colorStr + "Cyan : " + cyanCount + "%" + "₩n₩r";
    if (blueCount != 0) colorStr = colorStr + "Blue : " + blueCount + "%" + "₩n₩r";
    if (magentaCount != 0) colorStr = colorStr + "Magenta : " + magentaCount + "%" + "₩n₩r";
    if (pinkCount != 0) colorStr = colorStr + "Pink : " + pinkCount + "%" + "₩n₩r";
    if (blackCount != 0) colorStr = colorStr + "Black : " + blackCount + "%" + "₩n₩r";
    if (whiteCount != 0) colorStr = colorStr + "White" + whiteCount + "%";

    //가장 비율이 높은 색상
    int[] counts = { redCount, orangeCount, yellowCount, greenCount, cyanCount, blueCount,
        magentaCount, pinkCount, blackCount, whiteCount };
    string[] colorsr = { "Red", "Orange", "Yellow", "Green", "Cyan", "Blue", "Magenta",
        "Pink", "Black", "White"};
    // 색상 및 카운트 정렬
    SortColorsByCount(ref counts, ref colorsr);

    //컬러 끝 str에 지정
    int maxCount = counts[0];
    maxColor = colorsr[0];
    biggestColor = maxColor +" : "+ maxCount.ToString()+"%";
}
```

- 매개변수 Mat src 전체의 색상 농도를 파악해 퍼센트로 출력하기 때문에 메소드를 사용하고자 할땐 src에는 알고자하는 이미지의 범위를 관심영역으로 설정
- 여기선 마우스 이벤트를 통해 알고자하는 이미지의 일정 부분을 입력하여 사용

```
//가장 비율이 높은 색상 구하기
참조 1개
public static void SortColorsByCount(ref int[] counts, ref string[] colors)
{
    //색상 배열 동기화
    int length = counts.Length;
    var indices = Enumerable.Range(0, length).ToArray();

    Array.Sort(counts, indices, Comparer<int>.Create((a, b) => b.CompareTo(a)));

    string[] sortedColors = new string[length];
    for (int i = 0; i < length; i++)
    {
        sortedColors[i] = colors[indices[i]];
    }
    colors = sortedColors;
}
```

- 색상 전체 검출 메소드에서 사용할 정렬 메소드
  (전체 이미지에서 원을 검출한 뒤 각각의 원에서 가장 높은 색상을 텍스트로 표시하는 메소드)

색상 전체 검출

```
public Mat AllCircleColor(Mat src)
{
    circle = new Mat();
    Mat dst = new Mat();
    Cv2.CopyTo(src, dst);

    Mat kernel = Cv2.GetStructuringElement(MorphShapes.Rect, new OpenCvSharp.Size(3, 3));
    if (src.Channels() != 1) Cv2.CvtColor(src, circle, ColorConversionCodes.BGR2GRAY);
    else Cv2.CopyTo(src, circle);
    //블러 추가
    Cv2.GaussianBlur(circle, circle, new OpenCvSharp.Size(11, 11), 0, 0, BorderTypes.Default);

    CircleSegment[] circles = Cv2.HoughCircles(circle, HoughModes.Gradient, 1, 100, 100, 35, 0, 0);
    for (int i = 0; i < circles.Length; i++)
    {
        OpenCvSharp.Point center = new OpenCvSharp.Point(circles[i].Center.X- circles[i].Radius,
            circles[i].Center.Y- circles[i].Radius);
        Mat area = new Mat(src, new OpenCvSharp.Rect((int)(circles[i].Center.X - circles[i].Radius / 2),
                                        (int)(circles[i].Center.Y - circles[i].Radius / 2),
                                        (int)(circles[i].Radius),
                                        (int)(circles[i].Radius)
                                        )
                        );
        whatIsColor(area);
        //글자 표시하기
        Cv2.PutText(dst, biggestColor, center, HersheyFonts.HersheySimplex, 0.7, biggestColorScalar(maxColor));
    }
    return dst;
}
```

- 전체 이미지에서 원 형태의 물체를 검출하여 각각의 대상에 가장 짙은 색상 농도 표시

