

졸업작품 최종보고서

영상처리를 통한 악보인식 및 재생

지도교수
유준범 교수님

200611480 송명서
200611468 박상원

목차

1. 선정배경	3
- 필요기술	3
2. 개발목표	3
1) 전처리 과정.....	3
2) 본처리 과정.....	4
3) 배경이론	5
3. 아키텍처	7
1) CvvImage	8
2) MusicDlg::OnBnClickedFinal	15
3) MusicDlg::OnBnClickedPlay	22
4. 음표인식	31
1) 음표 종류별 분석	31
2) 음표분석 절차.....	33
3) 악보처리	40
4) 악보재생	44
5. 개발환경	45
6. UI 및 기능.....	46
7. 실행.....	47
1) 전처리 과정.....	47
2) 본처리 과정.....	50
3) 모폴로지	52
8. 일정.....	55
9. 참고문헌	56

1. 선정배경



영상처리를 통해서 기존의 이미지를 쉽고 빠르게 데이터화 할 수 있다.

악보라는 형식을 갖춘 이미지를 영상처리 해서 분류함으로써 특수한 조건에서 데이터를 다룰 수 있다.

영상처리 된 악보데이터를 음악데이터로 재생함으로써 데이터를 실제 음악처럼 효과적으로 사용할 수 있다.

- 필요기술

최적 이진화	기호인식	모폴로지 연산
소절단위 분할	음표분석	
소절 세분화 허프변환	음표연주	
악보의 오선영역을 검출		

2. 개발목표

다양한 악보 이미지들을 영상처리를 통해서 인식해 데이터화 한다.

오선의 다양한 음표들을 인식해서 음표의 음, 음표의 길이 등을 구분할 수 있다.


데이터화 한 악보의 데이터를 처리 해서 실제 들을 수 있는 음악으로 재생한다.

1) 전처리 과정

- 우선 종이에 인쇄된, 혹은 손으로 그려진 악보를 컴퓨터 프로그램이 분석하기에 알맞도록 변환하는 과정을 담았습니다.

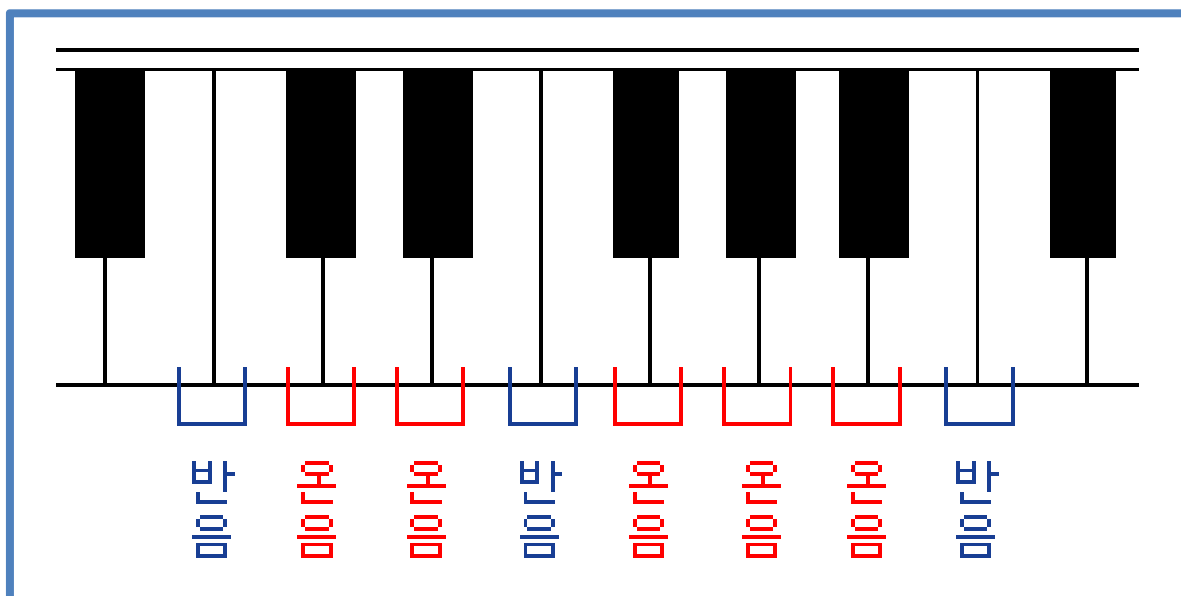
컬러이미지를 명암화 이미지로 전환	흑백으로 된 사물이라도 카메라로 사진을 찍으면 컬러 이미지가 생성되므로 이후에 필요한 이미지 분석을 편리하게 하기 위해 흑백 이미지로 전환
흑백이미지의 이진화 과정	흑백으로 전환된 이미지에서 나아가 어느 특정 threshold를 기준으로 흑, 백 두가지 이미지를 만들어 배경과 오선, 음표 부분을 구분이 확실하게 되도록 만들어 줍니다.
소절 단위 분할	세부적 분석이 가능하도록 소절단위를 분할합니다. 수평 히스토그램의 최대값을 기준으로 나누어 하나의 소절을 가지고 다음 연산을 수행할 수 있도록 만듭니다.

2) 본처리 과정

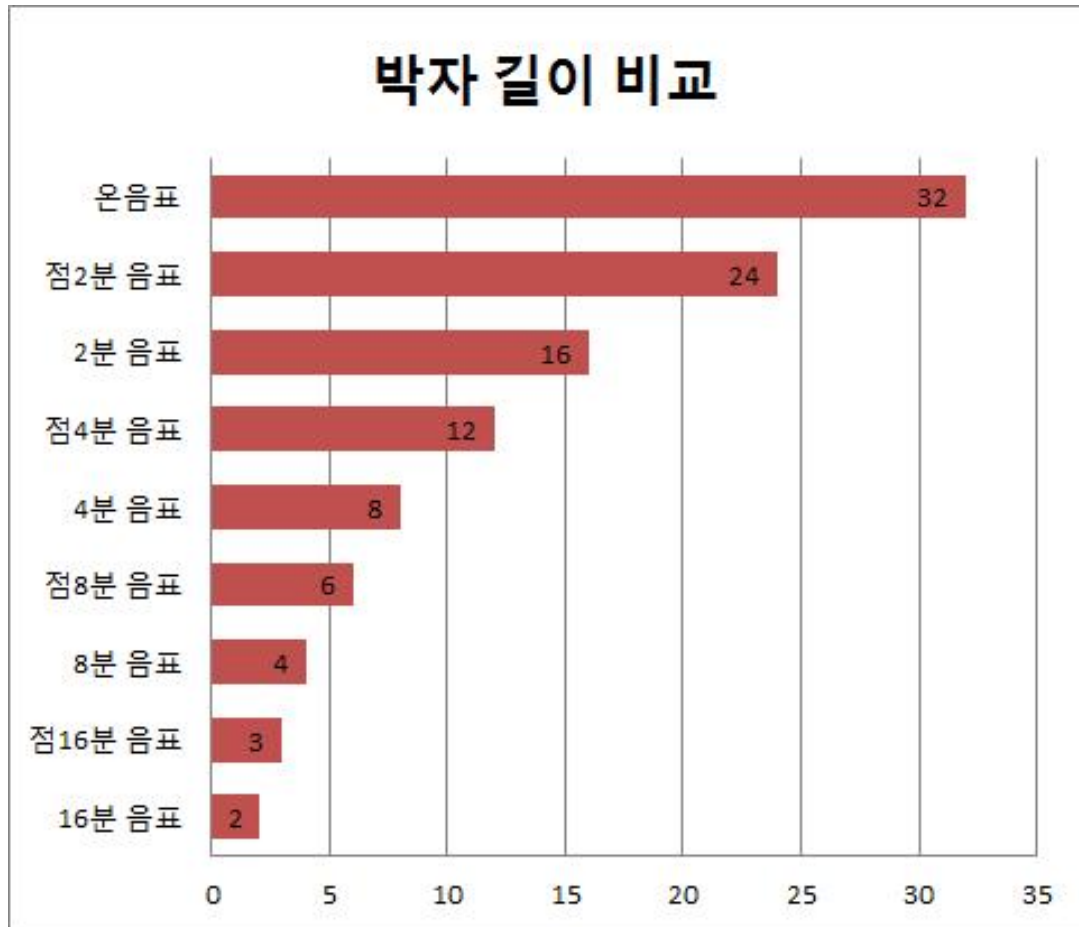
마디 단위 분할 (더욱더 세분화 할 수도 있음)	촬영된 이미지에서 우선 오선을 분석합니다. 이 경우 오선이 수평으로 곧은 직선이 아닐 가능성이 높아 더욱더 세분화 과정을 거침으로써 직선에 가까운 오선을 찾게 됩니다.
Hough Transform	직선을 찾는 알고리즘입니다. 앞에서 행해진 과정을 통해 얻어진 이미지에서 직선이라고 생겨진 부분을 검출합니다.
Hough Transform 보완 과정	허프 변환을 거친다 하더라도 검출 되지 않은 오선을 구하기 위해 추가적인 연산을 합니다. 분할된 이미지끼리 합하고 검출된 직선들끼리의 분석을 통해 검출되지 않았거나 실제와 차이가 있는 오선을 재수정해 찾아 냅니다.
오선제거	다시 수직성분의 차이를 이용해 오선만 있는 부분과 음표 등이 겹친 부분을 구분하고 나아가 오선을 제거합니다.
음표 분석	음표의 머리, 줄기, 꼬리, 점 등을 분석해 음표의 박자를 인식합니다. 
음표 위치 분석	음표의 위치를 분석함으로써 해당되는 음표

	마다 특정한 계이름을 부여 할 수 있습니다.
기타 기호 분석	#, ♯, b 등의 기호들을 분석합니다. 이를 통해 여러 가지 악보의 특성을 파악합니다.
멜로디 출력	음표 위치에 따른 악기 소리 출력

