論文 94-19-5-19

인쇄 악보의 인식과 병렬 알고리즘에 관한 연구

正會員 黃 永 吉* 正會員 金 聖 天**

A Study of Printed Score Recognition and its Parallel Algorithm

Yeong-Gil Hwang*, Sung Chun Kim** *Regular Members*

要 約

본 논문에서는 핸디 스캐너를 사용하여 인쇄 악보를 읽어들여서 이를 최종적으로 매쉬 컴퓨터에서 병렬 수행 하도록 한다. 일차적으로 특정 패턴에 따라 분류하고, 지식을 기반으로하여 인식하게 된다. 본 논문에서 제안하는 알고리즘은 전처리 과정을 최소화하고 단순한 연산을 사용한다. 인쇄 악보의 악상 기호 크기는 여러가지가 허용되도록하며 악상 기호의 종류의 다양성 때문에 모든 기호를 인식하는 것은 어려운 일이므로 우선 사용 빈도수가 높은 몇가지의 기호를 인식하도록 한다. 인식된 결과는 미디 표준파일 형식으로 변환하도록 한다. 영상 처리의 고속성이 요구되므로 다중프로세서를 갖는 병렬처리 시스템이 필요하다. 이차원적인 디지탈화된 영상은 SIMD 메쉬 컴퓨터 구조에서 처리되기에 적합하므로 이 구조에 대해서 설명하고 n^2의 프로세서를 갖는 SIMD 메쉬 컴퓨터 구조상에서의 시간 복잡도가 $O(n)$인 병렬 알고리즘을 기술한다.

ABSTRACT

In this thesis, a printed score is read by using handy scanner and the recognition process is executed in parallel, finally, on Mesh-Connected Computer. What is read is classified into certain patterns and is recognized, based on knowledge. The preprocessing steps are minimized and simple operations are used in the algorithm proposed in this thesis. The score symbols on a printed score can be recognized irrespective of their sizes but their diversity makes it difficult to recognize them all, so it is programmed so as to recognize some symbols that is used necessarily and frequently. The recognized result is transformed into the MIDI standard file format. It is required to use a parallel processing system with multiprocessors because the high speed image processing is required.

*西江大學校 電子計算學科
Dept. of Computer Science, Sogang University
論文番號 : 9444
接受日字 : 1994年 2月 15日

A digitized two-dimensional image is appropriate in processing on the SIMD Mesh-Connected Computer(MCC). Therefore, we explain this architecture and present parallel algorithms using SIMD MCC with n^2 processors that achieves time complexity $O(n)$.

I. 서 론

컴퓨터의 성능이 향상됨에 따라 다량의 데이타 및 정보 처리를 고속으로 수행함으로써 방대한 작업들을 신속하게 처리할 수 있게 되었다. 또한 이 컴퓨터에 데이타를 입력시키기 위해서는 여러 형태의 노력이 필요하며 또한 오류를 범할 여지가 항상 있게 되었다.

컴퓨터의 입력장치를 인간에게 편리하도록 할 필요성에 의해 여러가지 입력장치가 개발되어 왔다. 이중 하나인 컴퓨터의 시각장치에 해당하는 CCD 카메라같은 것에 의해 입력이 손쉬워졌으며, 유사 장치로써 문서 또는 영상 정보를 읽는 스캐너(scanner)를 사용할 수 있다. 그러나 이러한 입력 영상은 영상 자체로서는 의도하는 대로의 가치를 갖지 못하며 이것을 유용한 데이타로 변환하여야 한다. 따라서 이러한 영상에 대한 처리기법(image processing techniques)[1-4]과 이를 통한 영상의 이해에 대한 많은 연구가 진행되어 오고 있으며 그 범위도 대단히 넓다. CCD 카메라를 이용하여 입력한 인쇄 악보(printed score) 영상 데이타를 인식하여 특정 시스템에서 연주하기 위한 연구[5,6]가 있었다.

본 논문에서는 사용상의 편리함으로 인해 널리 보급되고 있는 핸디 스캐너를 사용하여 인쇄 악보를 읽어 자동 인식하도록 한다. 일반적인 문서(영문자, 숫자, 한글 및 한자 등)에 대한 연구[7-9]는 여러 나라에서 이루어져 왔고 현재는 대부분 상용화 되고 있는 단계이다.

