졸업 작품 보고서

작품명 : ⠅⠎⠐⠄⠨ 읽어주기

■ 일　　　자 : 2017년 11월 17일

■ 제　작　자 : 이승주, 김혜진

■ 학　과　명 : 컴퓨터공학과

졸업 작품 보고서

작품명: ⠠⠎⠠⠄⠠⠄⠠⠎ 읽어주기

■ 일　　자 : 2017년 11월 17일
■ 제 작 자 : 이승주, 김혜진
■ 학 과 명 : 컴퓨터공학과
■ 지도교수명 : 손용락, 김재현

목 차

1. 제작 동기 및 목적

가. 제작 동기

최근 엄청난 속도의 기술발전으로 편의증진이 가속화되고 있지만, 여전히 장애인들을 위한 기술발전의 속도는 더딜뿐더러 진입장벽 또한 높다. 정부는 1996년 장애인복지를 증진시키기 위해 장애인복지대책위원회에서 장애인정책종합계획을 추진해 장애인복지법을 근거로 5년마다 장애인정책종합계획을 수립·시행했고, 오는 2018년 제5차 장애인종합정책계획을 앞두고 있다. 하지만 제1차~제4차 장애인정책종합계획은 정부가 진행하는 최소한의 내용만 가지고 있어 실제 결과에 한계가 있다는 지적을 받고 있다. 따라서, 장애인의 80%이상을 차지하는 후천적 장애인을 위한 작품을 만드는 것이 의미있다고 생각하였고, 그중에서도 후천적 시각장애인을 위한 작품제작에 착수하였다.

나. 제작 목적

점자에는 다음과 같은 대표적 단점이 있다.

1. 읽기 속도가 목독이나 청독보다 현저하게 느리다.

2. 점자는 6개의 점으로 모든 글자를 나타내야 하기 때문에 같은 모양의 점자라도 상황에 따라 다른 뜻을 가질 수 있다.

3. 점자는 손으로 만져지는 부분만 감지할 수 있기 때문에 문장의 전체 내용을 파악하려면 앞에서 읽은 내용을 기억하고 있어야 하기 때문에 상당한 기억력과 이해력이 요구된다.

따라서, 점자의 단점을 보완하고, 새로 6점 점자를 학습해야 하는 후천적 시각 장애인에게 도움을 주는 장비 제작이라는 목적을 수립하였다.

2. 소요기술 동향

가. 시각장애인을 위한 최근 기술

(1) 점자 시계(Dot Watch)

Dot Watch는 기존의 스마트 시계와 달리 점자를 통해 정보를 전달하는 스마트 시계이다. 블루투스로 연결된 디지털 기기에서 전달된 정보를 전자 신호에 따라, 시계의 전면에 있는 24개의 점을 이용해 점자로 표시해주는 방식이다. 이런 방식을 통해 시간, 알람, SNS, 메시지, 통화 등 일반적인 스마트 시계의 기능을 상당 부분 구현하고 있다.[1)]

1) "시각장애인을 위한 과학기술", http://naver.me/xGx3bwDJ

(2) 아이폴리(Aipoly)

아이폴리(Aipoly)라는 애플리케이션이다. 이 애플리케이션은 인터넷을 연결하지 않아도 스마트폰 카메라를 통해 주변의 물체를 실시간으로 확인할 수 있다. 사용방법 또한 간단하다. 스마트폰에 무료로 아이폴리 앱을 다운받고, 사물을 스마트폰 카메라로 인식하며 색상과 브랜드와 같은 자세한 정보를 음성으로 알려준다. 아이폴리는 1초에 3개의 물체를 확인할 수 있으며, 약 1000가지의 물체를 인식할 수 있다. 또한 아이폴리의 특별한 매력은 이미 저장된 물체만 인식하는 것으로 끝나는 것이 아니라, 앱이 인식하지 못하는 물체를 가르칠 수 있기 때문에 사용하면 할수록 물체 인식 능력이 높아진다. 현재 아이폴리는 영어, 불어, 이탈리아어, 독일어, 스페인어, 아랍어, 일본어까지 총 7개 언어를 지원하고 있다.[2)]

2) "시각장애인을 위한 과학기술", http://naver.me/xGx3bwDJ

(3) 가상 지팡이(virtual cane)

거리를 다닐 때 지팡이가 이곳저곳에 부딪혀 불편함을 느꼈을 시각장애인들 위해 가상 지팡이가 탄생했다. 히브리대 연구진이 개발한 '가상 지팡이(virtual cane)'는 휴대폰 크기만 한 탐지기다. 눈에 보이지 않는 집속 빔(focused beam)을 쏘아 되돌아오는 신호를 감지하는 원리로 작동한다. 탐지기는 물체가 위치한 방향, 거리, 높이 등을 계산해 각각 다른 형태의 진동으로 정보를 전달한다. 반경 10M 거리에 있는 물체의 높낮이까지 식별 가능하며, 사람 얼굴에 집속 빔을 노출시키면 표정까지 알아낼 정도다. 가상 지팡이는 한번 충전하면 12시간 이상 사용 가능하고, 무게 역시 보통 지팡이와 비슷하다. 몇 분 정도만 교육받으면 탐지기가 전달하는 진동 정보를 무리 없이 해석할 수 있어, 적응기간 없이 바로 활용할 수 있다.[3)]

3) "'흰 지팡이의 날', 장애인을 위한 기술은 어디까지 발전했을까?", https://blog.naver.com/mocienews/220508438132

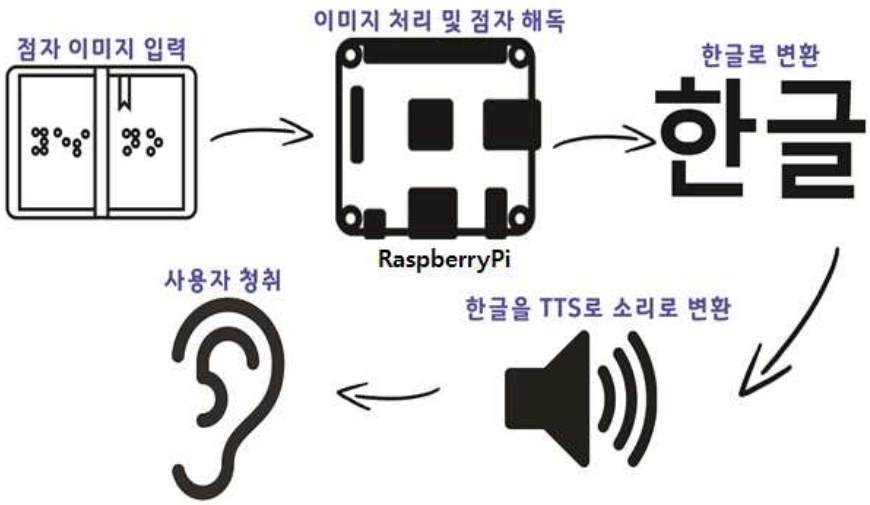
3. 개발 내용

가. 작품 소개

❖개발환경

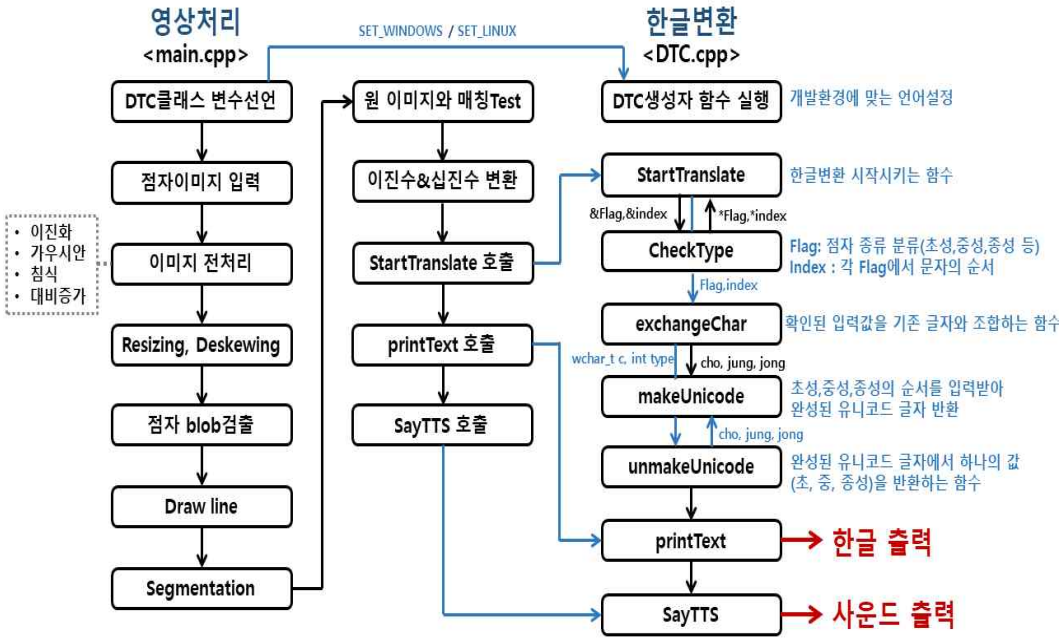
OS : Linux(raspbian)
Hardware : RaspberryPi
Software : OpenCV(3.2.0)
Language : C++

❖작품개요



이 작품은 라즈베리파이보드에 점자 이미지가 입력되면 영상처리를 거쳐 한글로 변환한다. 그리고 변환된 한글 결과를 TTS(Text To Speech)기능을 통해 소리로 출력시켜준다.

나. 시스템 구성도



먼저, 영상처리를 시작하기에 앞서 한글변환 클래스에 개발환경이 윈도우인지 리눅스인지를 알리는 값을 전달한다. 그러면 한글변환 클래스에 해당 개발환경에 맞는 언어가 설정되고 그 다음으로 영상처리 과정이 진행된다.

점자이미지를 입력하면 이진화/ 가우시안/ 침식/ thresholding 과정으로 점자만을 검출하기위한 이미지 전처리가 이루어진다. 원활한 영상처리를 위해 이 이미지의 크기를 2배로 확장시키고(Resizing) 기울임 보정을 한다(Deskewing). 그리고 OpenCV의 BlobDetector를 사용해 점자Blob들을 검출하고 line을 그려 점자간의 간격을 사용해 같은 문자인지 다른 문자인지 판단한다. 그 다음은 Segmentation과정으로 3행 2열씩 Rect구조체에 담는다. 이 구조체에는, 각각의 점자문자에 점자의 유무를 0과1

로 표현한 3x2의 데이터를 decimal로 변환해 놓은 Value값이 담긴다. 그 다음으로 addWeighted()함수를 사용해 원 이미지와 매칭시켜 매칭정도를 확인한다. 영상처리의 마지막 단계로, 점자문자들 각각의 이진수, 십진수 value들을 출력시키며 이 값을 StartTranslate함수에 전달해 한글변환 처리를 시작시킨다.