3) 배경이론

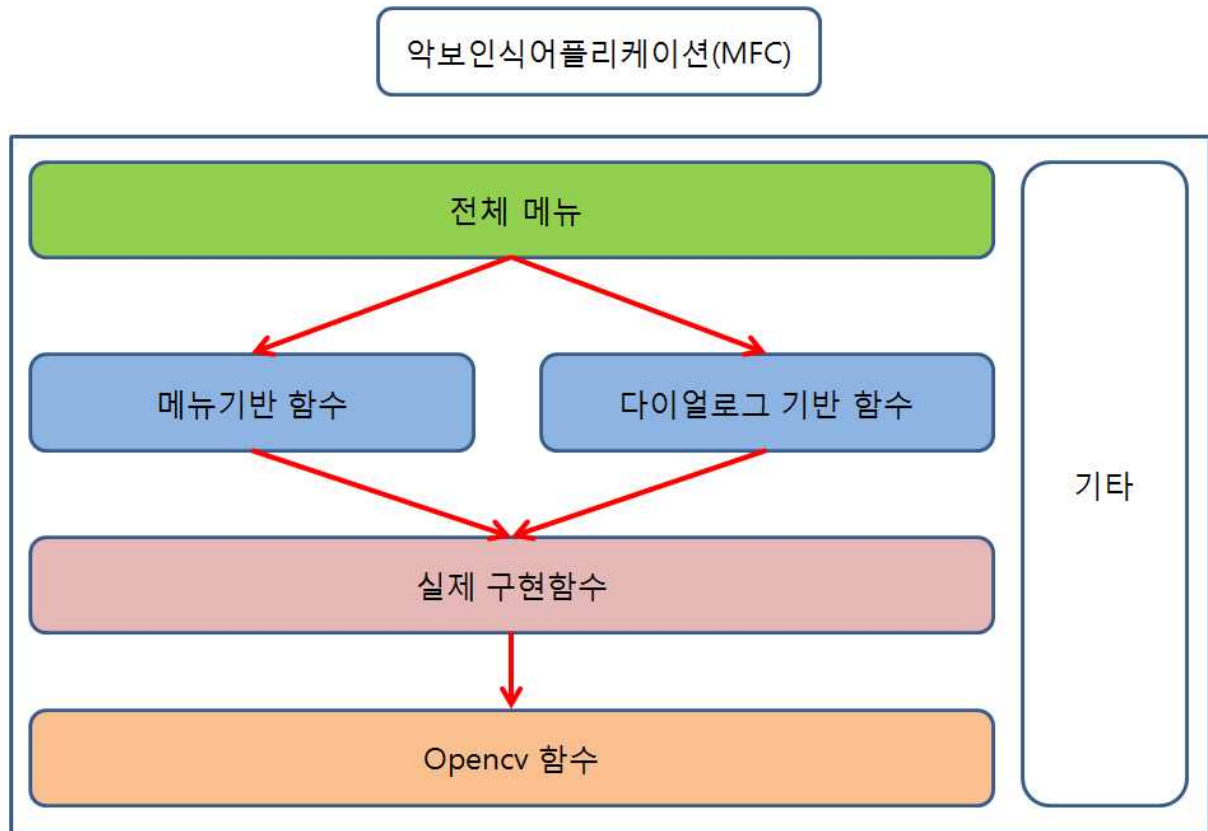


- 정확한 음정처리를 위해서 반음과 온음의 구분처리가 필요하다.



- 4분음표 8박자 기준의 박자길이 설정.

3. 아키텍처



전체메뉴는 사용자에게 보이는 MFC UI화면입니다.

화면의 UI에는 악보를 불러왔을 때 악보가 보여지는 화면과 전처리 과정, 본처리 과정, 모폴로지, 파일 재생 등의 버튼이 있습니다.

각 버튼 등을 실행했을 때는 메뉴기반 함수와 다이얼로그 기반 함수가 실행됩니다.

다이얼로그 기반 함수는 악보파일 불러오기와 같은 작업이며, 실행하면 해당 다이얼로그가 생성됩니다.

그리고 전처리 과정, 본처리 과정, 모폴로지와 같은 작업은 메뉴기반 함수이며, 실행하면 해당 작업을 수행합니다.

이렇게 UI기반 함수들은 실제구현 함수를 로드합니다.

실제구현 함수는 실질적인 영상처리 기능을 수행합니다. 그리고 이 함수들은 OpenCV 라이브러리내의 여러 API를 사용합니다.

이런 구조의 아키텍처를 통해서 OpenCV를 이용한 영상처리 프로그램을 개발합니다.

1) CvtColor

```
#pragma once
#include "stdafx.h"
#include "CvtColor.h"
////////////////////////////////////
// Construction/Destruction
////////////////////////////////////
CV_INLINE RECT NormalizeRect( RECT r );
CV_INLINE RECT NormalizeRect( RECT r )
{
    int t;
    if( r.left > r.right )
    {
        t = r.left;
        r.left = r.right;
        r.right = t;
    }
    if( r.top > r.bottom )
    {
        t = r.top;
        r.top = r.bottom;
        r.bottom = t;
    }

    return r;
}
CV_INLINE CvRect RectToCvRect( RECT sr );
CV_INLINE CvRect RectToCvRect( RECT sr )
{
    sr = NormalizeRect( sr );
    return cvRect( sr.left, sr.top, sr.right - sr.left, sr.bottom - sr.top );
}
CV_INLINE RECT CvRectToRect( CvRect sr );
CV_INLINE RECT CvRectToRect( CvRect sr )
{
    RECT dr;
    dr.left = sr.x;
    dr.top = sr.y;
```



```

        dr.right = sr.x + sr.width;
        dr.bottom = sr.y + sr.height;

        return dr;
    }
CV_INLINE IplROI RectToROI( RECT r );
CV_INLINE IplROI RectToROI( RECT r )
{
    IplROI roi;
    r = NormalizeRect( r );
    roi.xOffset = r.left;
    roi.yOffset = r.top;
    roi.width = r.right - r.left;
    roi.height = r.bottom - r.top;
    roi.coi = 0;

    return roi;
}
void FillBitmapInfo( BITMAPINFO* bmi, int width, int height, int bpp, int origin )
{
    assert( bmi && width >= 0 && height >= 0 && (bpp == 8 || bpp == 24 || bpp == 32));

    BITMAPINFOHEADER* bmih = &(bmi->bmiHeader);

    memset( bmih, 0, sizeof(*bmih));
    bmih->biSize = sizeof(BITMAPINFOHEADER);
    bmih->biWidth = width;
    bmih->biHeight = origin ? abs(height) : -abs(height);
    bmih->biPlanes = 1;
    bmih->biBitCount = (unsigned short)bpp;
    bmih->biCompression = BI_RGB;
    if( bpp == 8 )
    {
        RGBQUAD* palette = bmi->bmiColors;
        int i;
        for( i = 0; i < 256; i++ )
        {
            palette[i].rgbBlue = palette[i].rgbGreen = palette[i].rgbRed = (BYTE)i;
            palette[i].rgbReserved = 0;
        }
    }
}

```

```

        }
    }
}

CvImage::CvImage()
{
    m_img = 0;
}

void CvImage::Destroy()
{
    cvReleaseImage( &m_img );
}

CvImage::~CvImage()
{
    Destroy();
}

bool CvImage::Create( int w, int h, int bpp, int origin )
{
    const unsigned max_img_size = 10000;

    if( (bpp != 8 && bpp != 24 && bpp != 32) ||
        (unsigned)w >= max_img_size || (unsigned)h >= max_img_size ||
        (origin != IPL_ORIGIN_TL && origin != IPL_ORIGIN_BL))
    {
        assert(0); // most probably, it is a programming error
        return false;
    }

    if( !m_img || Bpp() != bpp || m_img->width != w || m_img->height != h )
    {
        if( m_img && m_img->nSize == sizeof(IplImage))
            Destroy();
        /* prepare IPL header */
        m_img = cvCreateImage( cvSize( w, h ), IPL_DEPTH_8U, bpp/8 );
    }

    if( m_img )
        m_img->origin = origin == 0 ? IPL_ORIGIN_TL : IPL_ORIGIN_BL;
    return m_img != 0;
}

void CvImage::CopyOf( CvImage& image, int desired_color )
{

```



```

if( r.width < 0 || r.height < 0 ) return false;

IplImage* img = cvLoadImage( filename, desired_color );
if( !img )
    return false;
if( r.width == 0 || r.height == 0 )
{
    r.width = img->width;
    r.height = img->height;
    r.x = r.y = 0;
}
if( r.x > img->width || r.y > img->height ||
    r.x + r.width < 0 || r.y + r.height < 0 )
{
    cvReleaseImage( &img );
    return false;
}
/* truncate r to source image */
if( r.x < 0 )
{
    r.width += r.x;
    r.x = 0;
}
if( r.y < 0 )
{
    r.height += r.y;
    r.y = 0;
}
if( r.x + r.width > img->width )
    r.width = img->width - r.x;

if( r.y + r.height > img->height )
    r.height = img->height - r.y;
cvSetImageROI( img, r );
CopyOf( img, desired_color );
cvReleaseImage( &img );
return true;
}

bool CwImage::Save( const char* filename )

```

```

{
    if( !m_img )
        return false;
    cvSaveImage( filename, m_img );
    return true;
}

void CwImage::Show( const char* window )
{
    if( m_img )
        cvShowImage( window, m_img );
}

void CwImage::Show( HDC dc, int x, int y, int w, int h, int from_x, int from_y )
{
    if( m_img && m_img->depth == IPL_DEPTH_8U )
    {
        uchar buffer[sizeof(BITMAPINFOHEADER) + 1024];
        BITMAPINFO* bmi = (BITMAPINFO*)buffer;
        int bmp_w = m_img->width, bmp_h = m_img->height;
        FillBitmapInfo( bmi, bmp_w, bmp_h, Bpp(), m_img->origin );
        from_x = MIN( MAX( from_x, 0 ), bmp_w - 1 );
        from_y = MIN( MAX( from_y, 0 ), bmp_h - 1 );
        int sw = MAX( MIN( bmp_w - from_x, w ), 0 );
        int sh = MAX( MIN( bmp_h - from_y, h ), 0 );
        SetDIBitsToDevice(
            dc, x, y, sw, sh, from_x, from_y, from_y, sh,
            m_img->imageData + from_y*m_img->widthStep,
            bmi, DIB_RGB_COLORS );
    }
}

void CwImage::DrawToHDC( HDC hDCDst, RECT* pDstRect )
{
    if( pDstRect && m_img && m_img->depth == IPL_DEPTH_8U && m_img->imageData )
    {
        uchar buffer[sizeof(BITMAPINFOHEADER) + 1024];
        BITMAPINFO* bmi = (BITMAPINFO*)buffer;
        int bmp_w = m_img->width, bmp_h = m_img->height;
        CvRect roi = cvGetImageROI( m_img );
        CvRect dst = RectToCvRect( *pDstRect );
        if( roi.width == dst.width && roi.height == dst.height )
    }

```

```

    {
        Show( hDCDst, dst.x, dst.y, dst.width, dst.height, roi.x, roi.y );
        return;
    }
    if( roi.width > dst.width )
    {
        SetStretchBltMode(
            hDCDst,          // handle to device context
            HALFTONE );
    }
    else
    {
        SetStretchBltMode(
            hDCDst,          // handle to device context
            COLORONCOLOR );
    }
    FillBitmapInfo( bmi, bmp_w, bmp_h, Bpp(), m_img->origin );
    ::StretchDIBits(
        hDCDst,
        dst.x, dst.y, dst.width, dst.height,
        roi.x, roi.y, roi.width, roi.height,
        m_img->imageData, bmi, DIB_RGB_COLORS, SRCCOPY );
}

}

void CwImage::Fill( int color )
{
    cvSet( m_img, cvScalar(color&255,(color>>8)&255,(color>>16)&255,(color>>24)&255) );
}