```
public Scalar biggestColorScalar(string str)
{
    switch (str)
    {
        case "Red":
            return new Scalar(0, 255, 255);
        case "Orange":
            return new Scalar(15, 255, 255);
        case "Yellow":
            return new Scalar(30, 255, 255);
        case "Green":
            return new Scalar(60, 255, 128);
        case "Cyan":
            return new Scalar(90, 255, 128);
        case "Blue":
            return new Scalar(120, 255, 255);
        case "Magenta":
            return new Scalar(150, 200, 255);
        case "Pink":
            return new Scalar(150, 128, 255);
        case "Black":
            return new Scalar(25, 25, 25);
        case "White":
            return new Scalar(240, 240, 240);
        default:
            return new Scalar(0, 0, 0);
    }
}
```

- 표시할 색상에 따라 글자색도 동일한 색상으로 표시하기 위한 메소드

---

3. 동전 검출

- 물체에서 원을 검출해 반지름을 계산하여 동전을 분류

동전 검출

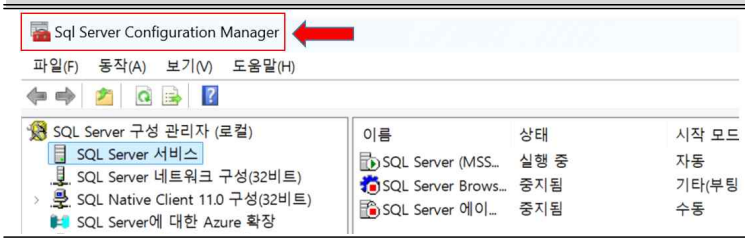
```
public Mat coinCheck(Mat src)
{
    Mat dst = new Mat();
    coin = new Mat();
    Cv2.CopyTo(src, dst);
    //그레이스케일
    if (src.Channels() != 1) Cv2.CvtColor(src, coin, ColorConversionCodes.BGR2GRAY);
    else Cv2.CopyTo(src, coin);
    //블러
    Cv2.GaussianBlur(coin, coin, new OpenCvSharp.Size(11, 11), 0, 0, BorderTypes.Default);
    //원검출
    CircleSegment[] circles = Cv2.HoughCircles(coin, HoughModes.Gradient, 1, 100, 100, 35, 0, 0);
    for (int i = 0; i < circles.Length; i++)
    {
        OpenCvSharp.Point center = new OpenCvSharp.Point(circles[i].Center.X - circles[i].Radius, circles[i].Center.Y - circles[i].Radius);
        Cv2.Rectangle(dst, new OpenCvSharp.Point(circles[i].Center.X - circles[i].Radius, circles[i].Center.Y - circles[i].Radius),
                           new OpenCvSharp.Point(circles[i].Center.X + circles[i].Radius, circles[i].Center.Y + circles[i].Radius),
                           Scalar.Red, 1);

        coinClassification(circles[i].Radius);
        Cv2.PutText(dst, coinSize, center, HersheyFonts.HersheySimplex, 0.7, Scalar.Red);
    }