한글변환을 시작시키는 StartTranslate함수가 실행되면 Flag와 Index의 주소값을 CheckType함수에 넘겨주며, CheckType함수는 Value값들 각각의 Flag(해당 점자가 어떤 의미를 지닌 점자인지 분류하는 값 (ex.초성,중성,종성 등))와 Index(각 Flag에서 문자의 순서)를 계산하여 이 값을 다시 StartTranslate함수에 반환한다. 그러면 StartTranslate함수는 이 Flag, Index값을 exchanegeChar함수에 전달한다. 점자는 이전의 문자가 뭐였는지에 따라 다른 뜻을 가지기 때문에 exchanegeChar함수는 이전 문자와 조합하는 역할을 해준다. 그리고 makeUnicode에 넘겨 유니코드 글자를 완성시키는데, 쌍받침이나 약어 등의 경우에는 UnmakeUnicode함수에 글자와 원하는 타입(초성, 중성, 종성)을 전달해 해당 글자에서 해당 타입을 분리해 반환하고, 이 값을 사용해 조합한 결과를 makeUnicode에서 유니코드 글자로 완성시켜준다. 유니코드 글자들이 다 만들어지면 main함수에서는 printText함수를 호출해 이 문자들을 터미널에 출력시키고, 마지막으로 SayTTS함수를 호출하면 이 글자들이 소리로 출력되게 된다.

다. 구체적인 개발내용

(1) 영상처리 과정

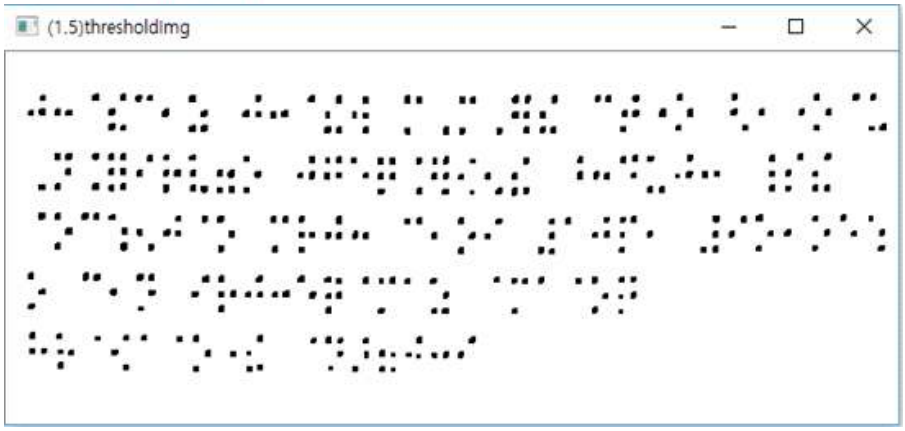
1. 양각, 음각에 민감한 점자의 특성을 살려 휘도(Y)성분만 추출하여 사용한다.

```
int main(int argc, char** argv)
{
    DTC C(SET_LINUX);

    // 1) read input image 점자 이미지 읽어오기
    Mat inputImg;
    //inputImg = imread(InputFile);
    inputImg = imread(argv[1]);

    cvtColor(inputImg, inputImg, CV_RGB2YCrCb);
    vector<cv::Mat> channels; //채널 분리를 위해
    cv::split(inputImg, channels); //채널 나누기
    inputImg = channels[0];//Y(휘도)성분사용 - 빛에 민감하기 위해(양각,음각)

    imshow("(1)inputImg", inputImg);
```

2. 이미지 전처리를 한다.

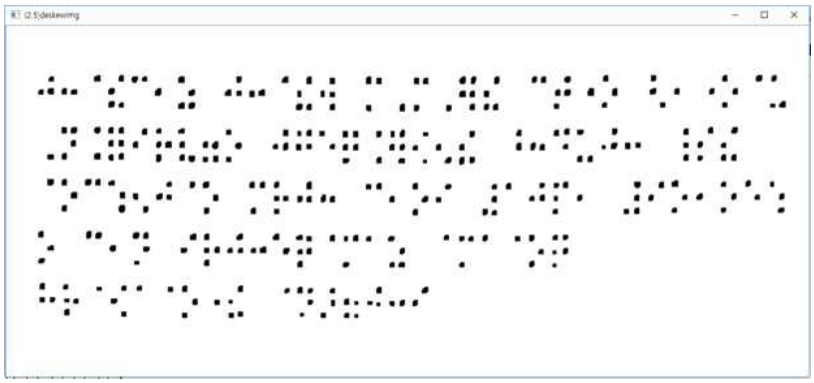
-이진화를 통해 점자와 배경을 뚜렷이 구분되게 하고, 가우시안 블러링기법으로 잡음을 제거하며, 침식기법으로 디테일을 강조한다. 또한, Thresholding으로 대비를 증가시켜 점자들이 검출되기 쉽도록 한다.

```
// 2) pre-processing input image 이미지 전처리
// apply adaptive threshold
Mat thresholdImg;
adaptiveThreshold(inputImg, thresholdImg, 255, CV_ADAPTIVE_THRESH_MEAN_C, CV_THRESH_BINARY, 21, 10);//이진화

//apply gaussian blur & erode and then threshold again
Mat morphologyElement3x3 = getStructuringElement(MORPH_RECT, Size(3, 3));//침식에 사용 할 사각형마스크
GaussianBlur(thresholdImg, thresholdImg, Size(3, 3), 0);//가우시안 블러링- 잡음제거
erode(thresholdImg, thresholdImg, morphologyElement3x3);//침식 - 디테일 강화
threshold(thresholdImg, thresholdImg, 21, 255, CV_THRESH_BINARY);//대비증가 시킴
imshow("(1.5)thresholdImg", thresholdImg);
```



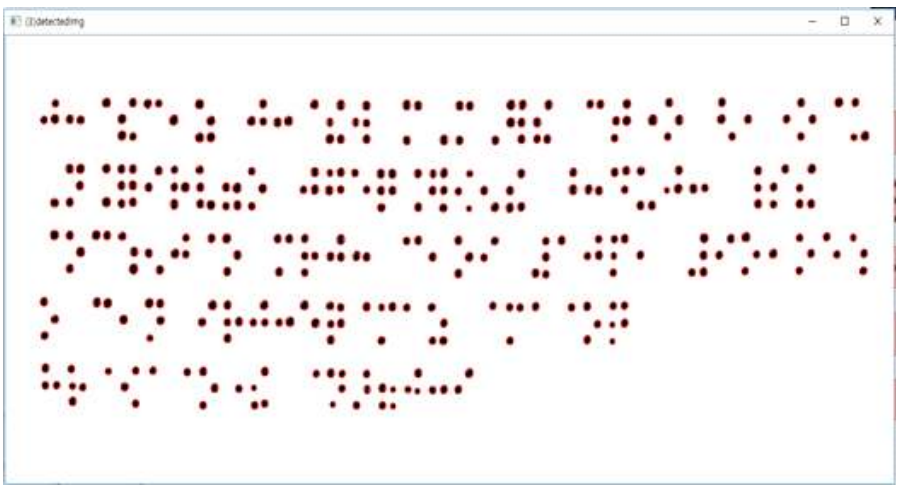
3. 원활한 영상처리를 위해 이미지 크기를 2배로 확대하고, 기울임을 보정한다.(Resizing & Deskewing)

```
    // make margin and resize from original image
    Mat measureImg = Mat(Size(thresholdImg.cols + MEASURE_MARGIN, thresholdImg.rows + MEASURE_MARGIN), CV_8UC1, 255);//margin추가
    thresholdImg.copyTo(measureImg(Rect(MEASURE_MARGIN / 2, MEASURE_MARGIN / 2, thresholdImg.cols, thresholdImg.rows)));
    resize(measureImg, measureImg, measureImg.size() * 2);//리사이징
    imshow("(2)measureImg", measureImg);


#ifdef OPTIONAL_PROC
    // 2.5) deskewing image //기울임 보정
    Mat deskewImg = measureImg;
    vector<Point> points;
    for (int i = 0; i < deskewImg.cols; ++i) {
        for (int j = 0; j < deskewImg.rows; ++j) {
            if (!deskewImg.at<uchar>(Point(i, j))) {
                points.push_back(Point(i, j));
            }
        }
    }
    RotatedRect box = minAreaRect(Mat(points));
    double angle = box.angle;
    if (angle < -45.0)
        angle += 90.0;
    wcout << "angle : " << angle << endl;

    Mat rotatedImg = getRotationMatrix2D(box.center, angle, 1);
    warpAffine(deskewImg, deskewImg, rotatedImg, deskewImg.size(), INTER_CUBIC);
    imshow("(2.5)deskewImg", deskewImg);
#endif
```



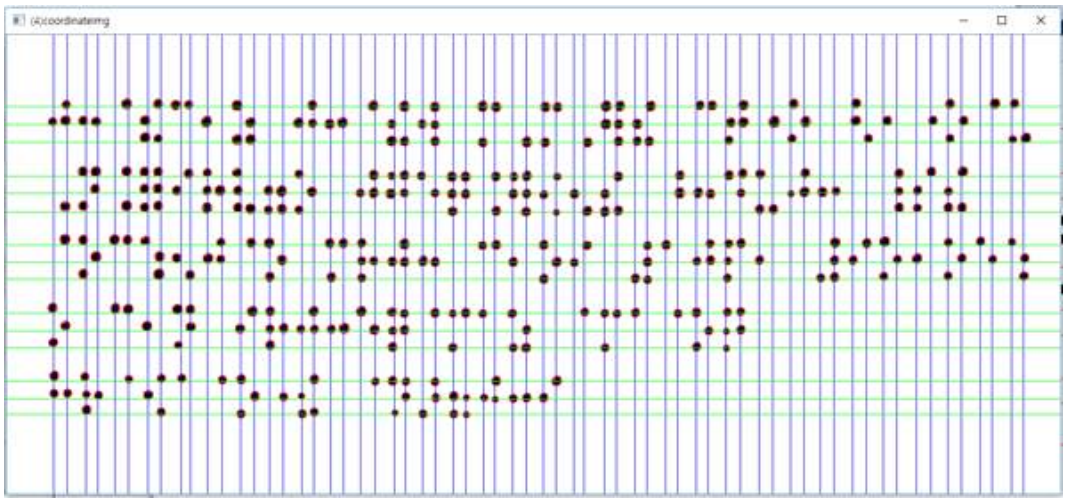
4. OpenCV의 BlobDetector사용하여 점자Blob들을 검출해 Vector에 담고, 이들을 빨간 테두리로 표시한다.

```
// 3) detect blobs 점자 blob찾기-> opencv의 SimpleBlobDetector사용
SimpleBlobDetector::Params params;
params.filterByArea = true;
params.minArea = PARAMS_MIN_SIZE * PARAMS_MIN_SIZE;
params.maxArea = PARAMS_MAX_SIZE * PARAMS_MAX_SIZE;

Ptr<SimpleBlobDetector> blobDetector = SimpleBlobDetector::create(params);
vector<KeyPoint> keypoints;
blobDetector->detect(measureImg, keypoints);//blob이 검출되면 vector인 keypoints에 넣음

if (keypoints.empty()) {//keypoints가 없으면
    wcout << "there is no braille existance condition" << endl;
    return 1;
}
Mat detectedImg = Mat(measureImg.size(), CV_8UC3);
//검출된 blob들을 담은 벡터 keypoints를 빨간색테두리로 표시
drawKeypoints(measureImg, keypoints, detectedImg, Scalar(0, 0, 255), DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
imshow("(3)detectedImg", detectedImg);
```