```

2) MusicDlg::OnBnClickedFinal

```
void CMusicDlg::OnBnClickedFinal()
{
    // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
    CPreprocessing cPreprocessing;
    CProcessing cProcessing;
    CExtra cExtra;
    CSymbol cSymbol;
    CRecognition cRecognition;
    CFinalStep cFinalStep;

    IplImage *gray_image = cPreprocessing.PRE_whiteblack(m_pImage);
    IplImage *binary_image = cPreprocessing.PRE_binary(gray_image);

    int a1=0, a2=0, a3=0, a4=0, a5=0, a6=0, a0=0;
    IplImage *hline = cvCloneImage(m_pImage);
    cPreprocessing.PRE_Hline(m_pImage, &hline, &a1, &a2, &a3, &a4, &a5, &a6, &a0);

    IplImage *section_6,*section_5,*section_4,*section_3,*section_2,*section_1;
    cPreprocessing.PRE_Segments(gray_image,&section_6,&section_5,&section_4,&section_3,&
section_2,&section_1,
                                a1,a2,a3,a4,a5,a6,a0);

    if(a0 == 6)
    {
        IplImage *Estaffline = cvCreateImage( cvGetSize(section_6), IPL_DEPTH_8U, 1);
        cvSet(Estaffline, CV_RGB(255,255,255), NULL);
        cProcessing.PRO_Hough(section_6, &Estaffline);    // 함수 안에서 이진화, 캐니
연산등이 실행된다.

        int type = CV_SHAPE_RECT;
        IplImage *temp = cvCreateImage( cvGetSize(section_6), IPL_DEPTH_8U, 3 );
        cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
        IplImage *opening_image = cExtra.EX_opening(temp, type, NULL ); // 열림연산
(모폴리지는 칼라만)

        IplImage *detection_image = cvCloneImage(section_6);
        int xStart=0, xEnd=0, yStart=0, yEnd=0;
```

```

        cProcessing.PRO_Detection(opening_image, &detection_image, &xStart, &xEnd,
&yStart, &yEnd);

        IplImage *guessing_image = cvCreateImage( cvGetSize(section_6), IPL_DEPTH_8U,
3 );

        cvCvtColor(section_6, guessing_image, CV_GRAY2BGR );
        IplImage *line_image = cvCreateImage( cvGetSize(section_6), IPL_DEPTH_8U, 3 );
        cvSet(line_image, CV_RGB(255,255,255), NULL);
        cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image,  xStart,
xEnd, yStart, yEnd);

        cFinalStep.Final_newsymana(line_image, guessing_image, &guessing_image, a0, 0);

//        DrawImage(guessing_image);

        cvReleaseImage(&Estaffline);
        cvReleaseImage(&temp);
        cvReleaseImage(&opening_image);
        cvReleaseImage(&detection_image);
        cvReleaseImage(&guessing_image);
        cvReleaseImage(&line_image);
        cvReleaseImage(&section_6);

        --a0;
    }
    if(a0 == 5)
    {
        IplImage *Estaffline = cvCreateImage( cvGetSize(section_5), IPL_DEPTH_8U, 1);
        cvSet(Estaffline, CV_RGB(255,255,255), NULL);
        cProcessing.PRO_Hough(section_5, &Estaffline);    // 함수 안에서 이진화, 캐니
연산등이 실행된다.

        int type = CV_SHAPE_RECT;
        IplImage *temp = cvCreateImage( cvGetSize(section_5), IPL_DEPTH_8U, 3 );
        cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
        IplImage *opening_image = cExtra.EX_opening(temp, type, NULL ); // 열림연산
(모폴리지는 칼라만)

        IplImage *detection_image = cvCloneImage(section_5);

```



```

int xStart=0, xEnd=0, yStart=0, yEnd=0;
cProcessing.PRO_Detection(opening_image, &detection_image, &xStart, &xEnd,
&yStart, &yEnd);

IplImage *guessing_image = cvCreateImage( cvGetSize(section_5), IPL_DEPTH_8U,
3 );

cvCvtColor(section_5, guessing_image, CV_GRAY2BGR );
IplImage *line_image = cvCreateImage( cvGetSize(section_5), IPL_DEPTH_8U, 3 );
cvSet(line_image, CV_RGB(255,255,255), NULL);
cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image, xStart,
xEnd, yStart, yEnd);

cFinalStep.Final_newsymana(line_image, guessing_image, &guessing_image, a0, 0);

// DrawImage(guessing_image);

cvReleaseImage(&Estaffline);
cvReleaseImage(&temp);
cvReleaseImage(&opening_image);
cvReleaseImage(&detection_image);
cvReleaseImage(&guessing_image);
cvReleaseImage(&line_image);
cvReleaseImage(&section_5);

--a0;
}
if(a0 == 4)
{
IplImage *Estaffline = cvCreateImage( cvGetSize(section_4), IPL_DEPTH_8U, 1);
cvSet(Estaffline, CV_RGB(255,255,255), NULL);
cProcessing.PRO_Hough(section_4, &Estaffline); // 함수 안에서 이진화, 캐니
연산등이 실행된다.

int type = CV_SHAPE_RECT;
IplImage *temp = cvCreateImage( cvGetSize(section_4), IPL_DEPTH_8U, 3 );
cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
IplImage *opening_image = cExtra.EX_opening(temp, type, NULL ); // 열림연산
(모폴리지는 칼라만)

```

```

IplImage *detection_image = cvCloneImage(section_4);
int xStart=0, xEnd=0, yStart=0, yEnd=0;
cProcessing.PRO_Detection(opening_image, &detection_image, &xStart, &xEnd,
&yStart, &yEnd);

IplImage *guessing_image = cvCreateImage( cvGetSize(section_4), IPL_DEPTH_8U,
3 );

cvCvtColor(section_4, guessing_image, CV_GRAY2BGR );
IplImage *line_image = cvCreateImage( cvGetSize(section_4), IPL_DEPTH_8U, 3 );
cvSet(line_image, CV_RGB(255,255,255), NULL);
cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image, xStart,
xEnd, yStart, yEnd);

cFinalStep.Final_newsymanana(line_image, guessing_image, &guessing_image, a0, 0);

// DrawImage(guessing_image);

cvReleaseImage(&Estaffline);
cvReleaseImage(&temp);
cvReleaseImage(&opening_image);
cvReleaseImage(&detection_image);
cvReleaseImage(&guessing_image);
cvReleaseImage(&line_image);
cvReleaseImage(&section_4);

--a0;
}
if(a0 == 3)
{
IplImage *Estaffline = cvCreateImage( cvGetSize(section_3), IPL_DEPTH_8U, 1);
cvSet(Estaffline, CV_RGB(255,255,255), NULL);
cProcessing.PRO_Hough(section_3, &Estaffline);    // 함수 안에서 이진화, 캐니
연산등이 실행된다.

int type = CV_SHAPE_RECT;
IplImage *temp = cvCreateImage( cvGetSize(section_3), IPL_DEPTH_8U, 3 );
cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
IplImage *opening_image = cExtra.EX_opening(temp, type, NULL ); // 열림연산
(모폴리지는 칼라만)

```

```

        IplImage *detection_image = cvCloneImage(section_3);
        int xStart=0, xEnd=0, yStart=0, yEnd=0;
        cProcessing.PRO_Detection(opening_image, &detection_image, &xStart, &xEnd,
&yStart, &yEnd);

        IplImage *guessing_image = cvCreateImage( cvGetSize(section_3), IPL_DEPTH_8U,
3 );

        cvCvtColor(section_3, guessing_image, CV_GRAY2BGR );
        IplImage *line_image = cvCreateImage( cvGetSize(section_3), IPL_DEPTH_8U, 3 );
        cvSet(line_image, CV_RGB(255,255,255), NULL);
        cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image, xStart,
xEnd, yStart, yEnd);

        cFinalStep.Final_newsymanana(line_image, guessing_image, &guessing_image, a0, 0);

//        DrawImage(guessing_image);

        cvReleaseImage(&Estaffline);
        cvReleaseImage(&temp);
        cvReleaseImage(&opening_image);
        cvReleaseImage(&detection_image);
        cvReleaseImage(&guessing_image);
        cvReleaseImage(&line_image);
        cvReleaseImage(&section_3);

        --a0;
    }
    if(a0 == 2)
    {
        IplImage *Estaffline = cvCreateImage( cvGetSize(section_2), IPL_DEPTH_8U, 1);
        cvSet(Estaffline, CV_RGB(255,255,255), NULL);
        cProcessing.PRO_Hough(section_2, &Estaffline);        // 함수 안에서 이진화, 캐니
연산등이 실행된다.

```

```

        int type = CV_SHAPE_RECT;
        IplImage *temp = cvCreateImage( cvGetSize(section_2), IPL_DEPTH_8U, 3 );
        cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
        IplImage *opening_image = cExtra.EX_opening(temp, type, NULL ); // 열림연산

```

(모폴리지는 칼라만)

```
    IplImage *detection_image = cvCloneImage(section_2);
    int xStart=0, xEnd=0, yStart=0, yEnd=0;
    cProcessing.PRO_Detection(opening_image, &detection_image, &xStart, &xEnd,
&yStart, &yEnd);

    IplImage *guessing_image = cvCreateImage( cvGetSize(section_2), IPL_DEPTH_8U,
3 );

    cvCvtColor(section_2, guessing_image, CV_GRAY2BGR );
    IplImage *line_image = cvCreateImage( cvGetSize(section_2), IPL_DEPTH_8U, 3 );
    cvSet(line_image, CV_RGB(255,255,255), NULL);
    cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image,  xStart,
xEnd, yStart, yEnd);

    cFinalStep.Final_newsymana(line_image, guessing_image, &guessing_image, a0, 0);

//      DrawImage(guessing_image);

    cvReleaseImage(&Estaffline);
    cvReleaseImage(&temp);
    cvReleaseImage(&opening_image);
    cvReleaseImage(&detection_image);
    cvReleaseImage(&guessing_image);
    cvReleaseImage(&line_image);
    cvReleaseImage(&section_2);

    --a0;
}
if(a0 == 1)
{
    IplImage *Estaffline = cvCreateImage( cvGetSize(section_1), IPL_DEPTH_8U, 1);
    cvSet(Estaffline, CV_RGB(255,255,255), NULL);
    cProcessing.PRO_Hough(section_1, &Estaffline);    // 함수 안에서 이진화, 캐니
연산등이 실행된다.

    int type = CV_SHAPE_RECT;
    IplImage *temp = cvCreateImage( cvGetSize(section_1), IPL_DEPTH_8U, 3 );
    cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
```

```
        IplImage *opening_image = cExtra.EX_opening(temp, type, NULL ); // 열림연산  
(모폴리지는 칼라만)
```

```
        IplImage *detection_image = cvCloneImage(section_1);  
        int xStart=0, xEnd=0, yStart=0, yEnd=0;  
        cProcessing.PRO_Detection(opening_image, &detection_image, &xStart, &xEnd,  
&yStart, &yEnd);
```

```
        IplImage *guessing_image = cvCreateImage( cvGetSize(section_1), IPL_DEPTH_8U,  
3 );
```

```
        cvCvtColor(section_1, guessing_image, CV_GRAY2BGR );  
        IplImage *line_image = cvCreateImage( cvGetSize(section_1), IPL_DEPTH_8U, 3 );  
        cvSet(line_image, CV_RGB(255,255,255), NULL);  
        cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image, xStart,  
xEnd, yStart, yEnd);
```

```
        cFinalStep.Final_newsymana(line_image, guessing_image, &guessing_image, a0, 0);
```

```
//        DrawImage(guessing_image);
```

```
        cvReleaseImage(&Estaffline);  
        cvReleaseImage(&temp);  
        cvReleaseImage(&opening_image);  
        cvReleaseImage(&detection_image);  
        cvReleaseImage(&guessing_image);  
        cvReleaseImage(&line_image);  
        cvReleaseImage(&section_1);  
    }
```

```
    cvReleaseImage(&gray_image);
```

```
    (&binary_image);
```

```
    Invalidate( FALSE );  
}
```