일반적으로, 스캐너로 읽어들인 영상에는 잡음이 생기며, 읽어들일때의 잘못으로 영상이 기울어질 수 있다. 본 논문에서 인식하고자 하는 인쇄 악보는 충분히 평행 상태로 읽을 수 있다고 가정한다. 이 가정은 조금의 주의로도 충분하며 이로인해 영상을 회전(image rotation)시켜야 하는 커다란 부담을 제거할 수 있다.

기존에 연구된 인쇄 악보의 인식 시스템에서는 악보의 크기가 고정되어 있거나 입력 영상에 들어가는 악상 기호의 수가 적은 경우[5]도 있으며 수행 시간이 긴 경우도 있다[6]. 본 논문에서는 200DPI 정도의 해상도(resolution)를 갖는 핸디 스캐너를 이용하여 악보를 읽어들임으로써 보다 많은 악상 기호를 처리할 수 있으며 여러가지 크기의 악보를 처리할 수 있고, 보표 인식에 있어서 개선된 알고리즘으로 속도를 향상시켰다. 또한 화음을 이루는 음도 인식할 수 있도록 하였다. 악상 기호의 종류의 다양성 때문에 모든 기호를 인식하는 것은 어려운 일이며 우선 사용 빈도수가 높은 몇가지의 기호를 인식하도록 한다. 이 인식된 결과를 현재의 전자악기 시스템인 미디(MIDI : Musical Instrument Digital Interface)시스템에서 연주하기 위해 미디 표준 파일 형식(MIDI standard file format)으로 변환하여 그 결과를 확인하도록 한다.

특히, 악보 영상이 커질 경우 그 처리 시간이 증가하므로 이를 해결하기 위해서는 다중프로세서를 갖는 병렬 처리 시스템[10]으로 구현해야 한다. 따라서 영상 처리를 위한 병렬 컴퓨터 구조(parallel computer architectures)[11] 중 본 논문에서 구현하는 알고리즘에 적합한 SIMD 메쉬 컴퓨터(Mesh-Connected Computer : MCC) 구조에서의 병렬 수행에 대해 연구하며 시간 복잡도면에서 SISD 시스템과 비교, 분석한다.

II. 인쇄 악보의 인식

2.1 관련 연구

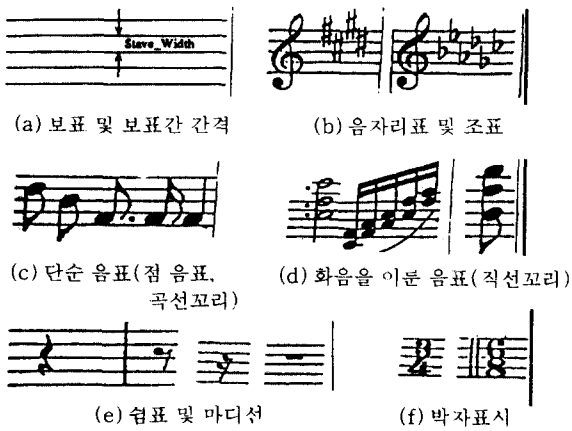
관련 연구 논문[5]에서는 CCD 카메라로 읽어들인 237*192 크기의 영상 데이타를 처리하였는데 한 화면에 5-8개의 기호가 있는 악보중 음표(단순 음표, 잇단 음표, 단순화음 음표, 잇단 화음 음표)만을 인식하였으며 그 인식률은 90% 이상을 나타냈다. 여기에서 잡음을 제거하기 위해 3*3 4-근방 마스크(mask)를 사용하였으며, 보표 추출을위해 히스토그램을 사용하였다. 논문[6]에서는 카메라로 인쇄 악보를 입력하며 영상을 강화(sharpening)하기 위한 전처리 단계를 거쳐서 인식된 보표 영역내에서 히스토그램의 분리성(separability)을 이용하여 개략적인 패턴 분류(coarse classification)를 한다. 이러한 전처리와 개략적인 패턴 분류에 소요되는 시간은 각각 7분 30초

와 24초 정도이다. 그리고 인식 가능한 범위는 보표, 음자리표, 조표, 박자표시, 음표등 필수적인 기호를 인식하였으며 인식률은 95-98%를 나타냈다. 그러나 패턴 분류 과정에서 정규화를 위해 각 기호들의 가로, 세로의 크기를 고려하여 분류함으로써 화음을 이루는 음표들의 정규화가 어려우므로 인식 범위에서 제외되었다. 본 논문에서는 이러한 음표들까지 처리하도록 하였다.