    return dst;
}

private void coinClassification(float rad)
{
    //높이 약15cm일때의 동전 크기 - 상황마다 조절 필요
    if (rad > 45 && rad <60) coinSize = "100";
    else if (rad < 80) coinSize = "500";
    else coinSize = "길이 : " + rad.ToString();
}
```

- 원 검출 알고리즘을 응용하여 반지름 검출
- 특정 높이(15cm)에서의 동전 반지름을 알아낸 뒤 범위를 지정
- 100원, 500원일 시 화면 텍스트에 동전의 종류를 출력
- 크기가 범위 이외일 시 반지름 크기를 출력

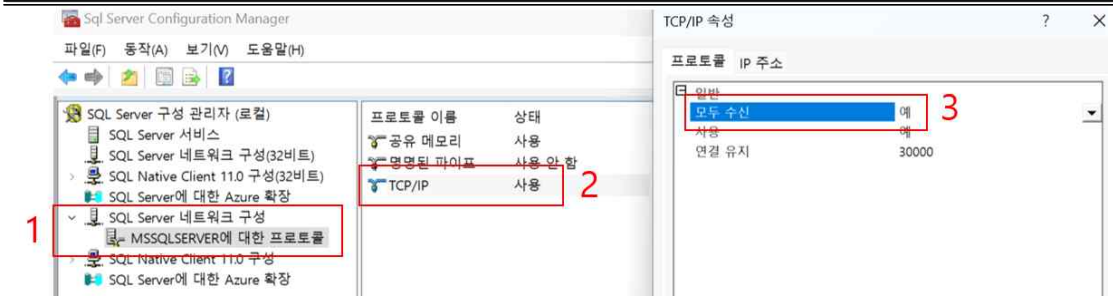
4. 바코드

- ZXing.Net을 사용하여 바코드에서 정보 취득
- 바코드에 관한 정보를 띄운 뒤 DB에 삽입(사용 DB는 MSSQL)
- 삽입된 바코드 정보에서 회사를 검색하는 검색창 기능 추가

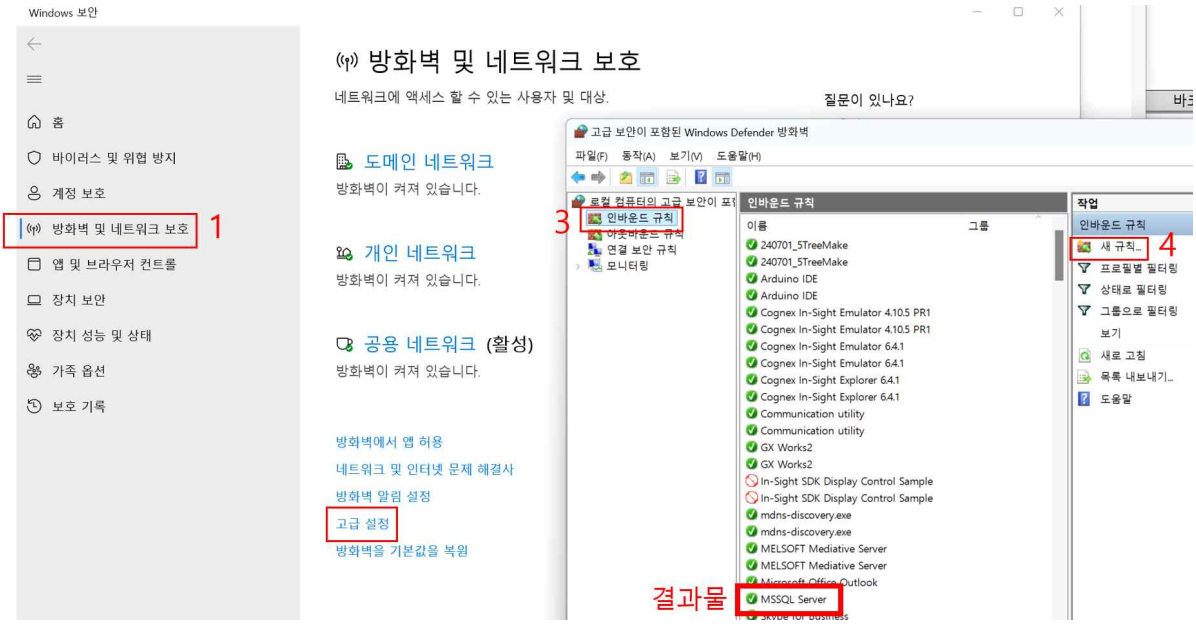
바코드 – MSSQL 설정



바코드 - MSSQL 설정 & 방화벽 설정