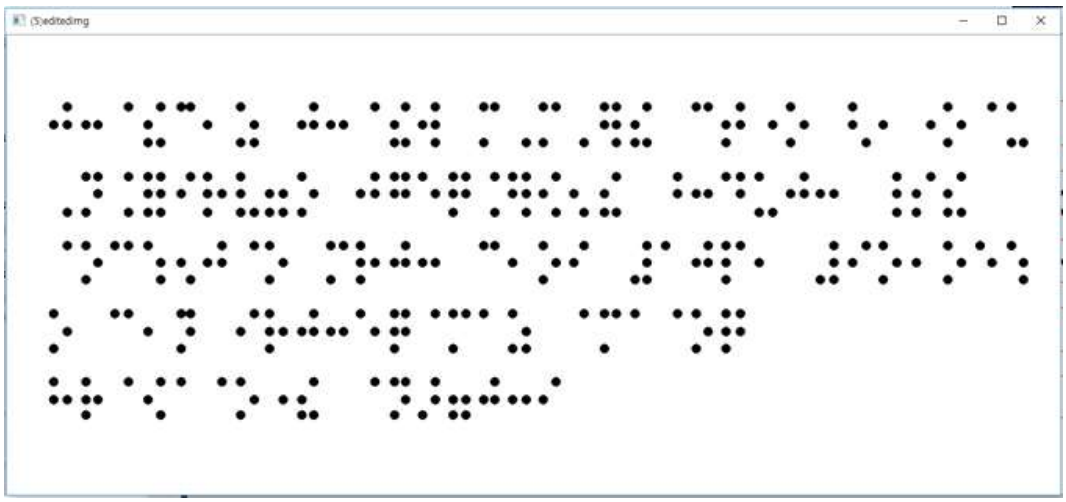
5. 검출된 Blob들에 가로, 세로라인을 그려 같은 문자인지 다른 문자인지를 판단한다.

```
// 4) normalize keypoints to coordinate line set
vector<int> coordinateX;//가로라인으로 사용할 vector (점의 위치만 검출)
vector<int> coordinateY;//세로라인으로 사용할 vector
vector<int> lineX;      //가로라인으로 사용할 vector (점이 없는 부분도 검출)
for (int i = 0; i < keypoints.size(); ++i) {
    bool isNew = true;
    for (vector<int>::iterator iter = coordinateX.begin(); iter < coordinateX.end(); ++iter) {
        if (abs(*iter - keypoints[i].pt.x) < blobSize) {//blob size보다 작은경우에는 기존라인과 동일한 좌표로 간주
            isNew = false;
            break;
        }
    }
    if (isNew) {//blob size보다 큰 경우에만 새로운 가로라인 생성=>좌표값 vector에 넣음
        coordinateX.push_back((int)keypoints[i].pt.x);
    }

    isNew = true;
    for (vector<int>::iterator iter = coordinateY.begin(); iter < coordinateY.end(); ++iter) {
        if (abs(*iter - keypoints[i].pt.y) < blobSize) {//blob size보다 작은경우에는 기존라인과 동일한 좌표로 간주
            isNew = false;
            break;
        }
    }
    if (isNew) {//blob size보다 큰 경우에만 새로운 세로라인 생성=>좌표값 vector에 넣음
        coordinateY.push_back((int)keypoints[i].pt.y);
    }
}
sort(coordinateX.begin(), coordinateX.end());
sort(coordinateY.begin(), coordinateY.end());
#ifdef OPTIONAL_PROC
    // draw coordinate lines for displaying
    Mat coordinateImg = detectedImg.clone();
    for (int i = 0; i < coordinateX.size(); ++i) {//vector인 coordinateX값들로 파란색의 가로라인 그림
        line(coordinateImg, Point(coordinateX[i], 0), Point(coordinateX[i], coordinateImg.rows), Scalar(255, 0, 0));
    }
    for (int i = 0; i < coordinateY.size(); ++i) {//vector인 coordinateY값들로 초록색의 세로라인 그림
        line(coordinateImg, Point(0, coordinateY[i]), Point(coordinateImg.cols, coordinateY[i]), Scalar(0, 255, 0));
    }
    imshow("(4)coordinateImg", coordinateImg);
#endif
```



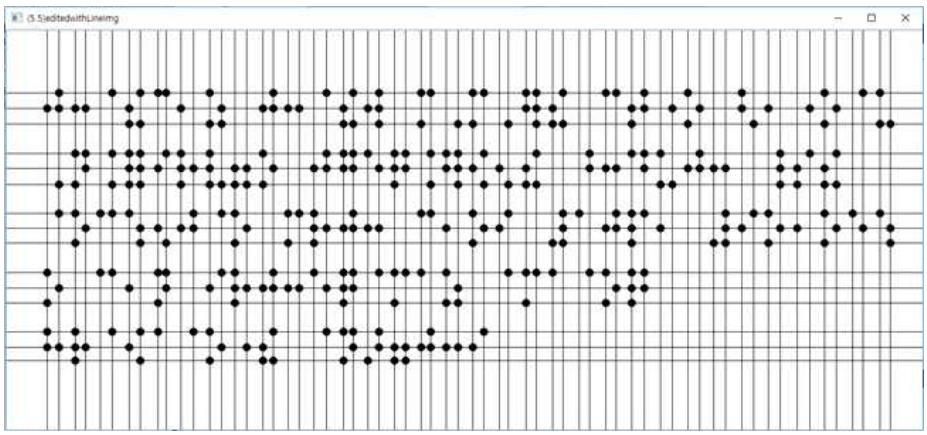
6. 깔끔한 영상처리를 위해 이전 단계와 같은 크기의 새 이미지를 생성한다. 그리고 검출 된 Blob들 각각의 위치와 동일한 위치에 평균 Blob크기의 검은 원을 그린다.

```
// 5) move keypoints to the nearest coordinate point
//빨간테두리로 표시한 keypoints들을 지움
for (int i = 0; i < keypoints.size(); ++i) {
    int ditanceX = detectedImg.cols / 2;
    int ditanceY = detectedImg.rows / 2;
    int tempX = 0;
    int tempY = 0;
    for (int j = 0; j < coordinateX.size(); ++j) {
        if (ditanceX > abs(keypoints[i].pt.x - coordinateX[j])) {
            ditanceX = abs(keypoints[i].pt.x - coordinateX[j]);
            tempX = coordinateX[j];
        }
    }
    keypoints[i].pt.x = tempX;

    for (int j = 0; j < coordinateY.size(); ++j) {
        if (ditanceY > abs(keypoints[i].pt.y - coordinateY[j])) {
            ditanceY = abs(keypoints[i].pt.y - coordinateY[j]);
            tempY = coordinateY[j];
        }
    }
    keypoints[i].pt.y = tempY;
}

// make image from the edited keypoint set(draw line for display)
Mat editedImg = Mat(detectedImg.size(), CV_8UC1);
editedImg.setTo(255);//다시 blob들을 깔끔하게 보기위한 새로운 editedImg이미지
for (int i = 0; i < keypoints.size(); ++i) {//blob들을그려줌
    circle(editedImg, Point(keypoints[i].pt.x, keypoints[i].pt.y), blobSize / 2, Scalar(0), -1, LINE_AA);
}
imshow("(5)editedImg", editedImg);
```



7. 위 이미지에 다시 가로, 세로라인을 그려 같은 문자인지 다른 문자인지를 판단한다.

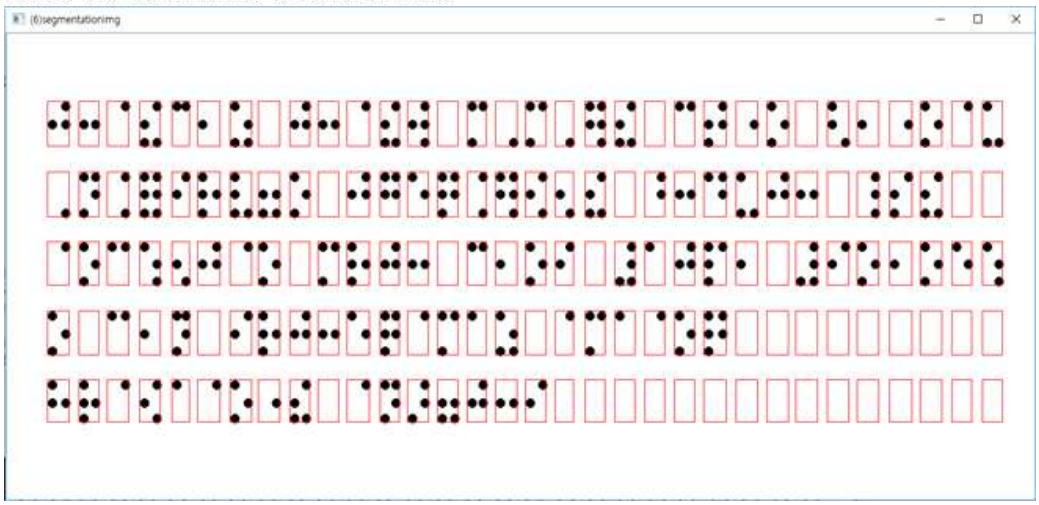
```
#ifdef OPTIONAL_PROC
        Mat editedwithLineImg = editedImg.clone();
        for (int i = 0; i < lineX.size(); ++i) {
            line(editedwithLineImg, Point(lineX[i], 0), Point(lineX[i], editedwithLineImg.rows), Scalar(0));
        }
        for (int i = 0; i < coordinateY.size(); ++i) {
            line(editedwithLineImg, Point(0, coordinateY[i]), Point(editedwithLineImg.cols, coordinateY[i]), Scalar(0));
        }
        imshow("(5.5)editedwithLineImg", editedwithLineImg);
#endif
```