2.2 악보 인식

2.2.1 인식 가능한 기호 및 제약 사항

인식 가능한 악상 기호(그림 1)는 악보에서 필수적인 음표와 쉼표 그리고 조표가 있다. 악보를 입력할 때 스캐너의 해상도에 따라 영상의 정확도(실제 영상과의 차이)가 다른데, 스캐너는 보통 100DPI(Dot-Per-Inch)에서 400DPI 또는 그 이상의 해상도를 가지고 있다. 그리고 입력하고자 하는 악보의 크기도 조금씩 다르다. 본 논문에서의 구현은, 낮은 해상도에서는 인식률이 떨어지고 높은 해상도에서는 입력되는 기호의 수가 적기때문에 인식률과 입력 영상의 크기를 고려하여 200DPI로 악보를 읽어들인다. 그러나 악보 자체의 크기는 가변적일 수 있다. 화음을 이루는 음은 같은 음표의 대를 공유하며 그 음의 길이도 동일하다고 가정한다.



(a) 보표 및 보표간 간격 (b) 음자리표 및 조표

(c) 단순 음표(점 음표, 곡선꼬리) (d) 화음을 이룬 음표(직선꼬리)

(e) 쉼표 및 마디선 (f) 박자표시

그림 1. 인식 가능한 기호의 분류

또한, 핸디 스캐너를 이용하여 읽어들인 영상에는 잡음이 생길 수 있으며 읽어들일 때의 오류로 인해 영상 전체가 기울어진 상태가 될 수 있다. 기울어진 영상을 회전시키기 위해서는 큰 부담이 되므로 조금의 노력으로 충분히 평행 상태의 악보 영상을 획득할 수 있다.

2.2.2 보표 위치 인식

보표의 위치를 인식하기 이전에, 입력된 악보 영상에 포함된 잡음을 제거하여 오인식을 방지하려고 시도한 이전의 연구에 비해 본 논문에서는 이러한 전처리과정을 실행하지 않으면서도 인식률에 영향을 미치지 않도록 노력하였다.

악상 기호 인식에 있어서 위치 정보의 가장 중요한 파라미터가 되는 것이 보표의 위치를 인식하는 것이다. 보표의 위치를 인식하므로서 음표의 위치(음의 높낮이)와 조표 및 쉼표등의 위치를 쉽게 인식할 수 있다. 기존의 연구-논문[5, 6]-에서는 보표의 위치를 인식하기 위한 방법으로 히스토그램 기법을 사용함으로써 영상처리의 시간적 부담을 증가시킨다. 본 논문에서는 이러한 시간 부담을 최소화하는것이 중요한 과제중의 하나이며, 이를 위해 히스토그램 기법을 사용하지 않고 단순한 점 탐색 및 선 탐색을 사용하였다. 먼저 보표중 각 보표의 중심점을 찾기위해 (그림 2)에서와 같이 영상의 최상위 위치에서 아래 방향으로 세 점을 병행하여 scanning down 한다. 세 점((x1, y_i), (x2, y_i), (x3, y_i))은 임의로 영상폭(Image-Width)의 1/4, 1/2, 7/8인 지점을 선정하여 수행한다. 세 점을 병행하여 scanning down하면서 black pixel을 찾는 점 탐색을 실시하고, 세 점에서 black pixel을 찾으면 그 세 점이 직선상에 있는가를 확인한다. 식 (2.1)을 만족하면 세 점((x1, y1), (x2, y2), (x3, y3))은 직선상에 있는 것으로 확인하고 식 (2.2)를 이용하여 기울기(Gradient)를 계산한다.