3) MusicDlg::OnBnClickedPlay

```
void CMusicDlg::OnBnClickedPlay()
{
    // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
    if(flag)
    {
        AfxMessageBox(L"Please open image file first.", MB_ICONSTOP );
    }
    else
    {
//        CProgressDlg dlg;
//        AfxMessageBox(L"Please wait until the next msg pops up", MB_OK );

        CPreprocessing cPreprocessing;
        CProcessing cProcessing;
        CExtra cExtra;
        CUtil cUtil;
        CSymbol cSymbol;
        CRecognition cRecognition;
        CFinalStep cFinalStep;

        IplImage *gray_image = cPreprocessing.PRE_whiteblack(image);
        IplImage *binary_image = cPreprocessing.PRE_binary(gray_image);

        int a1=0, a2=0, a3=0, a4=0, a5=0, a6=0, a0=0;
        IplImage *hline = cvCloneImage(image);
        cPreprocessing.PRE_Hline(image, &hline, &a1, &a2, &a3, &a4, &a5, &a6, &a0);

        IplImage *section_6,*section_5,*section_4,*section_3,*section_2,*section_1;

        cPreprocessing.PRE_Segments(gray_image,&section_6,&section_5,&section_4,&section_3,&
section_2,&section_1,

a1,a2,a3,a4,a5,a6,a0);

////////////////////////////////////
/

//        if(dlg.DoModal() == IDOK && image)
```

```

        if(image)
        {
//          cvNamedWindow( "1", 1 );
//          cvShowImage( "1", image );
//          IplImage *dst_image = cPreprocessing.PRE_whiteblack(image);
//          cvNamedWindow( "2", 1 );
//          cvShowImage( "2", dst_image );

        if(a0 == 6)
        {
            IplImage *Estaffline = cvCreateImage( cvGetSize(section_6),
IPL_DEPTH_8U, 1);

            cvSet(Estaffline, CV_RGB(255,255,255), NULL);
            cProcessing.PRO_Hough(section_6, &Estaffline);    // 함수 안
에서 이진화, 캐니 연산등이 실행된다.

            int type = CV_SHAPE_RECT;
            IplImage *temp = cvCreateImage( cvGetSize(section_6),
IPL_DEPTH_8U, 3 );

            cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
            IplImage *opening_image = cExtra.EX_opening(temp, type,
NULL ); // 열림연산(모폴리지는 칼라만)

            IplImage *detection_image = cvCloneImage(section_6);
            int xStart=0, xEnd=0, yStart=0, yEnd=0;
            cProcessing.PRO_Detection(opening_image, &detection_image,
&xStart, &xEnd, &yStart, &yEnd);

            IplImage *guessing_image =
cvCreateImage( cvGetSize(section_6), IPL_DEPTH_8U, 3 );
            cvCvtColor(section_6, guessing_image, CV_GRAY2BGR );
            IplImage *line_image = cvCreateImage( cvGetSize(section_6),
IPL_DEPTH_8U, 3 );

            cvSet(line_image, CV_RGB(255,255,255), NULL);

            cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image, xStart, xEnd,
yStart, yEnd);

            cFinalStep.Final_newsymana(line_image, guessing_image,

```

```

&guessing_image, a0, 1);

//                                DrawImage(guessing_image, "Sixth passage");

                                cvReleaseImage(&Estaffline);
                                cvReleaseImage(&temp);
                                cvReleaseImage(&opening_image);
                                cvReleaseImage(&detection_image);
                                cvReleaseImage(&guessing_image);
                                cvReleaseImage(&line_image);
                                cvReleaseImage(&section_6);

                                --a0;
                                }
                                if(a0 == 5)
                                {
                                    IplImage *Estaffline = cvCreateImage( cvGetSize(section_5),
IPL_DEPTH_8U, 1);

                                    cvSet(Estaffline, CV_RGB(255,255,255), NULL);
                                    cProcessing.PRO_Hough(section_5, &Estaffline);    // 함수 안
에서 이진화, 캐니 연산등이 실행된다.

                                    int type = CV_SHAPE_RECT;
                                    IplImage *temp = cvCreateImage( cvGetSize(section_5),
IPL_DEPTH_8U, 3 );

                                    cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
                                    IplImage *opening_image = cExtra.EX_opening(temp, type,
NULL ); // 열림연산(모폴리지는 칼라만)

                                    IplImage *detection_image = cvCloneImage(section_5);
                                    int xStart=0, xEnd=0, yStart=0, yEnd=0;
                                    cProcessing.PRO_Detection(opening_image, &detection_image,
&xStart, &xEnd, &yStart, &yEnd);

                                    IplImage                                *guessing_image                                =
cvCreateImage( cvGetSize(section_5), IPL_DEPTH_8U, 3 );
                                    cvCvtColor(section_5, guessing_image, CV_GRAY2BGR );
                                    IplImage *line_image = cvCreateImage( cvGetSize(section_5),
IPL_DEPTH_8U, 3 );

```



```

        cvSet(line_image, CV_RGB(255,255,255), NULL);

        cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image,  xStart,  xEnd,
yStart, yEnd);

        cFinalStep.Final_newsymana(line_image,      guessing_image,
&guessing_image, a0, 1);

//      DrawImage(guessing_image, "Fifth passage");

        cvReleaseImage(&Estaffline);
        cvReleaseImage(&temp);
        cvReleaseImage(&opening_image);
        cvReleaseImage(&detection_image);
        cvReleaseImage(&guessing_image);
        cvReleaseImage(&line_image);
        cvReleaseImage(&section_5);

        --a0;
    }
    if(a0 == 4)
    {
        IplImage *Estaffline = cvCreateImage( cvGetSize(section_4),
IPL_DEPTH_8U, 1);

        cvSet(Estaffline, CV_RGB(255,255,255), NULL);
        cProcessing.PRO_Hough(section_4, &Estaffline);    // 함수 안
에서 이진화, 캐니 연산등이 실행된다.

        int type = CV_SHAPE_RECT;
        IplImage *temp = cvCreateImage( cvGetSize(section_4),
IPL_DEPTH_8U, 3 );

        cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
        IplImage *opening_image = cExtra.EX_opening(temp, type,
NULL ); // 열림연산(모폴리지는 칼라만)

        IplImage *detection_image = cvCloneImage(section_4);
        int xStart=0, xEnd=0, yStart=0, yEnd=0;
        cProcessing.PRO_Detection(opening_image, &detection_image,
&xStart, &xEnd, &yStart, &yEnd);

```

```

        IplImage          *guessing_image          =
cvCreateImage( cvGetSize(section_4), IPL_DEPTH_8U, 3 );
        cvCvtColor(section_4, guessing_image, CV_GRAY2BGR );
        IplImage *line_image = cvCreateImage( cvGetSize(section_4),
IPL_DEPTH_8U, 3 );

        cvSet(line_image, CV_RGB(255,255,255), NULL);

        cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image,  xStart,  xEnd,
yStart, yEnd);

        cFinalStep.Final_newsymanal(line_image,          guessing_image,
&guessing_image, a0, 1);

//          DrawImage(guessing_image, "Forth passage");

        cvReleaseImage(&Estaffline);
        cvReleaseImage(&temp);
        cvReleaseImage(&opening_image);
        cvReleaseImage(&detection_image);
        cvReleaseImage(&guessing_image);
        cvReleaseImage(&line_image);
        cvReleaseImage(&section_4);

        --a0;
    }
    if(a0 == 3)
    {
        IplImage *Estaffline = cvCreateImage( cvGetSize(section_3),
IPL_DEPTH_8U, 1);

        cvSet(Estaffline, CV_RGB(255,255,255), NULL);
        cProcessing.PRO_Hough(section_3, &Estaffline);      // 함수 안
에서 이진화, 캐니 연산등이 실행된다.

        int type = CV_SHAPE_RECT;
        IplImage *temp = cvCreateImage( cvGetSize(section_3),
IPL_DEPTH_8U, 3 );

        cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
        IplImage *opening_image = cExtra.EX_opening(temp, type,

```

```
NULL ); // 열림연산(모폴리지는 칼라만)
```

```
        IplImage *detection_image = cvCloneImage(section_3);
        int xStart=0, xEnd=0, yStart=0, yEnd=0;
        cProcessing.PRO_Detection(opening_image, &detection_image,
&xStart, &xEnd, &yStart, &yEnd);
```

```
        IplImage          *guessing_image          =
cvCreateImage( cvGetSize(section_3), IPL_DEPTH_8U, 3 );
        cvCvtColor(section_3, guessing_image, CV_GRAY2BGR );
        IplImage *line_image = cvCreateImage( cvGetSize(section_3),
IPL_DEPTH_8U, 3 );
        cvSet(line_image, CV_RGB(255,255,255), NULL);
```

```
        cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image,  xStart,  xEnd,
yStart, yEnd);
```

```
        cFinalStep.Final_newsymanana(line_image,          guessing_image,
&guessing_image, a0, 1);
```

```
//          DrawImage(guessing_image, "Third passage");
```

```
        cvReleaseImage(&Estaffline);
        cvReleaseImage(&temp);
        cvReleaseImage(&opening_image);
        cvReleaseImage(&detection_image);
        cvReleaseImage(&guessing_image);
        cvReleaseImage(&line_image);
        cvReleaseImage(&section_3);
```

```
        --a0;
```

```
    }
```

```
    if(a0 == 2)
```

```
{
```

```
        IplImage *Estaffline = cvCreateImage( cvGetSize(section_2),
IPL_DEPTH_8U, 1);
```

```
        cvSet(Estaffline, CV_RGB(255,255,255), NULL);
```

```
        cProcessing.PRO_Hough(section_2, &Estaffline);    // 함수 안
```

에서 이진화, 캐니 연산등이 실행된다.

```

        int type = CV_SHAPE_RECT;
        IplImage *temp = cvCreateImage( cvGetSize(section_2),
IPL_DEPTH_8U, 3 );

        cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
        IplImage *opening_image = cExtra.EX_opening(temp, type,
NULL ); // 열림연산(모폴리지는 칼라만)

        IplImage *detection_image = cvCloneImage(section_2);
        int xStart=0, xEnd=0, yStart=0, yEnd=0;
        cProcessing.PRO_Detection(opening_image, &detection_image,
&xStart, &xEnd, &yStart, &yEnd);

        IplImage *guessing_image =
cvCreateImage( cvGetSize(section_2), IPL_DEPTH_8U, 3 );
        cvCvtColor(section_2, guessing_image, CV_GRAY2BGR );
        IplImage *line_image = cvCreateImage( cvGetSize(section_2),
IPL_DEPTH_8U, 3 );

        cvSet(line_image, CV_RGB(255,255,255), NULL);

        cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image, xStart, xEnd,
yStart, yEnd);

        cFinalStep.Final_newsymana(line_image, guessing_image,
&guessing_image, a0, 1);

// DrawImage(guessing_image, "Second passage");

        cvReleaseImage(&Estaffline);
        cvReleaseImage(&temp);
        cvReleaseImage(&opening_image);
        cvReleaseImage(&detection_image);
        cvReleaseImage(&guessing_image);
        cvReleaseImage(&line_image);
        cvReleaseImage(&section_2);

        --a0;
    }
    if(a0 == 1)