8. Segmentation과정을 통해, Blob들을 3행 2열 씩 Rect구조체로 묶는다. 이 구조체에는 각각의 점자문자에 3x2의 데이터를 decimal로 변환한 Value값이 담긴다.

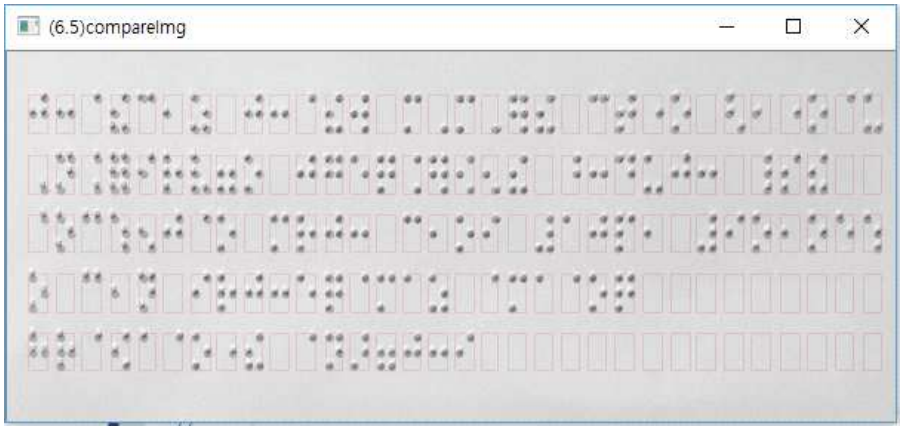

```
// 6) segmentation braille rectangle
int startXPos = 0;
int index = 0;
vector<braille> brailleSet; //braille구조체
Mat segmentationImg = Mat(editedImg.size(), CV_8UC3);   //5)단계에서 만든 blob이미지사용
cvtColor(editedImg, segmentationImg, CV_GRAY2BGR);
if ((coordinateX[1] - lineX[0]) > (lineX[2] - lineX[1])) {
    startXPos = 1;
}
for (int i = 0; i < coordinateY.size() - 2; i += 3) {//3행
    for (int j = startXPos; j < lineX.size() - 1; j += 2) {//2열씩 묶음으로 loop돌림
        braille tempBraille;
        Rect rect = Rect(Point(lineX[j] - blobSize / 2, coordinateY[i] - blobSize / 2),
            Point(lineX[j + 1] + blobSize / 2, coordinateY[i + 2] + blobSize / 2));
        int value = 0;
        rectangle(segmentationImg, rect, Scalar(0, 0, 255));//3행 2열씩 묶어 빨간색 rect로 표시
```

```
        // set the braille value(2x3 matrix)
        for (int k = 0; k < 2; ++k) {
            for (int l = 0; l < 3; ++l) {
                if (editedImg.at<uchar>(Point((int)lineX[j + k], (int)coordinateY[i + l])) == 0) {
                    value++;
                }
                value = value << 1;
            }
        }
        value = value >> 1;
        tempBraille.rect = rect;
        tempBraille.index = index++;
        tempBraille.value = value;
        brailleSet.push_back(tempBraille);//3행 2열씩 묶은 tempBraille들을 braille구조체인 brailleSet에 넣음
    }
}
if (brailleSet.empty()) {
    wcout << "there is no braille set !!" << endl;
    return 1;
}
imshow("(6)segmentationImg", segmentationImg);
```



9. addWeighted함수를 사용하여 원래 이미지와 매칭 시켜본다.

```
#ifdef OPTIONAL_PROC
        // image of comparision with the input image
        Mat compareImg, resizedImg;
        cvtColor(inputImg, compareImg, CV_GRAY2BGR);
        resize(segmentationImg, resizedImg, segmentationImg.size() / 2);
        addWeighted(compareImg, 0.8, resizedImg(Rect(MEASURE_MARGIN / 2, MEASURE_MARGIN / 2, inputImg.cols, inputImg.rows)), 0.2, 0.0, compareImg);
        imshow("(6.5)compareImg", compareImg);
#endif
```



10. 마지막으로, 각각의 구조체 위에 점자의 유무를 0과 1로 표현한 이진수와 십진수를 출력시키며, 한글변환을 시작하는 함수를 호출한다.

```
// 7) make result image
Mat resultImg = Mat(Size(segmentationImg.size()), CV_8UC3); //윗 단계와 동일한 사이즈의 결과이미지 생성
resultImg.setTo(255);
addWeighted(resultImg, 0.8, segmentationImg, 0.2, 0.0, resultImg);
int intFontFace = CV_FONT_HERSHEY_SIMPLEX; int intFontThickness = (int)round(dblFontScale * 2);
double dblFontScale = brailleSet[0].rect.size().width / 60.0;

for (int i = 0; i < brailleSet.size(); ++i) {//tempBraille들이 담긴 braille구조체 하나씩 loop돌림
    Point center, bottomLeft;
    center = (brailleSet[i].rect.tl() + brailleSet[i].rect.br()) / 2; //각tempBraille들의 가운데
    center.x -= getTextSize(int_to_string(brailleSet[i].value), intFontFace, dblFontScale, intFontThickness, 0).width / 2;
    center.y += getTextSize(int_to_string(brailleSet[i].value), intFontFace, dblFontScale, intFontThickness, 0).height / 2;
    bottomLeft = Point(brailleSet[i].rect.tl().x, brailleSet[i].rect.br().y);
    bottomLeft.x -= blobSize / 2;
    bottomLeft.y += getTextSize(bitset<6>(brailleSet[i].value).to_string(), intFontFace, dblFontScale * 0.7,
                                intFontThickness * 0.7, 0).height / 2 + blobSize / 2;
    //각tempBraille들의 중앙에 decimal값 출력
    putText(resultImg, int_to_string(brailleSet[i].value), center, intFontFace, dblFontScale, Scalar(255, 0, 0), intFontThickness);
    //각tempBraille들의 아래에 6bit의 이진코드 출력
    putText(resultImg, bitset<6>(brailleSet[i].value).to_string(), bottomLeft, intFontFace,
                                dblFontScale * 0.7, Scalar(0, 0, 0), intFontThickness * 0.7);

    C.startTranslate(brailleSet[i].value); Value값을 한글로 변환하는 함수 호출
}
wcout << L"Result = ";
C.printText(true);  변환된 글자를 출력시키는 함수 호출
imshow("(7)resultImg", resultImg);

C.SayTTS(C.printToUTF8(C.UnicodeText)); 글자를 사운드로 출력하는 함수 호출
waitKey(0);
return 0;
}
```

(2) 한글변환 과정

● 먼저, main함수에서 DTC클래스의 변수를 선언해 DTC생성자 함수를 실행시키고 SET_LINUX 혹은 SET_WIDOWS변수를 전달하여 개발환경을 알리면 DTC생성자 함수는 개발환경에 맞는 언어를 설정한다.

```
DTC::DTC(int i) {
    switch (i) {
    case 0:
        setlocale(LC_ALL, "korean");        // Windows 환경
        break;
    case 1:
        setlocale(LC_ALL, "ko_KR.UTF-8");   // Linux환경
        break;
    default:
        break;
    }
}
```

1. main함수에서 DTC클래스의 startTranslate함수를 호출하면 이 함수는 한글변환을 시작시킨다. 먼저, checkType함수에 Flag와 Index의 주소 값을 넘겨주고 그 값을 전달받는다.

```
void DTC::startTranslate(int v) {
    int Flag;
    int Index;
    DotValue = v;
    checkType(&Flag, &Index);   2)checkType함수에 Flag와 Index의 주소 값을 넘겨주고 그 값을 전달받음
    exchangeChar(Flag, Index);  3)exchangeChar함수에 Flag와 Index값을 넘겨주고 문자로 변환시킴
}
```

-Flag : 해당 점자가 어떤 의미를 지닌 점자인지 분류함

Flag	1	2	3	4	5	6	7
	초성	중성	종성	문장부호	숫자	띄어쓰기	약어

-Index : 각 Flag에서 문자들의 순서

ex)Flag=1(초성일 때)

	ㄱ	ㄲ	ㄴ	ㄷ	ㄸ	ㄹ	ㅁ	ㅂ	ㅃ	ㅅ	ㅆ	ㅇ	ㅈ	ㅉ	ㅊ	ㅋ	ㅌ	ㅍ	ㅎ
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

ex)Flag=2(중성일 때)

	ㅏ	ㅐ	ㅑ	ㅒ	ㅓ	ㅔ	ㅕ	ㅖ	ㅗ	ㅘ	ㅙ	ㅚ	ㅛ	ㅜ	ㅝ	ㅞ	ㅟ	ㅠ	ㅡ	ㅢ	ㅣ
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

2. checkType함수에서 각 문자들의 Flag와 Index값을 계산하여 다시 startTranslate함수에 반환한다.

```
switch (DotValue)
{
    // 해당 점자가 어떤 의미를 지닌 점자인지 분류 (flag 값을 통해)
    // 띄어쓰기(flag = 6)
case 0x00:  // 000000 : 빈 공간(띄어쓰기)
    flag = 6;
    index = -1;
    break;
    // 초성(flag = 1)
    // 순서 : ㄱ, ㅋ, ㄴ, ㄷ, ㄸ, ㄹ, ㅁ, ㅂ, ㅃ, ㅅ, ㅆ, ㅇ, ㅈ, ㅉ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ
case 0x04:  // 000100 : ㄱ
    flag = 1;
    index = 0;
    break;
    //case 'ㅋ' index = 1
case 0x24:  // 100100 : ㄴ, 숫자 3
    flag = 1;
    index = 2;
    break;
case 0x14:  // 010100 : ㄷ, 숫자 9
    flag = 1;
    index = 3;
    break;
    // case 'ㄸ' index = 4
case 0x02:  // 000010 : ㄹ
    flag = 1;
    index = 5;
    break;
case 0x22:  // 100010 : ㅁ, 숫자 5
    flag = 1;
    index = 6;
    break;
```

.....중간 생략

```
    flag = 7;
    index = 10;
    break;
case 0x37:  // 110111 : 영
    flag = 7;
    index = 11;
    break;
case 0x1d:  // 011101 : 를
    flag = 7;
    index = 12;
    break;
case 0x2b:  // 101011 : 은
    flag = 7;
    index = 13;
    break;
case 0x07:  // 000111 : '것' 앞 부분
    flag = 7;
    index = 20;
    break;

default:
    flag = 6;
    index = -1;
    break;
}

*AddressFlag = flag;
*AddressIndex = index;

return;
}
```

3. startTranslate함수는 Flag, Index값을 exchanegeChar함수에 전달한다. 점자는 이전의 문자에 따라 다른 뜻을 가지는데 exchanegeChar함수는 이전 문자와 조합하는 역할을 해준다.

```
wchar_t DTC::exchangeChar(int flag, int index) {
    wchar_t tmp;

    // 숫자일 경우
    if (BeforeFlag == 5) {   // 이전에 '숫자표'가 입력되었을 경우
        switch (DotValue) {
        case 0x16:  // 010110 : 숫자 0 or 초성 ㅎ
            index = 200;
            break;
        case 0x20:  // 100000 : 숫자 1 or 종성 ㄱ
            index = 201;
            break;
        case 0x30:  // 110000 : 숫자 2 or 종성 ㅂ
            index = 202;
            break;
        case 0x24:  // 100100 : 숫자 3 or 초성 ㄴ
            index = 203;
            break;
        case 0x26:  // 100110 : 숫자 4 or 초성 ㅍ
            index = 204;
            break;
        case 0x22:  // 100010 : 숫자 5 or 초성 ㅁ
            index = 205;
            break;
        case 0x34:  // 110100 : 숫자 6 or 초성 ㅋ
            index = 206;
            break;
        case 0x36:  // 110110 : 숫자 7 or 약어 '운'
            index = 207;
            break;
        case 0x32:  // 110010 : 숫자 8 or 초성 ㅌ
            index = 208;
            break;
        }
        if ((index >= 200) && (index < 210)) {
            index -= 200; // 숫자였다면 숫자 그대로 출력
                          /* 숫자일 때 index에 200 을 붙였다 없애주는 이유는
                           * 단순히 숫자값 0~9로 하게 되면 초,중,종성의 index값과 혼동이 있을 수 있으므로
                           * 분명히 숫자임을 표현하기 위해 이렇게 임시조치함 */
            tmp = L'0' + index;
            UnicodeText.push_back(tmp);
            return tmp;
        }
```