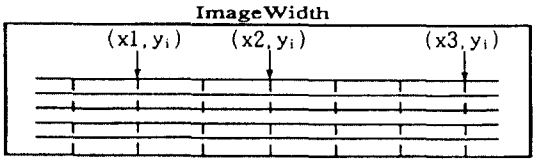


그림 2. 병행 scanning down으로 보표의 위치 인식

$$x1 = ImageWidth * \frac{1}{4}$$

$$x2 = ImageWidth * \frac{1}{2}$$

$$x3 = ImageWidth * \frac{7}{8}$$

$$\left| \frac{y3-y2}{x3-x2} - \frac{y2-y1}{x2-x1} \right| < Delta \quad (2.1)$$

Delta : 기울기의 허용 오차

$$Gradient = \frac{y3-y1}{x3-x1} \quad (2.2)$$

악보 영상의 입력시 제한된 입력으로 기울기는 충분히 0°에 가까와야 한다. 기울기가 계산되면 두 점 (x1, y1)과 (x3, y3)가 연결된 선분인가 확인한다. (그림 3)에서 처럼 기울기에 따라 2*3 윈도우 또는 3*3 윈도우를 우측으로 이동시키면서 선 탐색을 실시하여 윈도우내의 black pixel의 수를 계산하여 두 점 (x1, y1)과 (x3, y3) 사이의 black pixel의 수(Continuous)가 전체 가능한 black pixel (x3-x1+1)의 어느 정도(약 70%) 이상(식 (2.3))이 되면 연결된 선분으로서 보표중의 하나로 간주한다. 같은 방법으로 나머지 네 개의 보표의 한 성분을 찾고 식 (2.4), (2.5), (2.6), (2.7)을 이용하여 각각의 보표의 간격을 계산하여, 인접한 보표의 성분 사이의 간격이 평균 보표간 간격과의 차이가 허용 오차내에 있는가를 조사한다.

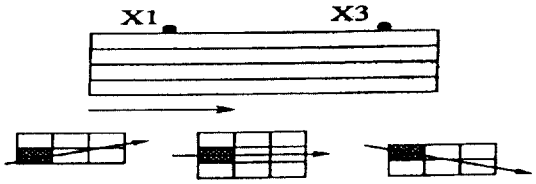


그림 3. 보표의 연결성 확인을 위한 방법

$$\left| (M2\_y - MT\_y) - \frac{MB\_y - MT\_y}{4} \right| < Delta \quad (2.4)$$

$$\left| (M3\_y - M2\_y) - \frac{MB\_y - MT\_y}{4} \right| < Delta \quad (2.5)$$

$$\left| (M4\_y - M3\_y) - \frac{MB\_y - MT\_y}{4} \right| < Delta \quad (2.6)$$

$$\left| (MB\_y - M4\_y) - \frac{MB\_y - MT\_y}{4} \right| < Delta \quad (2.7)$$

Delta : 간격의 허용 오차

$$Delta = \frac{\frac{MB\_y - MT\_y}{4}}{5}$$

(MT_x, MT_y) : 첫번째 보표 성분의 중심점
(M2_x, M2_y) : 두번째 보표 성분의 중심점
(M3_x, M3_y) : 세번째 보표 성분의 중심점
(M4_x, M4_y) : 네번째 보표 성분의 중심점
(MB_x, MB_y) : 마지막 보표 성분의 중심점

$$Stave\ Width = \frac{MB\_y - MT\_y}{4} \quad (2.8)$$

그리고 각 보표의 간격의 평균(식 (2.8))을 scale factor(Stave Width)로 사용하여 여러가지 크기의 악보 영상을 모두 처리할 수 있도록 한다.

보표가 모두 인식되었으면 그 보표의 최좌우측 끝점을 결정하기 위해 각 보표의 중심점(예 : (MT_x, MT_y))에서 연결된 선분을 이용하여 픽셀의 연결성을 조사하는데, 3*3 윈도우내에 black pixel이 있으면 좌우를 잇는 연결성을 가진 것으로 간주하여 그 윈도우를 각각 좌우측으로 이동시키는 선 탐색을 실시하여 좌우 끝점 ((LT_x, LT_y), (RT_x, RT_y))을 결정한다(그림 4). 이상의 정보를 이용하여 다음 단계에서는 여러가지 악상 기호를 분류, 인식하게 된다.