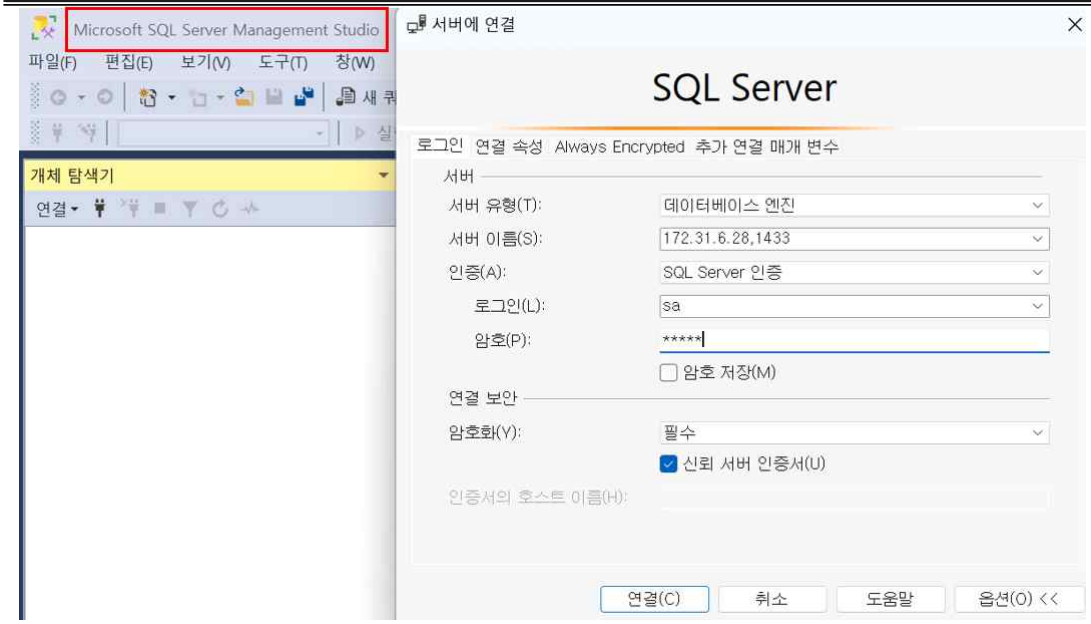


- MSSQL을 설치 후 SQL Server Configuration Manger를 실행하여 프로토콜 설정

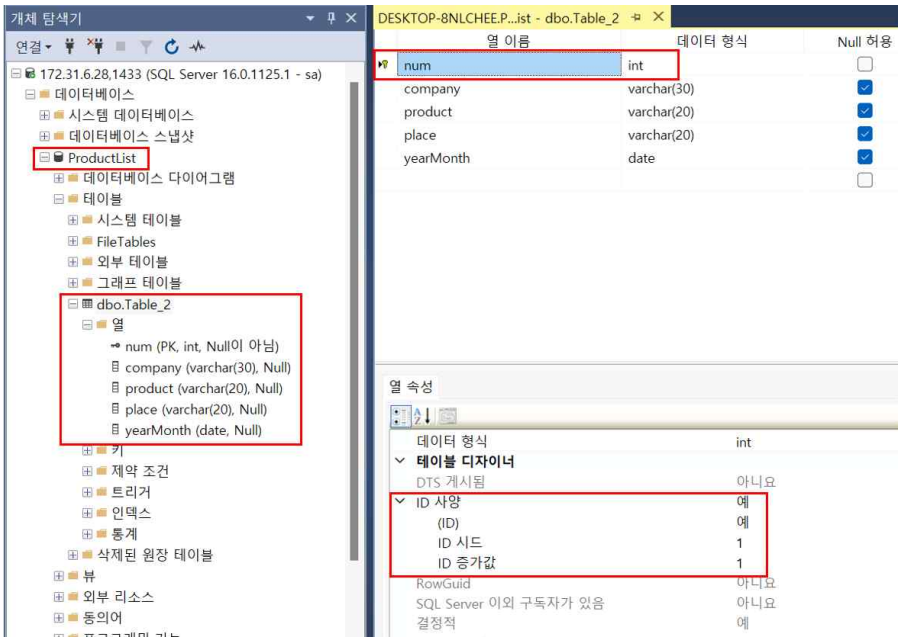


- 방화벽 설정
- 이후 Allow the connection(해당 포트로 접근 허용) -> 모든 영역 적용(Domain, Private, Public) -> 설정 이름, 간략한 설정 입력(MSSQL Server, 외부 접근 포트) 순으로 진행

바코드 - DB



- Microsoft SQL Server Management Studio 실행
- DB를 설치하며 설정한 아이디로 로그인
- 서버를 연 컴퓨터 IP와 지정한 포트 입력
- 각각의 설정을 입력후 연결



- DB를 생성 후 Table 추가
- 기본키를 설정한 뒤 Column(열) 설정
- 기본키는 데이터 형식을 INT로 설정해주고 ID를 자동으로 추가되게 설정
- 나머지 Column들의 데이터 형식을 varchar로 설정, 제조일자를 넣을 yearMonth만 date로 설정
- 각각 기본키, 회사명, 상품명, 생산지, 제조일자 순

바코드 – 입력