.....중간 생략

```
        tmp = makeUnicode(unmakeUnicode(tmp, 1), 6, 8);
        break;
    case 10:  // 111110 : 인
        tmp = makeUnicode(unmakeUnicode(tmp, 1), 20, 4);
        break;
    case 11:  // 110111 : 영
        tmp = makeUnicode(unmakeUnicode(tmp, 1), 6, 21);
        break;
    case 12:  // 011101 : 를
        tmp = makeUnicode(unmakeUnicode(tmp, 1), 18, 8);
        break;
    case 13:  // 101011 : 은
        tmp = makeUnicode(unmakeUnicode(tmp, 1), 18, 4);
        break;
    }
    }
    UnicodeText.push_back(tmp); // 앞에서 완성한 약어를 붙여줌
    resetFlag();
    break;
default:
    // 여기로 분기되었다면 알 수 없는 점자 값임.
    // 일단은 띄어쓰기 처리
    UnicodeText.push_back(L' ');
    resetFlag();
    break;
}

// 지금 입력된 값을 Before 값에 저장
BeforeFlag = Flag;
BeforeIndex = index;
return L'굿';
}
```

4. exchanegeChar함수가 makeUnicode함수를 호출하면 초성, 중성, 종성 순서를 입력받아 완성된 유니코드 글자를 반환한다.

```
wchar_t DTC::makeUnicode(int cho, int jung, int jong) {
    // 초성, 중성, 종성의 순서를 입력 받아 완성된 유니코드 글자로 반환해주는 함수
    /*
    참고 - 초, 중, 종성 순서
    초성 : ㄱ, ㄲ, ㄴ, ㄷ, ㄸ, ㄹ, ㅁ, ㅂ, ㅃ, ㅅ, ㅆ, ㅇ, ㅈ, ㅉ, ㅊ, ㅋ,  ㅌ, ㅍ, ㅎ
    중성 : ㅏ, ㅐ, ㅑ, ㅒ, ㅓ, ㅔ, ㅕ, ㅖ, ㅗ, ㅘ, ㅙ, ㅚ, ㅛ, ㅜ, ㅝ, ㅞ,  ㅟ, ㅠ, ㅡ, ㅢ, ㅣ
    종성 : (받침없음), ㄱ, ㄲ, ㄳ, ㄴ, ㄵ, ㄶ, ㄷ, ㄹ, ㄺ, ㄻ, ㄼ, ㄽ, ㄾ,  ㄿ, ㅀ, ㅁ, ㅂ, ㅄ, ㅅ, ㅆ, ㅇ, ㅈ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ
    */

    wchar_t base = L'가';
    wchar_t result;

    //원하는 글자 = '가' + (((초성순서 * 21) + 중성순서) * 28) + 종성순서
    result = base + (((cho * 21) + jung) * 28) + jong;

    //wcout << L"makeUnicode : " << w << endl;

    return result;
}
```

5. exchanegeChar함수에서 쌍받침, 약어 등의 경우, UnmakeUnicode함수에 글자와 원하는 타입(초성, 중성, 종성)을 전달하면 해당 글자에서 해당 타입을 분리해 반환하고, 이 값을 사용해 초성, 중성, 종성의 순서를 makeUnicode함수로 보내면 makeUnicode함수는 이를 유니코드 글자로 완성시켜준다.

```
int DTC::unmakeUnicode(wchar_t c, int type)
{   // 입력받은 글자에서 원하는 초성, 중성, 종성을 해체할 수 있게 도와주는 함수
    // 참고 사이트 : http://soooprmx.com/wp/archives/2165
    // type = 1~3의 값을 가짐. 각각 초, 중, 종성을 반환하도록 도와줌.

    int cho, jung, jong;
    /*
    * 초성 = ((문자코드 - 0xAC00) / 28) / 21
    * 중성 = ((문자코드 - 0xAC00) / 28) % 21
    * 종성 = (문자코드 - 0xAC00) % 28
    */
    if ((c < 0xAC00) || (c > 0xD7AF)) {
        return -1;
    }

    cho = ((c - L'가') / 28) / 21;
    jung = ((c - L'가') / 28) % 21;
    jong = (c - L'가') % 28;

    switch (type)
    {
    case 1:
        return cho;
    case 2:
        return jung;
    case 3:
        return jong;
    }
    return -1;
}
```

ex)exchangeChar함수에서 UnmakeUnicode 호출

```
case 7:
    // 약어일 때 (flag = 7)
    // 순서 : 억, 옹, 훌, 욱, 연, 운, 온, 언, 엘, 열, 인, 영, 흘, 은
        switch (index) {
        case 0:     // 억
            tmp = makeUnicode(unmakeUnicode(tmp, 1), 4, 1);
            break;
        case 1:  // 111111 : 옹
            tmp = makeUnicode(unmakeUnicode(tmp, 1), 8, 21);
            break;
        case 2:  // 111101 : 훌
            tmp = makeUnicode(unmakeUnicode(tmp, 1), 13, 8);
            break;
        case 3:  // 101101 : 욱
            tmp = makeUnicode(unmakeUnicode(tmp, 1), 8, 1);
            break;
        case 4:  // 100001 : 연
            tmp = makeUnicode(unmakeUnicode(tmp, 1), 6, 4);
            break;
```

6. 한글변환과정이 완료되고, main함수에서 printText함수를 호출하면 vector<wchar_t> 형태인 UnicodeText가 터미널에 출력된다.

```
void DTC::printText(bool Enter) {
    for (vector<wchar_t>::size_type i = 0; i < UnicodeText.size(); i++)
        wcout << UnicodeText[i];
    if (Enter)
        wcout << endl;
}
```

```
Result = 한글날은 한글의 우수성을 널리 알리고세종대왕이 훈민정음을 반포한 것을
기념하기 위한 날이다 10월 9일이며이 날에 대한민국은 국기인 태극기를 게양한다
```

7. main함수에서 SayTTS함수를 호출하면 TTS를 통해 점자를 한글로 변환한 결과가 소리로 출력된다.

main함수에서 C.SayTTS(C.printToUTF8(C.UnicodeText)); 호출

TTS를 위해 wstirng(wchar_t)을 string(char [])형태로 변환

```
void DTC::SayTTS(std::string Say) {
    std::string Base = "./speech.sh ";
    std::string msg;
    msg.clear();

    msg += Base;
    msg += Say;
    std::wcout<<std::endl;
    std::cout << msg << std::endl;

    system(msg.c_str());
}
```

System함수로 ./speech.sh **Say** 실행.

speech.sh파일을 사용해 사운드 출력됨.

```
std::string DTC::printToUTF8(vector<wchar_t> input) {
    std::string output;
    wchar_t Uni;
    char Utf8[3];
    output.clear();
    for (int i = 0; i < input.size(); i++) {
        Uni = input[i];
        Utf8[0] = 0xE0;
        Utf8[1] = 0x80;
        Utf8[2] = 0x80;

        if ((Uni < 0xAC00) || (Uni > 0xD7A3)) {
            // 한글의 범위를 벗어났다면 Unicode의 1byte값과 동일
            Utf8[2] = Uni & 0xff;
            output += Utf8[2];
        }
        else {
            // 한글이라면 Unicode를 UTF-8에 맞게 저장
            Utf8[0] |= (Uni & 0xF000) >> 12;
            Utf8[1] |= (Uni & 0x0FC0) >> 6;
            Utf8[2] |= (Uni & 0x003f);
            output += Utf8;
        }
    }
    return output;
}
```

<speech.sh - 네이버tts API사용>

```
*speech.sh
파일(F) 편집(E) 검색(S) 설정(O) 도움말(H)
#!/bin/bash
say() {
curl  -d "speaker=mijin&speed=0&text=$*" "https://openapi.naver.com/v1/voice/tts.bi
-H "Content-Type: application/x-www-form-urlencoded" -H "X-Naver-Client-Id: Jp6Wypr7
yf8xxHL31Rj" -H "X-Naver-Client-Secret: y30cufn0wk" > result.mp3
local IFS=+;/usr/bin/mplayer -ao alsa -really-quiet -noconsolecontrols result.mp3 &
}
say $*
```

4. 결과

결과적으로, 터미널화면에 점자이미지가 한글로 변환된 결과가 출력되며 해당 글자가 TTS기능을 통해 사운드로 출력된다.

가. 한계점

1. 점자를 한글로 번역했을 때의 오역

   6점 점자의 단점 중 두 번째로 인해 발생하는 문제이다. 6점 점자는 6점 안에 모든 내용을 표현해야한다. 이를 위해 여러 표현이 하나의 6점 점자에 중복되기도 하는데, 이는 번역의 결과물에 오역을 불러온다. 실제 사람이 읽는 경우 상황과 문맥에 따라 오역을 피하지만, 해당 프로그램은 이러한 능력이 없기 때문에 오역이 발생할 수 있다. 이를 해결하기 위해선 문맥을 파악해야하기 때문에, 수준 높은 기술력이 요구된다. 졸업작품은 이러한 중복의 문제를 피하기 위해 한글만을 번역할 수 있으며, 영어나 문장부호에 대해 번역할 수 없다.

2. TTS(Text to Speech)의 자연스러운 발음 한계

글을 읽어주는 기능은 TTS API를 통해 소리로 출력하고 있다. 그러나 프로그램의 번역 결과와 TTS의 기능적인 한계로 인해 자연스러운 발음이 이뤄지지 않고 있다.

나. 소감

위의 한계점을 해결하지 못한 것은 아쉬우나, 후천적 시각장애인의 교육 목적을 위한 도구로서의 가능성을 보여주었고, 점자의 의미를 컴퓨터를 통해 이해하고 해결할 수 있는 과정을 보여줄 수 있어 다행이다.

이를 위해 작품 방향을 지도해주신 멘토교수 손용락, 김재현 교수님과, 점자 관련 지문을 도와주신 강북점자도서실 김지혜 사서님께 감사의 말씀을 드린다.