```

```

{
    IplImage *Estaffline = cvCreateImage( cvGetSize(section_1),
IPL_DEPTH_8U, 1);

    cvSet(Estaffline, CV_RGB(255,255,255), NULL);
    cProcessing.PRO_Hough(section_1, &Estaffline);    // 함수 안에서 이진화, 캐니 연산등이 실행된다.

    int type = CV_SHAPE_RECT;
    IplImage *temp = cvCreateImage( cvGetSize(section_1),
IPL_DEPTH_8U, 3 );

    cvCvtColor(Estaffline, temp, CV_GRAY2BGR );
    IplImage *opening_image = cExtra.EX_opening(temp, type,
NULL ); // 열림연산(모폴리지는 칼라만)

    IplImage *detection_image = cvCloneImage(section_1);
    int xStart=0, xEnd=0, yStart=0, yEnd=0;
    cProcessing.PRO_Detection(opening_image, &detection_image,
&xStart, &xEnd, &yStart, &yEnd);

    IplImage *guessing_image =
cvCreateImage( cvGetSize(section_1), IPL_DEPTH_8U, 3 );
    cvCvtColor(section_1, guessing_image, CV_GRAY2BGR );
    IplImage *line_image = cvCreateImage( cvGetSize(section_1),
IPL_DEPTH_8U, 3 );

    cvSet(line_image, CV_RGB(255,255,255), NULL);

    cProcessing.PRO_Staffend(opening_image,&guessing_image,&line_image, xStart, xEnd,
yStart, yEnd);

    cFinalStep.Final_newsymana(line_image, guessing_image,
&guessing_image, a0, 1);

    // DrawImage(guessing_image, "First passage");

    cvReleaseImage(&Estaffline);
    cvReleaseImage(&temp);
    cvReleaseImage(&opening_image);
    cvReleaseImage(&detection_image);
    cvReleaseImage(&guessing_image);








```

```
        cvReleaseImage(&line_image);
        cvReleaseImage(&section_1);
    }
}
cvReleaseImage(&gray_image);
cvReleaseImage(&binary_image);
// AfxMessageBox(L"Complete computation!", MB_OK );
}
}
```

4. 음표인식

1) 음표 종류별 분석

구분	모양	실행정도	비고
직접촬영&필기 악보 이미지		X	인식 불가 본처리 과정만 가능
특이한 비율의 인쇄체 악보		X, 부분적으로 박자 오인식	부정확한 동작
일반 동요 인쇄체 악보 이미지	ex) 세마치동요악보	O 밑의 안되는 경우가 아니면 재생가능	정확한 동작
온음표		O	크기 인식
2분음표		O	특징 인식 알고리즘
4분음표		O	특징 인식 알고리즘
8분음표		O	특징 인식 알고리즘
16분음표		O	특징 인식 알고리즘
점 2분음표		O	특징 인식 알고리즘
점 4분음표		O	특징 인식 알고리즘
점 8분음표		O	특징 인식 알고리즘

점 16분음표		O	특징 인식 알고리즘
온쉼표		X	쉼표 간 크기 비교
2분쉼표		O	쉼표 간 크기 비교
4분쉼표		O	쉼표 간 크기 비교
8분쉼표		O	쉼표 간 크기 비교
16분쉼표		X	인식 불가
2 기(교리가목임)		O	특징 인식 알고리즘 각각 재생

4 기(교리가목임)		X	인식 불가
높은음자리표		O	기호 무시
낮은음자리표		X	무시된다 (높은음자리표로 재생)
셋잇단음표		X	무시되고 각각 재생됨
이음줄		O	무시하고 각각 재생됨
붙임줄		X	따로 따로 재생됨
올림표(샵)		O	조표 간 크기 비교

2) 음표분석 절차

note 배열 형식 : note[a][b](이차원 배열)

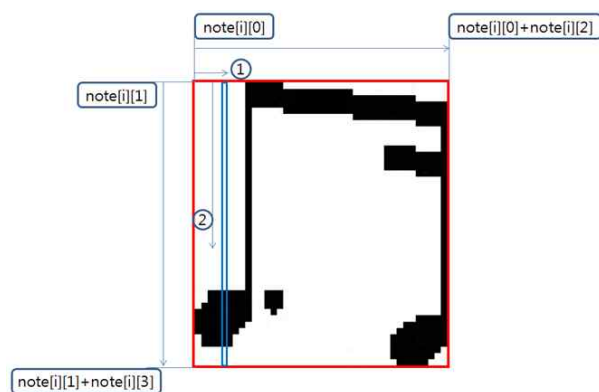
a : 음표기호가 저장되는 일종의 순서, b : 음표 기호의 특징이 저장됨.

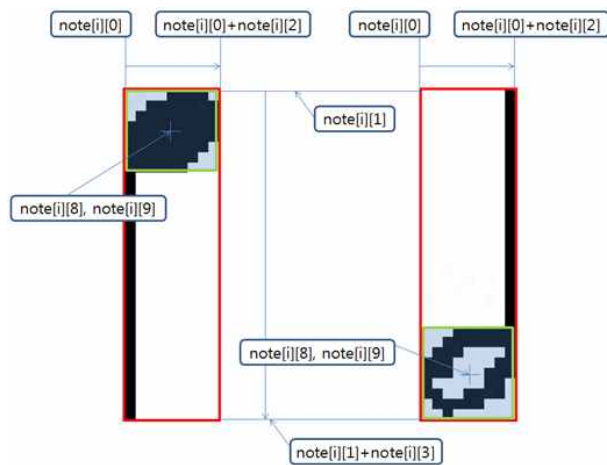
b의 값	설명	
0	기호를 검출한 직사각형의 좌상단 꼭지점의 x 좌표.	
1	기호를 검출한 직사각형의 좌상단 꼭지점의 y 좌표.	
2	기호를 검출한 직사각형의 너비	
3	기호를 검출한 직사각형의 높이	
4	기호에서 머리로 추정되는 것의 숫자(꼭 음표가 아니더라도 검출될 수 있음)	
5	머리의 위치(1 이면 아래, 2 이면 위)	
6	머리가 한 개인 음표의 경우 꼬리의 숫자	
	머리가 두 개인 음표의 첫 번째 기둥의 왼쪽 꼬리 개수	
7	머리가 한 개인 음표의 경우 머리 옆의 점의 유무(있으면 2, 없으면 1, 검사를 안 하면 0)	
	머리가 두 개인 음표의 첫 번째 기둥의 오른쪽 꼬리 개수	
8	첫 번째 음표의 머리 중심 x 좌표	
9	첫 번째 음표의 머리 중심 y 좌표	
10	두 번째 음표의 머리 중심 x 좌표	머리가 한 개일 경우, 머리의 너비
11	두 번째 음표의 머리 중심 y 좌표	머리가 한 개일 경우, 머리의 모양(2 이면 full, 1 이면 empty, 0 은 null)
12	첫 번째 대(or 기둥)의 x 좌표(여기서는 꼭지점의 x 좌표를 0 으로 보고 계산되었음)	
13	두 번째 대(or 기둥)의 x 좌표(여기서는 꼭지점의 x 좌표를 0 으로 보고 계산되었음)	
14	세 번째 대(or 기둥)의 x 좌표(여기서는 꼭지점의 x 좌표를 0 으로 보고 계산되었음)	
	머리가 두 개인 음표의 두 번째 기둥의 왼쪽 꼬리 개수	
15	첫 번째 대(or 기둥)의 왼쪽 꼬리 개수 (항상 0)	
	머리가 두 개인 음표의 두 번째 기둥의 오른쪽 꼬리 개수	

16	첫 번째 대(or 기둥)의 오른쪽 꼬리 개수
17	두 번째 대(or 기둥)의 왼쪽 꼬리 개수
18	두 번째 대(or 기둥)의 오른쪽 꼬리 개수
19	세 번째 대(or 기둥)의 왼쪽 꼬리 개수
20	세 번째 대(or 기둥)의 오른쪽 꼬리 개수
21	세 번째 음표의 머리 중심 x 좌표
22	세 번째 음표의 머리 중심 y 좌표
23	첫 번째 음표 머리 옆의 점의 유무(있으면 2, 없으면 1, 0은 null)단, 머리두개이상
24	두 번째 음표 머리 옆의 점의 유무(있으면 2, 없으면 1, 0은 null)단, 머리두개이상
25	empty
26	empty
27	empty
28	empty
29	empty
30	empty

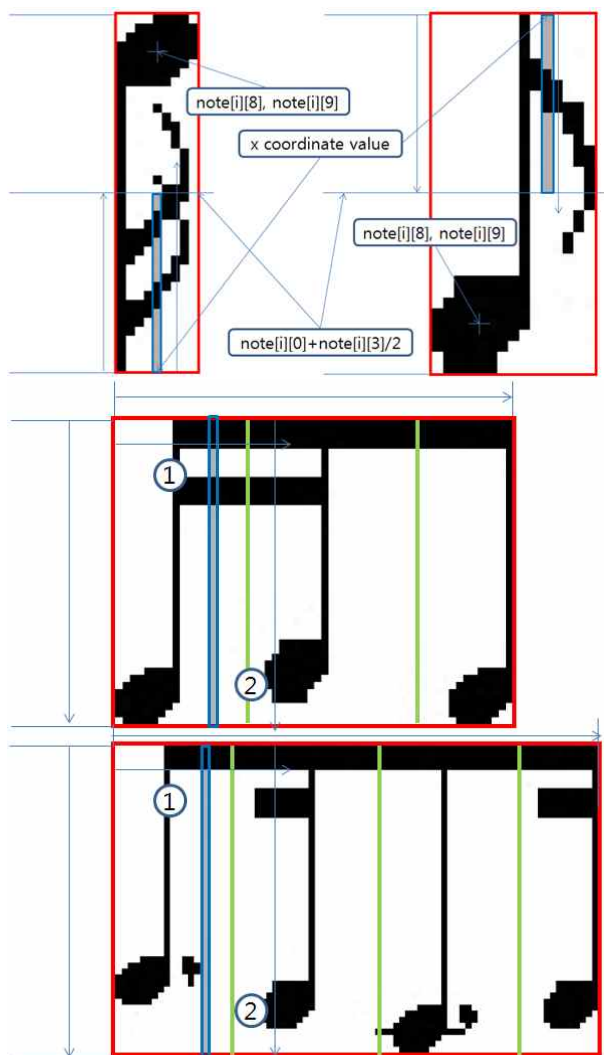
soundPosition 배열과 연주하는 소스의 계이름 위치 번호 비교

음표 분석(차례대로 대가 꼬리로 이어진 음표의 경우와 여러 가지 경우를 비교)