```
private void UserControl_Barcode_event_UC_barcodeBTN_Click()
{
    userControl_Barcode.PictureBox1.Image = pictureBox1.Image;
    string decoded = "";
    BarcodeReader reader = new BarcodeReader();//인스턴스 생성
    Result result = reader.Decode((Bitmap)userControl_Barcode.PictureBox1.Image);//바코드 해석
    db.connect();
    // 결과를 확인합니다.
    if (result != null)
    {
        int[] num = { 2, 4, 2, 4, 1 };
        int currentPosition = 0;

        for (int i = 0; i < num.Length; i++)
        {
            if (currentPosition + num[i] <= result.Text.Length)
            {
                string segment = result.Text.Substring(currentPosition, num[i]);
                currentPosition += num[i];

                if (i == 0)
                {
                    switch (segment)
                    {
                        case "11":
                            company = "삼성";
                            break;
                        case "21":
```

- BarcodeReader 클래스를 사용하여 바코드 해석 후 회사명, 상품명, 생산지, 제조일자로 분리

```
else if (i == 4)
{
    string rt = result.Text; // EAN-13 바코드 문자열
    int len = rt.Length;
    if (len != 13) MessageBox.Show("EAN-13 바코드는 13자리 숫자로 구성되어야 합니다.");
    int sumOdd = 0; // 홀수 자리 숫자의 합
    int sumEven = 0; // 짝수 자리 숫자의 합
    // 홀수 및 짝수 자리에 있는 숫자 합산 (1-based index 기준)
    for (int j = 0; j < len - 1; j++)
    {
        int digit = int.Parse(rt[j].ToString());
        if (j % 2 == 0) // 0-based index에서 홀수 자리에 해당 (1, 3, 5, ...)
        {
            sumOdd += digit;
        }
        else
        {
            sumEven += digit;
        }
    }
    // 홀수 자리 합에 3을 곱하고, 짝수 자리 합과 더함
    int total = sumOdd + (sumEven * 3);
    // 총합을 10으로 나눈 나머지를 구하고, 체크디지트 계산
    int remainder = total % 10;
    int checkDigit = (10 - remainder) % 10;
    // 실제 체크디지트 (마지막 자리 숫자) 추출
    int actualCheckDigit = int.Parse(rt.Substring(len - 1, 1));
    // 체크디지트 검증
    luhn = (checkDigit == actualCheckDigit);
}
```

- EAN-13 바코드 체크디지트를 검증한 후 바코드 정보 출력

```
    //해석된 텍스트, 바코드 형식
    decoded = result.ToString() + "₩r₩n" + result.BarcodeFormat.ToString()
         + $"₩r₩n {company}₩r₩n {product}₩r₩n {production}₩r₩n {date} ₩r₩n {luhn}";
    userControl_Barcode.Label1.Text = decoded;
    if (luhn) db.insert(company, product, production, date);
    MessageBox.Show($"바코드 오류 :{sum}");
    userControl_Barcode.DataGridView1.DataSource = db.dt;
}
```

- DB 클래스를 생성 후 DB에 정보를 입력

바코드 – DB클래스

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace openCV0820
{
    참조 2개
    internal class DBConnect
    {
        string Connecting = "server=172.31.6.28,1433;" +
            $"database=ProductList;" +
                $"uid=sa;" +
                $"pwd=12345;";

        SqlConnection conn;
        SqlDataAdapter dataAdapter;
        DataSet dataSet;
        public DataTable dt;
        public string msg = "";//메세지
        public string dbresult = "";//디비 결과

        참조 1개
        public void connect()
        {
            try
            {
                conn = new SqlConnection(Connecting);
                conn.Open();
                if (conn.State == ConnectionState.Open) msg = "연결";
                else msg = "실패";
            }
            catch (Exception ex) { }
            finally
            {
                conn.Close();
            }

        }
```