5. 부록 - Full Code

<main.cpp>

```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <opencv/cv.h>
#include <iostream>
#include <string>
#include <bitset>
#include "DTC.h"

#define OPTIONAL_PROC
#define MEASURE_MARGIN  20
#define PARAMS_MIN_SIZE 2.0f
#define PARAMS_MAX_SIZE 30.0f
#define SET_WINDOWS 0
#define SET_LINUX   1
using namespace std;
using namespace cv;

typedef struct _braille {
    _braille() : rect(), value(0), index(0) {}
    virtual ~_braille() {}
    Rect rect;
    int value;  // decimal
    int index;
}braille
string int_to_string(int value);

int main(int argc, char** argv) {
    DTC C(SET_LINUX);
// 1) read input image 점자 이미지 읽어오기
    Mat inputImg;
    inputImg = imread(argv[1]);
    cvtColor(inputImg, inputImg, CV_RGB2YCrCb);
    vector<cv::Mat> channels;        // 채널 분리를 위해
    cv::split(inputImg, channels);   // 채널 나누기
```

```
    inputImg = channels[0];    // Y(휘도)성분 사용 – 빛에 민감하기 위해 (양각, 음각)
    imshow("(1)inputImg", inputImg);

// 2) pre-processing input image 이미지 전처리
    Mat thresholdImg;
    adaptiveThreshold(inputImg,    thresholdImg,    255,    CV_ADAPTIVE_THRESH_MEAN_C,
CV_THRESH_BINARY, 21, 10); // 이진화

//apply gaussian blur & erode and then threshold again
    Mat morphologyElement3x3 = getStructuringElement(MORPH_RECT, Size(3, 3));
    // 침식에 사용할 사각형 마스크
    GaussianBlur(thresholdImg, thresholdImg, Size(3, 3), 0)
    // 가우시안 블러링 – 잡음 제거
    erode(thresholdImg, thresholdImg, morphologyElement3x3);// 침식 – 디테일 강화
    threshold(thresholdImg, thresholdImg, 21, 255, CV_THRESH_BINARY);// 대비 증가
    imshow("(1.5)thresholdImg", thresholdImg);

// make margin and resize from original image
    Mat measureImg = Mat(Size(thresholdImg.cols + MEASURE_MARGIN, thresholdImg.rows +
MEASURE_MARGIN), CV_8UC1, 255);    // margin 추가
    thresholdImg.copyTo(measureImg(Rect(MEASURE_MARGIN  /  2,  MEASURE_MARGIN  /  2,
thresholdImg.cols, thresholdImg.rows)));
    resize(measureImg, measureImg, measureImg.size() * 2);    // 리사이징
    imshow("(2)measureImg", measureImg);

#ifdef OPTIONAL_PROC
// 2.5) deskewing image 기울임 보정
    Mat deskewImg = measureImg;
    vector<Point> points;
    for (int i = 0; i < deskewImg.cols; ++i)
        for (int j = 0; j < deskewImg.rows; ++j)
            if (!deskewImg.at<uchar>(Point(i, j)))
                points.push_back(Point(i, j));

    RotatedRect box = minAreaRect(Mat(points));
    double angle = box.angle;
    if (angle < -45.0)
        angle += 90.0;
    wcout << "angle : " << angle << endl;
```

```
    Mat rotatedImg = getRotationMatrix2D(box.center, angle, 1);
    warpAffine(deskewImg, deskewImg, rotatedImg, deskewImg.size(), INTER_CUBIC);
    imshow("(2.5)deskewImg", deskewImg);
#endif

// 3) detect blobs (점자 blob 찾기) => opencv의 SimpleBlobDetector 사용
    SimpleBlobDetector::Params params;
    params.filterByArea = true
    params.minArea = PARAMS_MIN_SIZE * PARAMS_MIN_SIZE
    params.maxArea = PARAMS_MAX_SIZE * PARAMS_MAX_SIZE

    // Change thresholds
    params.minThreshold = 10;
    params.maxThreshold = 200;

    // Filter by Circularity
    params.filterByCircularity = true;
    params.minCircularity = 0.1;

    // Filter by Convexity
    params.filterByConvexity = true;
    params.minConvexity = 0.87;

    // Filter by Inertia
    params.filterByInertia = true;
    params.minInertiaRatio = 0.01;

    Ptr<SimpleBlobDetector> blobDetector = SimpleBlobDetector::create(params);
    vector<KeyPoint> keypoints;
    blobDetector->detect(measureImg, keypoints);//blobvectorkeypoints

    if (keypoints.empty()) {
        // keypoints가 없다면
        wcout << "there is no braille existance condition" << endl;
        return 1;
}
    Mat detectedImg = Mat(measureImg.size(), CV_8UC3);
    // 검출된 blob들을 담은 벡터 keypoints를 빨간색테두리로 표시
    drawKeypoints(measureImg,    keypoints,    detectedImg,    Scalar(0,    0,    255),
```

```
DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
    imshow("(3)detectedImg", detectedImg);

// 4) normalize keypoints to coordinate line set
    vector<int> coordinateX;    // vector (가로줄, 점의 위치만 검출)
    vector<int> coordinateY;    // vector (세로줄)
    vector<int> lineX;          // vector (가로줄, 점이 없는 위치도 검출)
    float blobSize = 0.0f;
    for (int i = 0; i < keypoints.size(); ++i) { // 모든 blob 크기를 blobSize에 더함.
        blobSize += keypoints[i].size;
    }
    blobSize /= keypoints.size(); // blob 수로 나누어 blob 평균 크기 확인
    wcout << "mean of the blob sizes : " << blobSize << endl; // blob 평균 크기 출력

    for (int i = 0; i < keypoints.size(); ++i) {
        bool isNew = true;
        for    (vector<int>::iterator    iter    =    coordinateX.begin();    iter    <
coordinateX.end(); ++iter) {
            if (abs(*iter - keypoints[i].pt.x) < blobSize) {
                //blob size보다 작은 경우에는 기존 라인과 동일한 좌표로 간주
                isNew = false;
                break;
            }
        }
        if (isNew) {
            //blob size보다 큰 경우에만 새로운 가로 라인 생성 => 좌표값 vector에 넣음
            coordinateX.push_back((int)keypoints[i].pt.x);
        }
        isNew = true;
        for    (vector<int>::iterator    iter    =    coordinateY.begin();    iter    <
coordinateY.end(); ++iter) {
            if (abs(*iter - keypoints[i].pt.y) < blobSize) {
                //blob size보다 작은경우에는 기존라인과 동일한 좌표로 간주
                isNew = false;
                break;
            }
        }
        if (isNew) {
            //blob size보다 큰 경우에만 새로운 세로라인 생성=>좌표값 vector에 넣음
```

```
            coordinateY.push_back((int)keypoints[i].pt.y);
        }
    }
    sort(coordinateX.begin(), coordinateX.end());
    sort(coordinateY.begin(), coordinateY.end());

#ifdef OPTIONAL_PROC
// draw coordinate lines for displaying
    Mat coordinateImg = detectedImg.clone();
    for (int i = 0; i < coordinateX.size(); ++i) {
        //vector인 coordinateX값들로 파란색의 가로 라인 그림
        line(coordinateImg,     Point(coordinateX[i],     0),     Point(coordinateX[i],
coordinateImg.rows), Scalar(255, 0, 0));
    }
    for (int i = 0; i < coordinateY.size(); ++i) {
        //vector인 coordinateY값들로 초록색의 세로 라인 그림
        line(coordinateImg,    Point(0,    coordinateY[i]),    Point(coordinateImg.cols,
coordinateY[i]), Scalar(0, 255, 0));
    }
    imshow("(4)coordinateImg", coordinateImg);
#endif

// 5) move keypoints to the nearest coordinate point
    //빨간테두리로 표시한 keypoints들을 지움
    for (int i = 0; i < keypoints.size(); ++i) {
        int ditanceX = detectedImg.cols / 2;
        int ditanceY = detectedImg.rows / 2;
        int tempX = 0;
        int tempY = 0;
        for (int j = 0; j < coordinateX.size(); ++j) {
            if (ditanceX > abs(keypoints[i].pt.x - coordinateX[j])) {
                ditanceX = abs(keypoints[i].pt.x - coordinateX[j]);
                tempX = coordinateX[j];
            }
        }
    keypoints[i].pt.x = tempX;

        for (int j = 0; j < coordinateY.size(); ++j) {
            if (ditanceY > abs(keypoints[i].pt.y - coordinateY[j])) {
```

```
                ditanceY = abs(keypoints[i].pt.y - coordinateY[j]);
                tempY = coordinateY[j];
            }
        }
        keypoints[i].pt.y = tempY;
    }

// make image from the edited keypoint set(draw line for display)
    Mat editedImg = Mat(detectedImg.size(), CV_8UC1);
    editedImg.setTo(255);    // blob을 깔끔하게 보여주기위한 새로운 editedImg
    for (int i = 0; i < keypoints.size(); ++i) {
        // blob들을 반듯한 원으로 다시 그려줌
        circle(editedImg, Point(keypoints[i].pt.x, keypoints[i].pt.y), blobSize / 2,
Scalar(0), -1, LINE_AA);
    }
    imshow("(5)editedImg", editedImg);

#ifdef OPTIONAL_PROC
    Mat editedwithLineImg = editedImg.clone();
    for (int i = 0; i < lineX.size(); ++i) {
        line(editedwithLineImg,        Point(lineX[i],        0),        Point(lineX[i],
editedwithLineImg.rows), Scalar(0));
    }
    for (int i = 0; i < coordinateY.size(); ++i) {
        line(editedwithLineImg, Point(0, coordinateY[i]), Point(editedwithLineImg.cols,
coordinateY[i]), Scalar(0));
    }
    imshow("(5.5)editedwithLineImg", editedwithLineImg);
#endif

// 6) segmentation braille rectangle
    int startXPos = 0;
    int index = 0;
    vector<braille> brailleSet; //braille 구조체
    Mat segmentationImg = Mat(editedImg.size(), CV_8UC3);
    // 5)단계에서 만든 blob 이미지 사용
    cvtColor(editedImg, segmentationImg, CV_GRAY2BGR);
    if ((coordinateX[1] - lineX[0]) > (lineX[2] - lineX[1])) {
        startXPos = 1;
```

```
    }
    for (int i = 0; i < coordinateY.size() - 2; i += 3) {        // 3행
        for (int j = startXPos; j < lineX.size() - 1; j += 2) {  //2열씩 묶음으로 loop
            braille tempBraille;
            Rect rect = Rect(Point(lineX[j] - blobSize/2, coordinateY[i] - blobSize/2),
Point(lineX[j + 1] + blobSize / 2, coordinateY[i + 2] + blobSize / 2));
            int value = 0;
            rectangle(segmentationImg, rect, Scalar(0, 0, 255));
            //3행 2열씩 묶어 빨간색 rect로 표시

            // set the braille value(2x3 matrix)
            for (int k = 0; k < 2; ++k) {
                for (int l = 0; l < 3; ++l) {
            if (editedImg.at<uchar>(Point((int)lineX[j + k], (int)coordinateY[i + l]))
== 0) {
                        value++;
                    }
                    value = value << 1;
                }
            }
            value = value >> 1;
            tempBraille.rect = rect;
            tempBraille.index = index++;
            tempBraille.value = value;
            brailleSet.push_back(tempBraille);
            // 3행 2열씩 묶은 tempBraille들을 braille구조체인 brailleSet에 넣음
        }
    }
    if (brailleSet.empty()) {
        wcout << "there is no braille set !!" << endl;
        return 1;
    }
    imshow("(6)segmentationImg", segmentationImg);

#ifdef OPTIONAL_PROC // image of comparision with the input image
    Mat compareImg, resizedImg;
    cvtColor(inputImg, compareImg, CV_GRAY2BGR);
    resize(segmentationImg, resizedImg, segmentationImg.size() / 2);
    addWeighted(compareImg, 0.8, resizedImg(Rect(MEASURE_MARGIN / 2, MEASURE_MARGIN /
```

```
2, inputImg.cols, inputImg.rows)), 0.2, 0.0, compareImg);
    imshow("(6.5)compareImg", compareImg);
#endif

// 7) make result image
    Mat resultImg = Mat(Size(segmentationImg.size()), CV_8UC3);
    // 윗 단계와 동일한 사이즈의 결과 이미지 생성
    resultImg.setTo(255);
    addWeighted(resultImg, 0.8, segmentationImg, 0.2, 0.0, resultImg);
    int intFontFace = CV_FONT_HERSHEY_SIMPLEX;
    double dblFontScale = brailleSet[0].rect.size().width / 60.0;
    int intFontThickness = (int)round(dblFontScale * 2);

    for (int i = 0; i < brailleSet.size(); ++i) {
    //tempBraille들에 담긴 braille구조체 모두를 loop
        Point center, bottomLeft;
        center = (brailleSet[i].rect.tl() + brailleSet[i].rect.br()) / 2;
        // center = 각 tempBraille들의 중심
        center.x    -=    getTextSize(int_to_string(brailleSet[i].value),    intFontFace,
dblFontScale, intFontThickness, 0).width / 2;
        center.y    +=    getTextSize(int_to_string(brailleSet[i].value),    intFontFace,
dblFontScale, intFontThickness, 0).height / 2;
        bottomLeft = Point(brailleSet[i].rect.tl().x, brailleSet[i].rect.br().y);
        bottomLeft.x -= blobSize / 2;
        bottomLeft.y     +=     getTextSize(bitset<6>(brailleSet[i].value).to_string(),
intFontFace, dblFontScale * 0.7, intFontThickness * 0.7, 0).height / 2 + blobSize / 2;

        putText(resultImg,   int_to_string(brailleSet[i].value),   center,   intFontFace,
dblFontScale, Scalar(255, 0, 0), intFontThickness);
        // decimal 값으로 Braille Value 출력

        putText(resultImg,    bitset<6>(brailleSet[i].value).to_string(),    bottomLeft,
intFontFace, dblFontScale * 0.7, Scalar(0, 0, 0), intFontThickness * 0.7);
        // 6bit binary 값으로 Braille Value 출력 출력
        C.startTranslate(brailleSet[i].value);
    }
    wcout << L"Result = "
    C.printText(true);
    imshow("(7)resultImg", resultImg);
```

```
    C.SayTTS(C.printToUTF8(C.UnicodeText));
    waitKey(0);
    return 0;
}

string int_to_string(int value) {
    string s;
    char c;
    vector<int> st;
    while (value != 0) {
        st.push_back(value % 10);
        value /= 10;
    }
    s = "";
    while (st.size() != 0) {
        c = '0' + (char)st.back();
        s += c;
        st.pop_back();
    }
    return s;
}
```