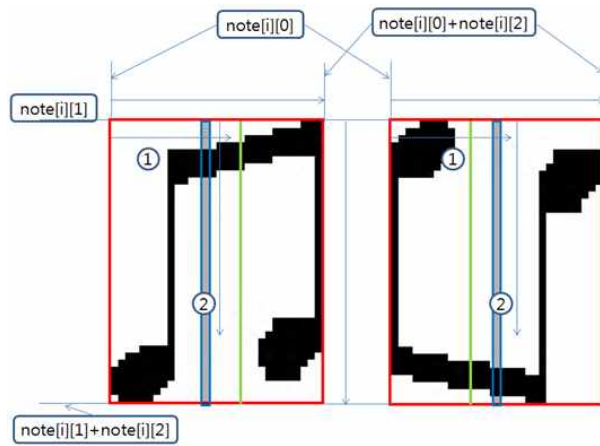




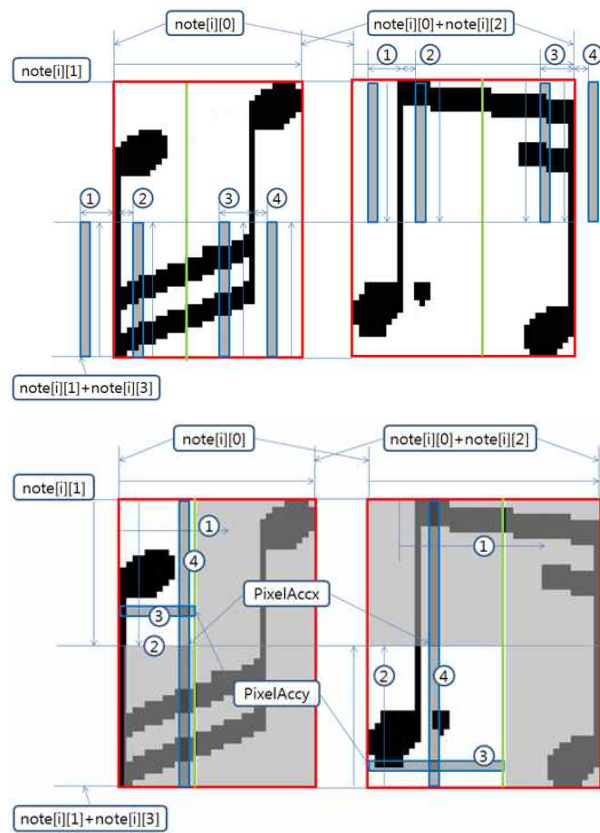
위의 세 그림은 차례대로 머리의 중점을 구해 머리 중점의 y 좌표와 오선의 상대적은 위치를 이용해 계이름을 정하는 모습과 머리가 꼭 찬 모양인지 빈 모양인지를 구하는 방식 그리고 머리 옆 점이 존재하는 지를 찾는 그림이 되겠습니다.

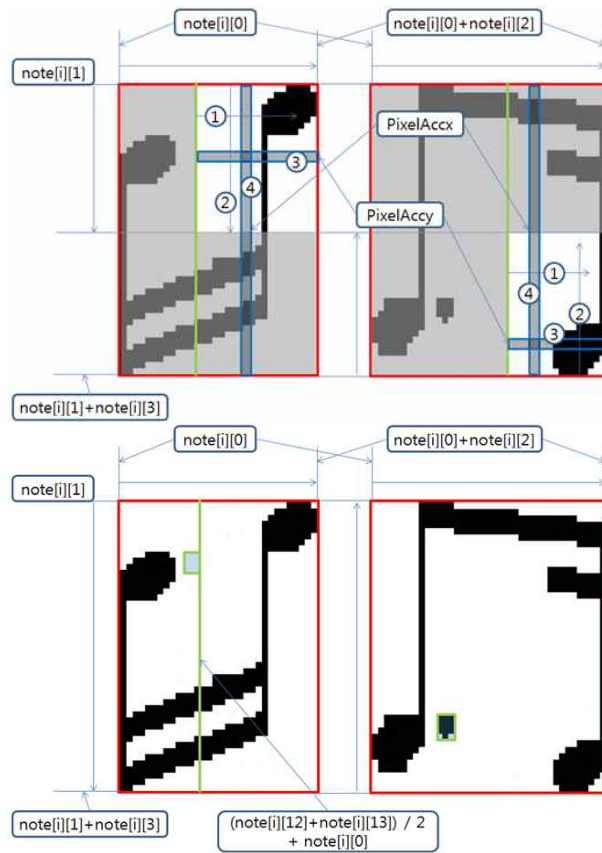


마지막으로 꼬리의 개수를 세면서 머리와 대가 하나인 경우의 음표 분석을 마치고
 이제는 beamed musical notes 에 대해서 분석해 보았습니다. 먼저 대의 위치를 기준으로

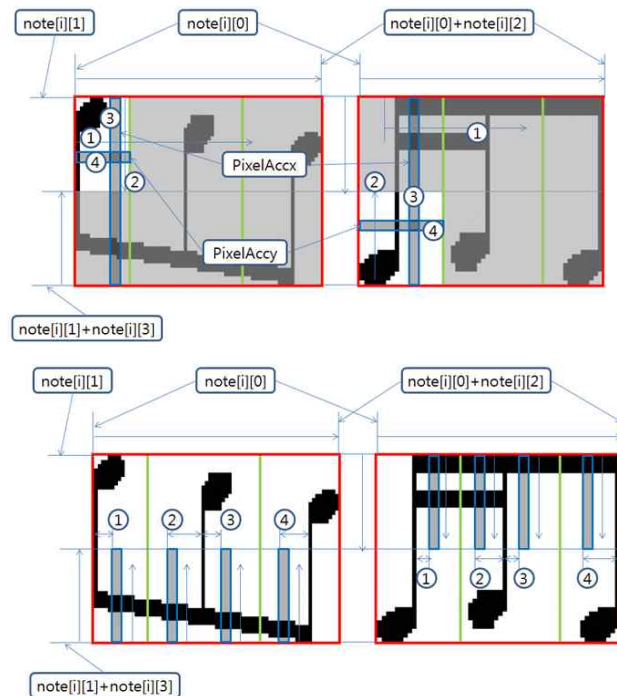


하나의 기호로 분리한 다음 계속해서 분석을 진행 하였습니다.

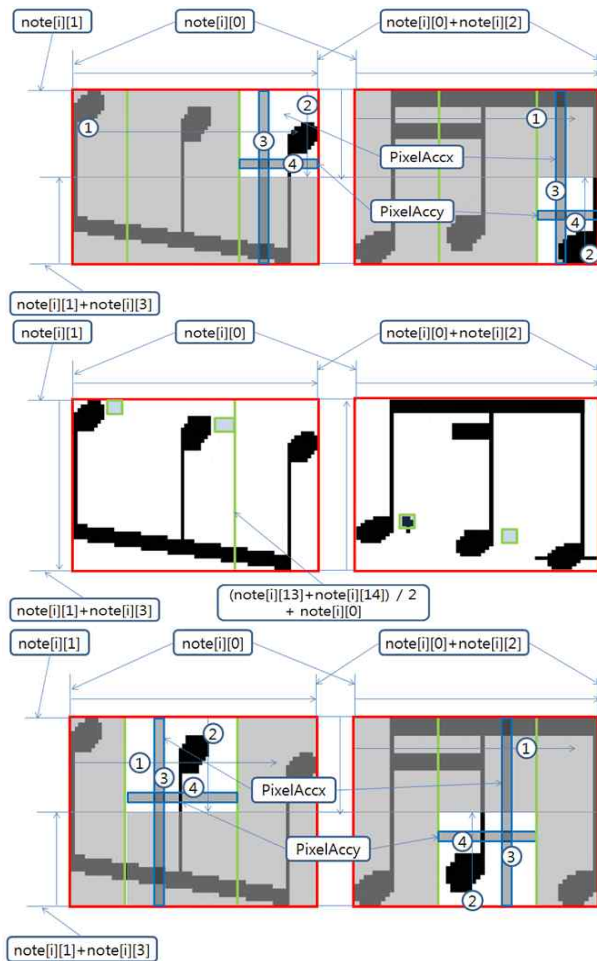




두 개의 음표를 하나씩 나눈 상태에서 머리의 중점과 꼬리의 개수, 점의 유무 등을 다시 살폈습니다.



이번에는 3 개의 음표가 연결된 경우를 살펴 보았습니다. 역시 꼬리와 머리의 중점, 점의 유무를 같은 방식으로 찾아 note 배열에 저장시켜 주었습니다.



기차길 옆

조금 빠르게

기차길 옆 오막살이 아 기 아 기 잘도 큰 다
기차길 옆 옥수수밭 옥수수는 잘도 큰 다

h 4P 4B 8B 8B 4B 4R 4B 4B 4B 4B 8B 8B 4B 4R

기차길 옆 오막살이 아 기 아 기 잘도 큰 다

그리고 나서 다시 템플릿 영상 결과처럼 인식 결과를 눈으로 볼 수 있도록 화면에 그 결과를 나타내 보았습니다.

3) 악보처리

학교종

김메리 요곡
김메리



학 교 종 이 땡 땡 땡 어 서 모 이 자
학 교 종 이 땡 땡 땡 어 서 모 이 자

새로운 오선추정

학교종

김메리 요곡
김메리

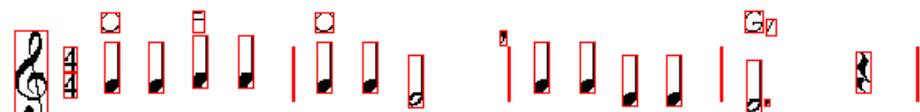


학 교 종 이 땡 땡 땡 어 서 모 이 자
학 교 종 이 땡 땡 땡 어 서 모 이 자

오선제거

학교종

김메리 요곡
김메리



학 교 종 이 땡 땡 땡 어 서 모 이 자
학 교 종 이 땡 땡 땡 어 서 모 이 자

기호인식

학교 종

김메리 요
김메리 곡



학 교 종 이 땡 땡 땡 어 서 모 이 자
학 교 종 이 땡 땡 땡 어 서 모 이 자

오선영역 밖 제거

여기서 기존 필기 형태의 그림을 보완하기 위하여 잡티 제거, 안쪽 겹친 부분 통합, 높이가 작은 박스 제거, 머리와 꼬리 통합 등의 알고리즘을 시행하였습니다. 하지만 인쇄체 악보의 경우는 바로 이 과정을 생략할 수 있습니다.

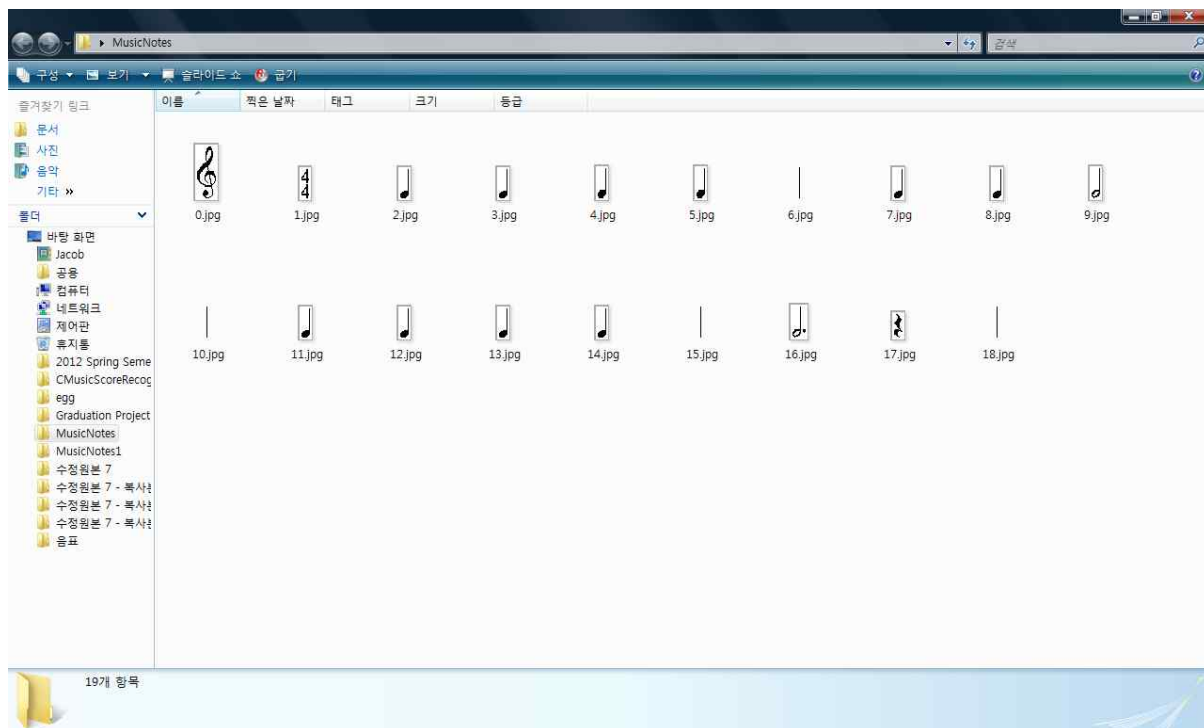
학교 종

김메리 요
김메리 곡



학 교 종 이 땡 땡 땡 어 서 모 이 자
학 교 종 이 땡 땡 땡 어 서 모 이 자

필요한 기호만 선택



각 박스로 구분된 이미지를 파일로 저장.