- DB 연결 및 연결 테스트
- 비연결형 방식으로 구현
  (SqlConnection, DataAdapter, DataSet으로 구현)
  (연결형은 SqlConnection, SqlCommand, SqlDataReader으로 구성)
- 비연결형은 DataSet을 활용하여 오프라인 DB를 생성하여 사용
  (여러개의 테이블 조작 가능)
- DataAdapter를 사용하여 DB를 조작
- Connection를 사용하여 연결 후 종료

바코드 – INSERT문

```
public void insert(string company, string product, string place, DateTime yearMonth)
{
    try
    {
        dataAdapter = new SqlDataAdapter();
        dt = new DataTable();
        dataSet = new DataSet();
        if (conn.State != ConnectionState.Open) conn.Open();
        string queryString = "INSERT INTO Table_2 (company, product, place, yearMonth)" +
            "VALUES (@company, @product, @place, @yearMonth)";

        dataAdapter.InsertCommand = new SqlCommand(queryString, conn);
        dataAdapter.InsertCommand.Parameters.AddWithValue("@company", company);
        dataAdapter.InsertCommand.Parameters.AddWithValue("@product", product);
        dataAdapter.InsertCommand.Parameters.AddWithValue("@place", place);
        dataAdapter.InsertCommand.Parameters.AddWithValue("@yearMonth", yearMonth);

        dataAdapter.InsertCommand.ExecuteNonQuery();

        string selectQuery = "SELECT * FROM Table_2";
        dataAdapter.SelectCommand = new SqlCommand(selectQuery, conn);

        dataSet.Clear();
        dataAdapter.Fill(dataSet, "Table_2");
        dt = dataSet.Tables["Table_2"];
    }
    catch (Exception ex)
    {
        msg = $"{ex.Message} : 입력 부분";
    }
    finally
    {
        conn.Close();
        dataAdapter.InsertCommand.Dispose();
    }
}
```

- SQL 쿼리문을 사용하여 데이터를 입력
- Connection -> DataAdapter -> DataSet -> Connection 종료 순

바코드 – SELECT문

```
public void select(string company)
{
    try
    {
        dataAdapter = new SqlDataAdapter();
        dt = new DataTable();
        dataSet = new DataSet();

        conn.Open();

        string queryString = "select * from Table_2 where company = @company";

        dataAdapter.SelectCommand = new SqlCommand(queryString, conn);
        dataAdapter.SelectCommand.Parameters.AddWithValue("@company", company);

        dataSet.Clear();
        dataAdapter.Fill(dataSet, "Table_2");
        dt = dataSet.Tables["Table_2"];

    }
    catch (Exception ex)
    {
        msg = $"{ex.Message} : 조회 부분";
    }
    finally
    {
        conn.Close();
        dataAdapter.SelectCommand.Dispose();
    }

}
private void UserControl_Barcode_event_UC_select_Click()
{
    string name = userControl_Barcode.TextBox1.Text;
    db.select(name);
    userControl_Barcode.DataGridView1.DataSource = db.dt;
}
```