<DTC.h>

```
#pragma once
#include <iostream>
#include <string>
#include <locale>
#include <vector>
using namespace std;
class DTC {
private:
    int BeforeIndex; // 이전에 입력된 글자의 Index 값
    int BeforeFlag;  // 이전에 입력된 글자의 Flag 값
    bool PowerFlag;  // 쌍자음 Flag
    bool NumberFlag; // 숫자 Flag
public:
    DTC();           // 미사용
    DTC(int i);      // 초기 설정. if i => 0 = Windows, 1 = Liunx
```

```
    vector<wchar_t> UnicodeText;    // 점자의 한글 변환 결과 문장
    int DotValue;    // 현재 확인 중인 Braille Value. (곧 한글로 변환될 점자)
    void startTranslate(int v);     // 한글로의 변환 시작
    wchar_t exchangeChar(int flag, int index);    //  입력된 type에 따라 글자 반환
    void checkType(int* AddressFlag, int* AddressValue);
    // Braille Value의 글자 type 확인
    void printText(bool Enter = true); // UnicodeText를 출력
    void resetFlag();// NumberFlag와 PowerFlag를 false로 초기화함.
    string printToUTF8(vector<wchar_t> input);// wstirng(wchar_t) -> string(char [])
    void SayTTS(std::string Say);// TTS를 통해 소리로 반환
    wchar_t makeUnicode(int cho, int jung, int jong);
    // 한글의 초성, 중성, 종성을 입력받아 글자를 조합
    wchar_t remakeUnicode(wchar_t c, int type, int value);
    // 한글의 초성, 중성, 종성 중 일부분을 변경함.
    int unmakeUnicode(wchar_t c, int type); // 한글자에서 초성, 중성, 종성의 값을 반환
}; // DoTConverter
```