학교 종

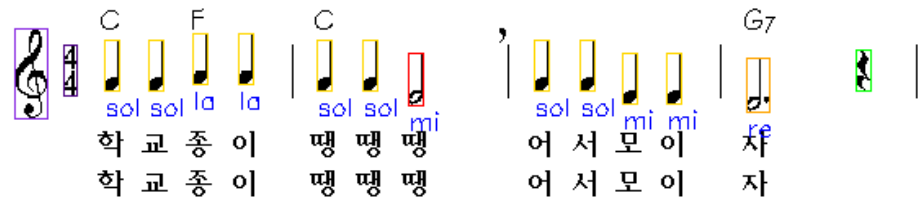
김메리 요곡
김메리

학 교 종 이 땡 땡 땡 어 서 모 이 자
학 교 종 이 땡 땡 땡 어 서 모 이 자

이 영상은 4분 음표만 가지고 사이즈 조절 (70~120%), 회전 조절 (-5 ~ +5도)을 하면서 악보의 첫 소절을 템플릿 매칭을 통해서 출력한 이미지 입니다. 하지만 약 3분에 이르는 소요 시간으로 인해 사이즈와 회전을 각각 100%, 0도로 고정하고 다른 기호까지 분석해 보았습니다. 현 이미지는 축소된 4분 음표의 머리 만을 가지고 프로그램이 분석한 결과 입니다.

학교 종

김메리 요
김메리 곡



각 기호 별로 색깔을 넣어 구분을 하였으며, 음표의 경우 머리와 오선의 상대적인 위치 관계에 의하여 계이름을 구분하고 네모박스 아래에 계이름이 출력 되도록 만들었습니다. 이전에 시행하였던 사이즈 조정과 각도 조정은 고정해 놓은 상태 입니다.



나머지 악보도 같은 방식으로 하였습니다. 이제까지 한 소절씩 해오던 방법을 원 악보에서 클릭 한번으로 소절 별, 음표 별로 박자, 계이름이 분석되도록 처리 하였습니다.

여기서 나온 박자, 계이름 정보를 미디 소스를 이용해 재생을 시도해 보았습니다.

4) 악보재생

mmsystem.h 미디 라이브러리에서 제공하는 함수와 변수타입을 이용하여 멜로디 출력 부분을 구현하였습니다. 사용할 여러 가지 구조체들과 함수를 정의하고

```
for(i=0; i<50; i++)  
{  
    Sleep(3);  
    Midi_SendShortMsg(m_DevHandle, 0x90, (BYTE)(0x30+p_Note[i].seq), 120);  
    Sleep(300*p_Note[i].beat);  
    Midi_SendShortMsg(m_DevHandle, 0x80, (BYTE)(0x30+p_Note[i].seq), 120);  
}
```

이 함수를 통해 처리된 악보의 정보를 하나의 구조체에 정리하여 삽입 후에 그 구조체 정보를 재생합니다.

해당 음높이의 멜로디를 출력하며 박자에 비례하여 Sleep함수를 호출 함으로써 박자를 맞춰줍니다.

5. 개발환경

OS – Windows

OS는 기본적인 윈도우 환경에서 구동할 예정입니다.

IDE - Visual Studio 2012

최신버전보다는 안정성을 위해서 Visual Studio 2012 버전을 사용할 계획입니다.

Language - MFC, C/C++

효율적인 UI 설계를 위해서 MFC를 사용할 것입니다. 이를 통해서 악보와 음표 등의 영상처리 화면을 효율적으로 보여줄 수 있을 것입니다.



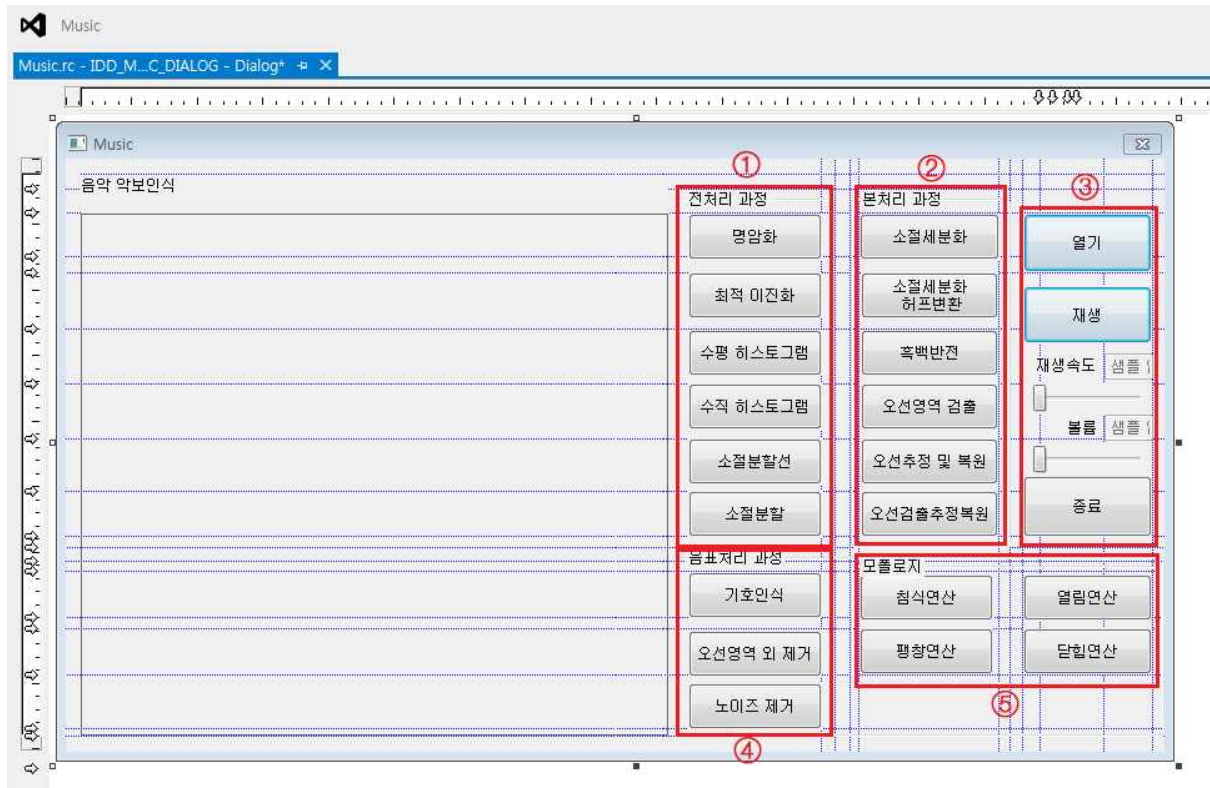
그리고 OpenCV와의 연동을 위해서 C/C++를 사용할 예정입니다.

Library - OpenCV 2.4.8

OpenCV 1.0 기준으로 설명되어 있는 자료와 문헌들이 많았고, 예제 역시 1.0 기준이었습니다. 하지만 최신버전인 이번 프로젝트에서는 OpenCV 2.4.8을 사용하고 있습니다. 그래서 바뀐 API와 함수를 공식 레퍼런스를 통해서 참고해가면서 작업하고 있습니다.



6. UI 및 기능



- ① 전처리과정

명암화, 최적이진화, 수평 히스토그램, 수직 히스토그램

- ② 본처리과정

소절세분화, 허프변환, 흑백반전, 오선영역 검출, 오선추정 및 복원, 오선검출추정복원

- ③ 옵션 UI

파일열기, 재생 및 재생옵션 UI

- ④ 음표처리 과정

기호인식, 오선영역 외 제거, 노이즈 제거

- ⑤ 모폴로지

침식연산, 열림연산, 팽창연산, 닫힘연산

7. 실행

1) 전처리 과정

풍 당 풍 당

음성
중간
악보

풍 당 풍 당 돌 을 던 지 자 누 나 물 래 돌 을 던 지 자
풍 당 풍 당 돌 을 던 지 자 누 나 물 래 돌 을 던 지 자

넋 물 아 퍼 저 라 널 리 널 리 퍼 저 라
넋 물 아 퍼 저 라 퍼 질 대 로 퍼 저 라

건 너 편 에 앞 아 서 나 물 을 씻 는
고 운 노 래 한 마 디 들 러 달 라 고

우 리 누 나 손 등 을 간 질 어 주 어 라
우 리 누 나 손 등 을 간 질 어 주 어 라

명암화

풍 당 풍 당

음성
중간
악보

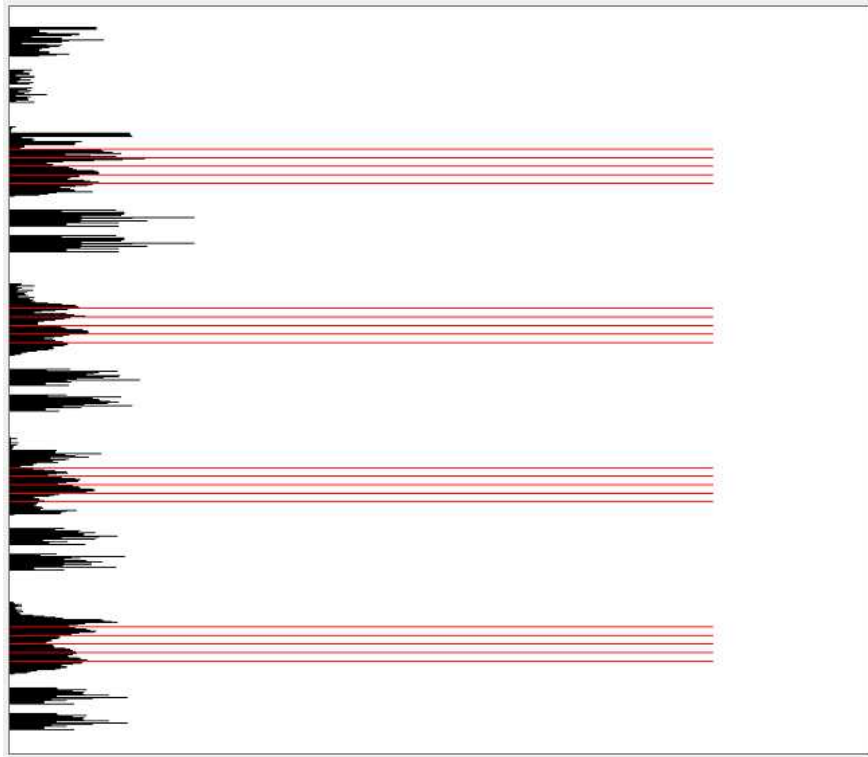
풍 당 풍 당 돌 을 던 지 자 누 나 물 래 돌 을 던 지 자
풍 당 풍 당 돌 을 던 지 자 누 나 물 래 돌 을 던 지 자