- 회사명을 검색하는 SELECT문

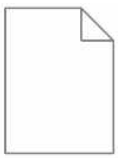
5. 얼굴인식

영상이나 이미지에서 얼굴을 검출하여 카메라 앱 필터와 같이 이미지 편집

- Haar Classifier Cascade 사용
- 사용한 이미지와 검출기 xml 파일


baldHead


girl_hair


haarcascade_frontalface_alt.xml


rabbit_ear


sunglasses

얼굴 인식 – 얼굴 찾기(기본)

```
public Mat faceDecter(Mat src)
{
    using (haarface = new Mat())
    {
        Cv2.CopyTo(src, haarface);
        if (haarface.Channels() != 1) Cv2.CvtColor(haarface, haarface, ColorConversionCodes.RGB2GRAY);
        Cv2.EqualizeHist(haarface, haarface);
        using (faceCascade = new CascadeClassifier(filePath))
        {
            OpenCvSharp.Rect[] faces = faceCascade.DetectMultiScale(haarface, scaleFactor: 1.139, minNeighbors: 3);
            foreach (var face in faces)
            {
                Cv2.Rectangle(src, face, new Scalar(0, 0, 255), 2);
            }
        }
    }
    return src;
```

- 얼굴을 찾아 표시해주는 메소드
- 단순히 얼굴만 찾는 기능

얼굴 인식 – 이미지 전처리

```
public Mat faceDecoPreprocessing(Mat src, string path)
{
    decoImage = Cv2.ImRead(path, ImreadModes.Unchanged);//알파값을 받기 위해 unchanged
    Mat decoration = src.Clone();// src 이미지를 복사하여 decoration을 만듬
    if (decoration.Channels() != 1)// 이미지를 회색조로 변환
    {
        Cv2.CvtColor(decoration, decoration, ColorConversionCodes.BGR2GRAY);
    }
    Cv2.EqualizeHist(decoration, decoration);// 히스토그램 평활화
    return decoration;
}
```

- 얼굴을 꾸미기 전 이미지를 활용하기 위해 전처리해주는 메소드
- 알파값이 존재하는 PNG파일을 받아 그레이스케일화 해주고 평활화 실행
  (히스토그램 평활화 : 명암 값의 분포가 치우친 영상을 고르게 분포시켜주는 방법
  명암 대비를 증가시켜 인지도를 높이고 화질을 개선할 때 사용)

얼굴 인식 – 얼굴 꾸미기

```
public Mat faceDeco(Mat src, string path, double faceWidth, double faceHeight, double FX, double FY, bool up)
{
    //인수 원본, 전경 이미지 위치, 전경 가로, 전경 세로, 전경 시작좌표 X, Y, 머리 위?
    decoration = faceDecoPreprocessing(src, path);//이미지 전처리
    using (var faceCascade = new CascadeClassifier(filePath))// 얼굴 탐지기 로드
    {
        OpenCvSharp.Rect[] faces = faceCascade.DetectMultiScale(decoration, scaleFactor: 1.139, minNeighbors: 2);
        foreach (var face in faces)
        {
            //인수(원본, 전경, 가로, 세로, 시작좌표 x, y, 머리 위?)
            AlphaBlending(src, face, faceWidth, faceHeight, FX, FY, up);//알파블랜딩
        }
    }
    return src;
}
```

- 얼굴을 꾸미는 메소드 본문
- 이미지를 전처리 한 뒤 얼굴을 검출해 전경 이미지의 알파값을 이용해 배경 이미지와 합성
- 특정 색상을 (주로 초록색, 크로마키)제거하여 블렌딩할 수도 있음