<DTC.cpp>

```
#include "DTC.h"
#include <stdio.h>
#include <string>
#include <cstdlib>
#include <locale>
#include <wchar.h>

void DTC::startTranslate(int v) {
    int Flag;
    int Index;
    DotValue = v
    checkType(&Flag, &Index);
    exchangeChar(Flag, Index);
}

DTC::DTC(int i) {
    switch (i) {
    case 0: // Windows
        setlocale(LC_ALL,"korean");
        break;
    case 1: // Linux
```

```
        setlocale(LC_ALL, "ko_KR.UTF-8");
        break;
    default:
        break;
    }
}

void DTC::checkType(int* AddressFlag, int* AddressIndex) {
    int flag;
    int index;
    // DotValue에 대해 flag(초성, 중성, 종성, 약어 등)과 index를 분류함.
    switch (DotValue) {
```

```
case 0x00:
flag = 6;
index = -1;
break;
case 0x04:
flag = 1;
index = 0;
break;
case 0x24:
flag = 1;
index = 2;
break;
case 0x14:
flag = 1;
index = 3;
break;
```

```
case 0x02:
flag = 1;
index = 5;
break;
case 0x22:
flag = 1;
index = 6;
break;
case 0x06:
flag = 1;
index = 7;
break;
case 0x01:
flag = 1;
index = 9;
break;
case 0x05:
flag = 1;
index = 12;
break;
case 0x03:
flag = 1;
index = 14;
break;
case 0x34:
flag = 1;
index = 15;
break;
case 0x32:
flag = 1;
index = 16;
break;
case 0x26:
flag = 1;
index = 17;
break;
```

```
case 0x16:
flag = 1;
index = 18;
break;
case 0x31:
flag = 2;
index = 0;
break;
case 0x3a:
flag = 2;
index = 1;
break;
case 0x0e:
flag = 2;
index = 2;
break;
case 0x1c:
flag = 2;
index = 4;
break;
case 0x2e:
flag = 2;
index = 5;
break;
case 0x23:
flag = 2;
index = 6;
break;
case 0x0c:
flag = 2;
index = 7;
break;
case 0x29:
flag = 2;
index = 8;
break;
```

```
case 0x39:
flag = 2;
index = 9;
break;
case 0x2f:
flag = 2;
index = 11;
break;
case 0x0d:
flag = 2;
index = 12;
break;
case 0x2c:
flag = 2;
index = 13;
break;
case 0x3c:
flag = 2;
index = 14;
break;
case 0x25:
flag = 2;
index = 17;
break;
case 0x15:
flag = 2;
index = 18;
break;
case 0x17:
flag = 2;
index = 19;
break;
case 0x2a:
flag = 2;
index = 20;
break;
```

```
case 0x20:
flag = 3;
index = 1;
break;
case 0x12:
flag = 3;
index = 4;
break;
case 0x0a:
flag = 3;
index = 7;
break;
case 0x10:
flag = 3;
index = 8;
break;
case 0x11:
flag = 3;
index = 16;
break;
case 0x30:
flag = 3;
index = 17;
break;
case 0x08:
flag = 3;
index = 19;
break;
case 0x1b:
flag = 3;
index = 21;
break;
case 0x28:
flag = 3;
index = 22;
break;
```

```
case 0x18:
flag = 3;
index = 23;
break;
case 0x1a:
flag = 3;
index = 24;
break
case 0x19:
flag = 3;
index = 25;
break;
case 0x13:
flag = 3;
index = 26;
break;
case 0x0b:
flag = 3;
index = 27;
break;
case 0x09:
flag = 4;
index = 13;
break;
case 0x0f:
flag = 5;
index = -1;
break;
```

```
case 0x35:
flag = 1;
index = 0;
break;
case 0x38:
flag = 1;
index = 9;
break;
case 0x27:
flag = 7;
index = 0;
break;
case 0x3f:
flag = 7;
index = 1;
break;
case 0x3d:
flag = 7;
index = 2;
break;
case 0x2d:
flag = 7;
index = 3;
break;
case 0x21:
flag = 7;
index = 4;
break;
```

```
case 0x36:
flag = 7;
index = 5;
break;
case 0x3b:
flag = 7;
index = 6;
break;
case 0x1f:
flag = 7;
index = 7;
break;
case 0x1e:
flag = 7;
index = 8;
break;
case 0x33:
flag = 7;
index = 9;
break;
case 0x3e:
flag = 7;
index = 10;
break;
case 0x37:
flag = 7;
index = 11;
break;
```

```
case 0x1d:
flag = 7;
index = 12;
break;
case 0x2b:
flag = 7;
index = 13;
break;
case 0x07:
flag = 7;
index = 20;
break;
case 0x35:
flag = 7;
index = 14;
break;
case 0x38:
flag = 7;
index = 15;
break;
default:
flag = 6;
index = -1;
break;
```

```
    }
    *AddressFlag = flag;
    *AddressIndex = index;
    return;
}

wchar_t DTC::exchangeChar(int flag, int index) {
    wchar_t tmp;
    if (BeforeFlag == 5) {
    // 이전에 숫자가 입력되었을 경우
```

```
        switch (DotValue) {
case 0x16:          | case 0x24:          | case 0x34:          | case 0x14:
index = 200;        | index = 203;        | index = 206;        | index = 209;
break;              | break               | break               | break
case 0x20:          | case 0x26:          | case 0x36:          | default:
index = 201;        | index = 204;        | index = 207;        | break;
break;              | break;              | break;              |
case 0x30:          | case 0x22:          | case 0x32:          |
index = 202;        | index = 205;        | index = 208;        |
break;              | break;              | break;              |
        }

        if ((index >= 200) && (index < 210)) {
        // 숫자였다면
            index -= 200; // 200을 빼고 출력. 200은 숫자를 구분하기위한 정크 값.
            tmp = L'0' + index
            UnicodeText.push_back(tmp);
            return tmp;
        }
        NumberFlag = false; // 숫자가 아니라면 NumberFlag off. 그리고 작업 계속 진행
    }
    switch (flag) {
    case 1: // 초성이라면
        // 우선적으로 약어들을 처리하도록 함.
        if ((BeforeFlag == 3) && (BeforeIndex == 1) && (index == 2)) {
if (UnicodeText.size() > 0) {       |
tmp  =  remakeUnicode(UnicodeText.back(),  |
3, 0);                              |
UnicodeText.pop_back();             |  else {
UnicodeText.push_back(tmp);         |  UnicodeText.push_back(L'그‘);
UnicodeText.push_back(L'그‘);       |  UnicodeText.push_back(L'러‘);
UnicodeText.push_back(L'러‘);       |  UnicodeText.push_back(L'나‘);
UnicodeText.push_back(L'나‘);       |  }
}                                   |

        }
        else if ((BeforeFlag == 1) && (UnicodeText.size() > 0)) {
            tmp = UnicodeText.back();
```

```
            if (unmakeUnicode(tmp, 1) == 9) {
                switch (DotValue) {
case 0x04:
index = 1;
UnicodeText.pop_back();
break;
case 0x14:
index = 4;
UnicodeText.pop_back();
break;
case 0x06:
index = 8;
UnicodeText.pop_back();
break;
case 0x01:
index = 10;
UnicodeText.pop_back();
break;
case 0x05:
index = 13;
UnicodeText.pop_back();
break;

default:
break;
                }
            }
        }
        resetFlag();
        if (index == 9)
            PowerFlag = true;
        else
            PowerFlag = false;
        UnicodeText.push_back(makeUnicode(index, 0, 0));
        break;

    case 2: // 중성이라면
        if ((BeforeFlag == 2) && (index == 4))
            UnicodeText.push_back(L'것');
        else if ((BeforeFlag == 2) && (index == 7)) {
            tmp = UnicodeText.back();
            UnicodeText.pop_back();
            UnicodeText.push_back(remakeUnicode(tmp, 3, 20));
        }
        else if ((BeforeFlag == 3) && (BeforeIndex == 1) && (index == 4)) {
if (UnicodeText.size() > 0) {
tmp = remakeUnicode(UnicodeText.back(), 3, 0);
UnicodeText.pop_back();
UnicodeText.push_back(tmp);
UnicodeText.push_back(L'그');
UnicodeText.push_back(L'래');
```

```
UnicodeText.push_back(L'서');
}
else {
UnicodeText.push_back(L'그');
UnicodeText.push_back(L'래');
UnicodeText.push_back(L'서');
}
```

```
    }
    else if ((BeforeFlag == 3) && (BeforeIndex == 1) && (index == 5)) {
```

```
if (UnicodeText.size() > 0) {
tmp  =  remakeUnicode(UnicodeText.back(),
3, 0);
UnicodeText.pop_back();
UnicodeText.push_back(tmp);
UnicodeText.push_back(L'그');
UnicodeText.push_back(L'런');
UnicodeText.push_back(L'데');
}
else {
UnicodeText.push_back(L'그');
UnicodeText.push_back(L'런');
UnicodeText.push_back(L'데');
}
```

```
    }
    else if ((BeforeFlag == 3) && (BeforeIndex == 1) && (index == 6)) {
```

```
if (UnicodeText.size() > 0) {
tmp  =  remakeUnicode(UnicodeText.back(),
3, 0);
UnicodeText.pop_back();
UnicodeText.push_back(tmp);
UnicodeText.push_back(L'그');
UnicodeText.push_back(L'리');
UnicodeText.push_back(L'하');
UnicodeText.push_back(L'여');
}
else {
UnicodeText.push_back(L'그');
UnicodeText.push_back(L'리');
UnicodeText.push_back(L'하');
UnicodeText.push_back(L'여');
}
```

```
    }
    else if ((BeforeFlag == 3) && (BeforeIndex == 1) && (index == 8)) {
```

```
if (UnicodeText.size() > 0) {
tmp = remakeUnicode(UnicodeText.back(), 3, 0);
UnicodeText.pop_back();
UnicodeText.push_back(tmp);
UnicodeText.push_back(L'그');
UnicodeText.push_back(L'리');
UnicodeText.push_back(L'고');
}
```

```
else {
UnicodeText.push_back(L'그');
UnicodeText.push_back(L'리');
UnicodeText.push_back(L'고');
}
```

```
    }

    else if (BeforeFlag == 1) {
        // 초성 + 중성의 형태라면 글자를 조합, 반환
        tmp = remakeUnicode(UnicodeText.back(), 2, index);
        UnicodeText.pop_back();
        UnicodeText.push_back(tmp);
    }

    else {
        // 그렇지 않다면 중성 + 중성. 조금 더 확인할 필요가 있음.
        if (index == 1) {
            tmp = UnicodeText.back();
            UnicodeText.pop_back();
            if (unmakeUnicode(tmp, 2) == 13)       // ㅟ
                UnicodeText.push_back(remakeUnicode(tmp, 2, 16));
            else if (unmakeUnicode(tmp, 2) == 2)   // ㅒ
                UnicodeText.push_back(remakeUnicode(tmp, 2, 3));
            else if (unmakeUnicode(tmp, 2) == 9)   // ㅚ
                UnicodeText.push_back(remakeUnicode(tmp, 2, 10));
            else if (unmakeUnicode(tmp, 2) == 14)  // ㅞ
                UnicodeText.push_back(remakeUnicode(tmp, 2, 15));
            else {
                UnicodeText.push_back(tmp); // 이중모음이 아니라면
                UnicodeText.push_back(makeUnicode(11, index, 0));
                // 새로운 글자로 생각하고 이어 붙임.
            }
        }
        else {
            UnicodeText.push_back(makeUnicode(11, index, 0));
        }
    }
    break;
```

```
    case 3: // 종성이라면
        if (UnicodeText.size() == 0) {
            BeforeFlag = flag
            BeforeIndex = index
            break;
        }
        else if ((BeforeFlag == 3) && (BeforeIndex == 1) && (index == 4)) {
```

```
if (UnicodeText.size() > 0) {
tmp  =  remakeUnicode(UnicodeText.back(),
3, 0);
UnicodeText.pop_back();
UnicodeText.push_back(tmp);
UnicodeText.push_back(L'그‘);
UnicodeText.push_back(L'러‘);
UnicodeText.push_back(L'면‘);
}
```

```
else {
UnicodeText.push_back(L'그‘);
UnicodeText.push_back(L'러‘);
UnicodeText.push_back(L'면‘);
}
```

```
        }
        else if ((BeforeFlag == 3) && (BeforeIndex == 1) && (index == 16)) {
```

```
if (UnicodeText.size() > 0) {
tmp  =  remakeUnicode(UnicodeText.back(),
3, 0);
UnicodeText.pop_back();
UnicodeText.push_back(tmp);
UnicodeText.push_back(L'그‘);
UnicodeText.push_back(L'러‘);
UnicodeText.push_back(L'므‘);
UnicodeText.push_back(L'로‘);
}
```

```
else {
UnicodeText.push_back(L'그‘);
UnicodeText.push_back(L'러‘);
UnicodeText.push_back(L'므‘);
UnicodeText.push_back(L'로‘);
}
```

```
        }
```

```
else {
    tmp = UnicodeText.back();
    UnicodeText.pop_back();
    switch (unmakeUnicode(tmp, 3)) {
    case 1:
        if (index == 1)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 2));
        else if (index == 19)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 3));
        break;
    case 4: // ㄴ 받침 + 다른 자음
        if (index == 22)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 5));  // ㄴㅈ
        else if (index == 27)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 6));  // ㄴㅎ
    break;
    case 8:  // ㄹ 받침 + 다른 자음
        if (index == 1)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 9));  // ㄺ
        else if (index == 16)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 10)); // ㄻ
        else if (index == 17)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 11)); // ㄼ
        else if (index == 8)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 12)); // ㄽ
        else if (index == 25)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 13)); // ㄾ
        else if (index == 26)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 14)); // ㄿ
        else if (index == 27)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 15)); // ㅀ
        break;
    case 17: // ㅂ 받침 + 다른 자음
        if (index == 8)
            UnicodeText.push_back(remakeUnicode(tmp, 3, 18)); // ㅄ
        break;
    default:
        UnicodeText.push_back(remakeUnicode(tmp, 3, index));
        break;
```

```
        }
}
resetFlag();
break;

case 5: // 처음 숫자표가 입력될 경우
    BeforeFlag = flag
    BeforeIndex = index
    NumberFlag = true
    PowerFlag = false
    break;
case 6: // 띄어쓰기
    if (UnicodeText.size() == 0)
        break;
    else if (UnicodeText.back() != L' ')
        UnicodeText.push_back(L' ');
    resetFlag();
break;
case 7: // 약어일 때
    if (index == 20) {
        BeforeFlag = 20;
        PowerFlag = false
        NumberFlag = false
        return L'쉼‘;
    }
    else {
        if (BeforeFlag == 1) {
            tmp = UnicodeText.back();
            UnicodeText.pop_back();
        }
        else
            tmp = L'아’;
        switch (index) {
        case 0: // 억
            tmp = makeUnicode(unmakeUnicode(tmp, 1), 4, 1);
            break;
        case 1: // 옹
            tmp = makeUnicode(unmakeUnicode(tmp, 1), 8, 21);
            break;
```

```
case 2: // 올
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 13, 8);
    break;
case 3: // 옥
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 8, 1);
    break;
case 4:  // 연
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 6, 4);
    break;
case 5:  // 운
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 13, 4);
    break;
case 6:  // 온
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 8, 4);
    break;
case 7:  // 언
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 4, 4);
    break;
case 8:  // 얼
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 4, 8);
    break;
case 9:  // 열
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 6, 8);
    break;
case 10: // 인
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 20, 4);
    break;
case 11: // 영
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 6, 21);
    break;
case 12: // 을
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 18, 8);
    break;
case 13: // 은
    tmp = makeUnicode(unmakeUnicode(tmp, 1), 18, 4);
    break;
case 14: // 약자 '가'
    tmp = L'가';
    break;
```

```
            case 15: // 약자 'ㅅ'
                tmp = L'ㅅ';
                break;
            }
        }
        UnicodeText.push_back(tmp);
        resetFlag();
        break;
    default:
        UnicodeText.push_back(L' ');
        resetFlag();
        break;
    }
    BeforeFlag = flag
    BeforeIndex = index
    return L'굿';
}

void DTC::printText(bool Enter) {
    for (vector<wchar_t>::size_type i = 0; i < UnicodeText.size(); i++)
        wcout << UnicodeText[i]
    if (Enter)
        wcout << endl;
}

void DTC::resetFlag() {
    PowerFlag = false;
    NumberFlag = false;
}

wchar_t DTC::makeUnicode(int cho, int jung, int jong) {
    wchar_t base = L'가';
    wchar_t result;
    result = base + (((cho * 21) + jung) * 28) + jong
    return result;
}
```

```
wchar_t DTC::remakeUnicode(wchar_t c, int type, int value) {
    int cho, jung, jong;
    wchar_t result;
    if ((c < 0xAC00) || (c > 0xD800)) {
        return L' '  ;
    }
    cho = ((c - L'가') / 28) / 21;
    jung = ((c - L'가') / 28) % 21;
    jong = (c - L'가') % 28;

    switch (type) {
case 1:                     | case 2:                     | case 3:
result = makeUnicode(value, | result = makeUnicode(cho, v | result = makeUnicode(cho, j
jung, jong);                | alue, jong);                | ung, value);
break;                      | break                       | break;
    }
    return result;
}

int DTC::unmakeUnicode(wchar_t c, int type) {
    int cho, jung, jong;
    if ((c < 0xAC00) || (c > 0xD7AF)) {
        return -1;
    }
    cho = ((c - L'가') / 28) / 21;
    jung = ((c -L'가') / 28) % 21;
    jong = (c - L'가') % 28;
    switch (type) {
case 1:                     | case 2:                     | case 3:
return cho;                 | return jung;                | return jong;
    }
    return -1;
}
```

```
std::string DTC::printToUTF8(vector<wchar_t> input) {
    std::string output;
    wchar_t Uni;
    char Utf8[3];
    output.clear();

    for (int i = 0; i < input.size(); i++) {
        Uni = input[i]
        Utf8[0] = 0xE0;
        Utf8[1] = 0x80;
        Utf8[2] = 0x80;
        if ((Uni < 0xAC00) || (Uni > 0xD7A3)) {
            Utf8[2] = Uni & 0xff;
            output += Utf8[2];
        }
        else {
            Utf8[0] |= (Uni & 0xF000) >> 12;
            Utf8[1] |= (Uni & 0x0FC0) >> 6;
            Utf8[2] |= (Uni & 0x003f);
            output += Utf8;
        }
    }
    return output;
}

void DTC::SayTTS(std::string Say) {
    std::string Base = "./speech.sh "
    std::string msg;
    msg.clear();

    msg += Base;
    msg += Say
    std::wcout << std::endl;
    std::cout << msg << std::endl;

    system(msg.c_str());
}
```