넋 물 아 퍼 저 라 널 리 널 리 퍼 저 라
넋 물 아 퍼 저 라 퍼 질 대 로 퍼 저 라

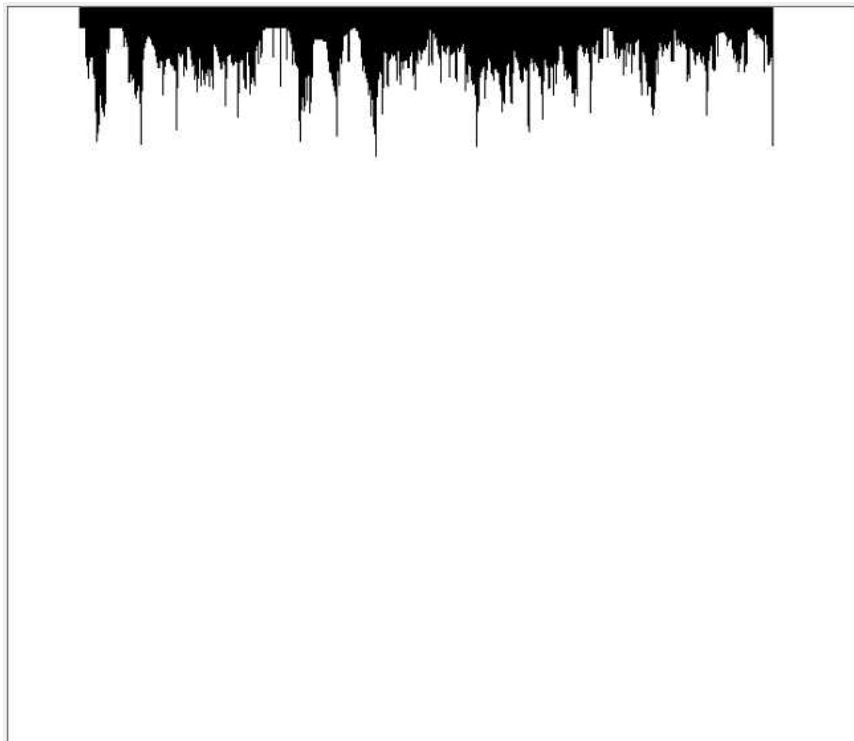
건 너 편 에 앞 아 서 나 물 을 씻 는
고 운 노 래 한 마 디 들 러 달 라 고

우 리 누 나 손 등 을 간 질 어 주 어 라
우 리 누 나 손 등 을 간 질 어 주 어 라

최적이진화



수평 히스토그램



수직 히스토그램

풍당풍당

유리
정리
음향

풍당풍당 돌을 던지 자 누나 몰래 돌을 던지 자
 풍당풍당 돌을 던지 자 누나 몰래 돌을 던지 자

넋 몰아 퍼 저라 널 리 널 리 퍼 저라
 넋 몰아 퍼 저라 퍼 질 대로 퍼 저라

건 너편 에 앞 아 서 나 물을 찢 는
 고 온 노 래 한 마 디 들 러 달 라 고

우 리 누 나 손 등 을 간 질 어 주 어 라
 우 리 누 나 손 등 을 간 질 어 주 어 라

소절분할선

풍당풍당

유리
정리
음향

풍당풍당 돌을 던지 자 누나 몰래 돌을 던지 자
 풍당풍당 돌을 던지 자 누나 몰래 돌을 던지 자

포다포다 도오 더지 지 누나 몰래 도오 더지 지
 포다포다 도오 더지 지 누나 몰래 도오 더지 지

소절분할

2) 본처리 과정

[illegible]

소절세분화

[illegible]

소절세분화

허프변환

풍당풍당

오리
정규
음성

풍당풍당 돌을 던지 자 누나 몰래 돌을 던지 자
풍당풍당 돌을 던지 자 누나 몰래 돌을 던지 자

넋 물아 퍼 저라 널 리 널 리 퍼 저라
넋 물아 퍼 저라 퍼 질 대로 퍼 저라

건 너편 에 앞 아 서 나 물을 찢 는
고 운노 래 한 마 디 들 러 달 라 고

우 리누 나 손 등 을 간 짚어 주 어 라
우 리누 나 손 등 을 간 짚어 주 어 라

흑백변환

풍당풍당

오리
정규
음성

풍당풍당 돌을 던지 자 누나 몰래 돌을 던지 자
풍당풍당 돌을 던지 자 누나 몰래 돌을 던지 자

넋 물아 퍼 저라 널 리 널 리 퍼 저라
넋 물아 퍼 저라 퍼 질 대로 퍼 저라

건 너편 에 앞 아 서 나 물을 찢 는
고 운노 래 한 마 디 들 러 달 라 고

우 리누 나 손 등 을 간 짚어 주 어 라
우 리누 나 손 등 을 간 짚어 주 어 라

오선영역 검출

풍 당 풍 당

윤석중
오선추정

풍 당 풍 당 돌 을 던 지 자 누 나 몰 래 돌 을 던 지 자
 풍 당 풍 당 돌 을 던 지 자 누 나 몰 래 돌 을 던 지 자

넋 물 아 퍼 저 라 널 리 널 리 퍼 저 라
 넋 물 아 퍼 저 라 퍼 질 대 로 퍼 저 라

건 너 편 에 앞 아 서 나 물 을 씻 는
 고 운 노 래 한 마 디 들 러 달 라 고

우 리 누 나 손 등 을 간 질 어 주 어 라
 우 리 누 나 손 등 을 간 질 어 주 어 라

오선추정 및 복원

3) 모폴로지

풍 당 풍 당

윤석중
오선추정

풍 당 풍 당 돌 을 던 지 자 누 나 몰 래 돌 을 던 지 자
 풍 당 풍 당 돌 을 던 지 자 누 나 몰 래 돌 을 던 지 자

넋 물 아 퍼 저 라 널 리 널 리 퍼 저 라
 넋 물 아 퍼 저 라 퍼 질 대 로 퍼 저 라

건 너 편 에 앞 아 서 나 물 을 씻 는
 고 운 노 래 한 마 디 들 러 달 라 고

우 리 누 나 손 등 을 간 질 어 주 어 라
 우 리 누 나 손 등 을 간 질 어 주 어 라

침식연산

풍 당 풍 당

팽창연산

풍 당 풍 당

운석풍요
출판부

풍 당 풍 당 돌 올 먼지 자 누 나 물 려 돌 올 먼지 자
풍 당 풍 당 돌 올 먼지 자 누 나 물 려 돌 올 먼지 자

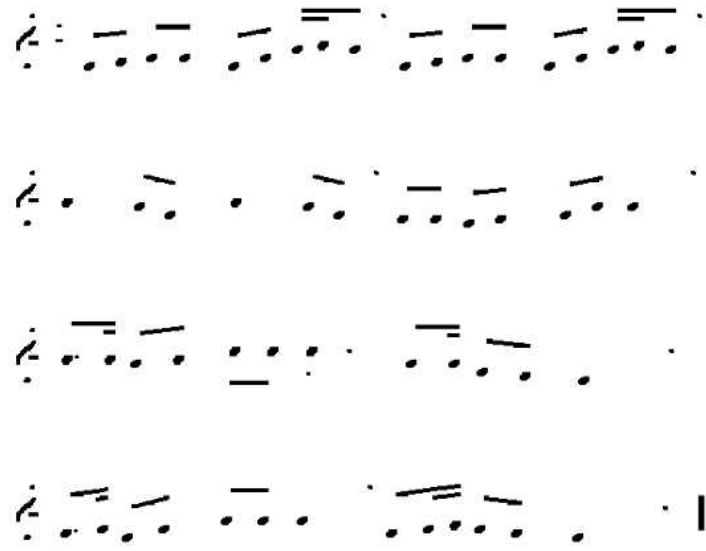
넋 물 아 떠 저 라 널 리 널 리 떠 저 라
넋 물 아 떠 저 라 떠 집 데 보 떠 저 라

견 너 편 에 앉 아 서 나 물 을 쫓 는
고 운 노 려 한 마 디 돌 러 달 라 고

우 리 누 나 손 동 올 간 집 어 주 이 라
우 리 누 나 손 동 올 간 집 어 주 이 라

열림연산

풍 당 풍 당



단힘연산

8. 일정

담당자	세부사항	4월				5월			
		1주차	2주차	3주차	4주차	1주차	2주차	3주차	4주차
송명서	명암화, 이진화 영상처리								
	수평 수직 히스토그램을 통한 악보분석								
	소절분할을 위한 선작업								
	소절분할된 데이터를 세분화								
	흑백반전을 통한 악보 분석								
	오선영역 추정 및 검출								
	오선영역 복원작업								
	디버깅 및 테스트								
박상원	이진화 및 OpenCV API 단위 함수화								
	함수 모듈 MFC 연동								
	음표처리 작업을 위한 영상처리								
	기호인식, 높이인식, 폭인식, 노이즈 제거								
	음표분석 및 연주음표 인식								
	디버깅 및 테스트								
전체	디버깅 및 테스트								

영상처리에 대한 이론을 바탕으로 작업을 하는 것이지만, OpenCV 라이브러리에 대한 이해가 필요하므로 사전에 기반지식 및 조사를 갖추고 작업했습니다.

특히 이론적으로 공부했던 OpenCV 1.0 버전이 아닌 OpenCV 2.4.8버전으로 작업하기로 했기 때문에 달라진 API에 대해서 공부하고 작업해야 할 필요가 있었습니다.

영상처리를 위한 알고리즘부분을 송명서 학생이 맡기로 했고, OpenCV의 직접적인 API 작업 및 MFC 작업등은 박상원 학생이 주로 맡기로 했습니다. 물론 음표에 대한 인식과 재생을 위한 모듈 등은 거의 공동작업이 될 것입니다.

디버깅 및 테스트 기간에는 여러 악보와 다양한 조건 등의 환경에서 테스트해 볼 예정이며, 인식률을 높이기 위해서 다른 알고리즘도 시도해볼 계획입니다.

9. 참고문헌

OpenCV를 이용한 컴퓨터 비전 실무 프로그래밍

Visual C++을 이용한 디지털 영상처리(Digital Image Processing)

IT EXPERT 영상처리 프로그래밍 by Visual C++

OpenCV API Reference

(<http://docs.opencv.org/modules/refman.html>)

Microsoft Visual Studio 2012 MSDN

([http://msdn.microsoft.com/ko-kr/library/dd831853\(v=vs.110\).aspx](http://msdn.microsoft.com/ko-kr/library/dd831853(v=vs.110).aspx))