얼굴 인식 – 알파블렌딩

```
public void AlphaBlending(Mat src, OpenCvSharp.Rect face,
    double faceWidth, double faceHeight, double FX, double FY, bool up)
{
    Mat resizedEar = new Mat();//머리
    Cv2.Resize(decoImage, resizedEar, new OpenCvSharp.Size(
        (int)face.Width * faceWidth, (int)face.Height * faceHeight));//머리길이
    //위치 조정
    int earX1 = face.X - (int)(face.Width * FX);//왼
    int earY1;//위
    if (up) earY1 = face.Y - (int)(face.Height * FY);
    else earY1 = face.Y + (int)(face.Height * FY);
    int earX2 = earX1 + resizedEar.Cols;//오
    int earY2 = earY1 + resizedEar.Rows;//하
    // 이미지 경계 조정
    earX1 = Math.Max(0, earX1);//0보다 작을경우 0
    earY1 = Math.Max(0, earY1);
    earX2 = Math.Min(src.Cols, earX2);//src보다 클경우 earX2
    earY2 = Math.Min(src.Rows, earY2);
    //이미지 합성
    for (int y = earY1; y < earY2; y++)
    {
        for (int x = earX1; x < earX2; x++)
        {
            // 인덱스 범위 체크
            if (y - earY1 < resizedEar.Rows && x - earX1 < resizedEar.Cols)
            {
                //Vec4b earPixel: resizedEar 이미지에서 현재 픽셀의 RGBA 값
                //Vec4b는 4개의 바이트(각각 Red, Green, Blue, Alpha)를 포함
                //위치에 있는 픽셀의 값을 가져옴
                Vec4b earPixel = resizedEar.At<Vec4b>(y - earY1, x - earX1);
                if (earPixel[3] > 0) // 알파 값이 0이 아닌 픽셀만 처리
                {
                    Vec3b imgPixel = src.At<Vec3b>(y, x);
                    // 알파 블렌딩
                    // 두 개 이상의 이미지를 결합할 때 사용하는 기술로 -
                    // - 각 이미지의 투명도(알파 값)를 고려하여 합성하는 과정
                    //이미지의 각 픽셀에서 색상과 투명도를 조절하여 최종 이미지를 생성
                    //수식 : 출력 색상=(배경 색상×(255-알파)+전경 색상×알파)/255
                    //*전경 : 합성할 이미지
                    src.At<Vec3b>(y, x) = new Vec3b(
                        (byte)((imgPixel.Item0 * (255 - earPixel[3]) + earPixel[0] * earPixel[3]) / 255),
                        (byte)((imgPixel.Item1 * (255 - earPixel[3]) + earPixel[1] * earPixel[3]) / 255),
                        (byte)((imgPixel.Item2 * (255 - earPixel[3]) + earPixel[2] * earPixel[3]) / 255)
                    );
                }
            }
        }
    }
    if (resizedEar != null) resizedEar.Dispose();//메모리 해제
}
```

4 총평

4-1. 작품 후기

1) OpenCV와 Cognex 비전 라이브러리의 사용법과 검출 방법을 이해할 수 있었다.
2) 이미지를 활용하기 위한 전처리 과정을 이해할 수 있었다.
3) OCR과 Haar Classifier Cascade을 사용하며 다양한 응용 방식을 알 수 있었다.
4) 바코드 정보를 DB에 저장하는 과정을 통해 데이터베이스에 대해 이해할 수 있었다.

4-2. 개선 방안

1) 문제점
- 영상 파일만 되는 기능이 있고, 이미지 파일만 되는 기능이 있다.
- 동전의 경우 높이가 달라지면 제대로 분류하지 못하고 다른 동전을 검출해내지 못한다.
- 색상 검출의 경우 난수 100개를 발생시켜 검출하기 때문에 정확도가 떨어진다.

2) 해결 방안
- flag변수를 활용하여 만든 기능을 사용시 timer_tick 메소드에서 카메라나 영상에서 이미지를 30fps마다 출력할 때 변환 클래스를 거쳐 출력한다.
- 높이는 고정한다. 혹은 높이를 알 수 있는 요인을 추가한다.
  윤곽선 검출과 색상 검출을 추가하여 동전마다 특징을 이용하여 동전을 분류한다.
- 더 많은 난수를 발생시키거나 흐림 효과(블러) 정도를 높인다.

4-3. 아쉬웠던 점

- 포토샵처럼 레이어를 나눠 이미지 변환이나 필터를 추가하고 제거하는 기능을 만들지 못해 아쉬웠다.
- Vision Pro가 없어서 사용해보지 못하고 스마트 카메라를 사용한 것이 아쉬웠다.

깃허브 주소 : https://github.com/io-sh/OpenCV_VisionCraft