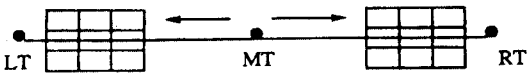


그림 4. 보표의 좌우 끝점을 결정하기 위한 방법

2.2.3 보표의 삭제

악보 인식에 있어서 가장 중요한 요소가 보표이며 보표 인식이 끝나면 다른 악상 기호에 대한 인식을 해야한다. 그런데 보표를 먼저 인식한 단계에서 보표는 다음 인식 단계에서는 필요없게 되며 또한 방해요소가 될 수도 있으므로 보표를 삭제하는 것이 좋다. 보표는 음표의 꼬리와 머리부분, 쉼표등과는 다르게 수직 방향의 black run length(BRL)가 작으므로 인

식된 보표의 위치를 중심으로 상하의 수평선에 대해서 그 run length가 작은 픽셀들을 삭제한다. 논문[6]에서는 수직 방향으로 연결성을 조사하여 현재의 바로 윗 픽셀이 black pixel이거나 아래 두 픽셀이 black pixel, 또는 위의 네 픽셀과 아래 네 픽셀중 다섯개 이상이 black pixel인 이러한 세 가지 경우가 아니면 보표의 일부인 것으로 판단하고 이 픽셀을 삭제하게 된다. 이러한 보표 삭제 알고리즘을 조금 수정하여 본 논문에 적용한다. 보표를 이루는 선이 하나의 픽셀이 아닌 여러개의 픽셀로 이루어질 수 있기 때문에(스캐너의 해상도에 따라) 그 상하의 픽셀에 대해서도 보표 삭제 알고리즘을 적용한다.

Stave_Width=8 일때 (그림 5)의 (a), (b), (c) 세 가지 – 일반적으로 scale factor만큼의 폭을 갖는다. 즉, y′=[Stave_Width/2] 이고 (x, y−y′), (x, y−y′+1), ..., (x, y−y′+(y′−1)) 그리고 (x, y+1), (x, y+2), ..., (x, y+y′) 픽셀을 검사한다 – 는 그 픽셀이 삭제되지 않고 이외의 경우는 삭제된다. 보표가 기울어지거나 부분적으로 두께가 두꺼운 경우 즉, 잡음이 있는 경우는 완전히 삭제되지 않고 남아 있게된다. 수행된 결과는 (그림 6)과 같다.

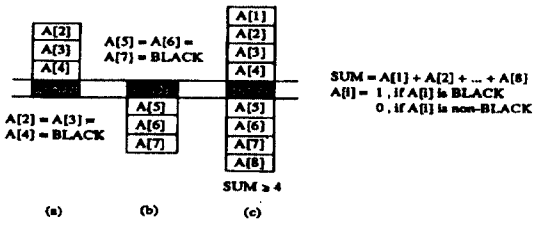
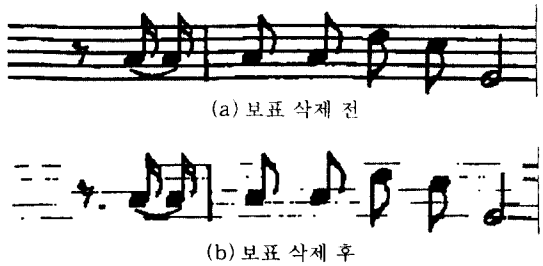


그림 5. 삭제되지 않을 픽셀의 조건



(a) 보표 삭제 전

(b) 보표 삭제 후

그림 6. 보표 삭제 수행의 결과

2.2.4 히스토그램

악상 기호의 인식 과정에 앞서 그 인식에 사용되는 주요 기법인 히스토그램(histogram)을 살펴본다. 히스토그램이란 영상 또는 영상의 일부에 대한 픽셀-강도값의 분포 그래프이며 영상 처리 작업에 있어 질적 양적으로 귀중한 도구이다.

앞으로 악상 기호의 패턴 인식에는 수평 히스토그램과 수직 히스토그램을 조합하여 그 특징에 따라 마디선(bar line), 음표 대(note stem), 음표 머리(note head), 음표 꼬리(note flag, note beam), 쉼표(rest), 조표(sharp, flat) 및 기타 특수 기호(special symbols)를 인식하도록 한다. 수평 히스토그램(horizontal histogram)은 얻어진 보표의 위치 정보를 이용하여 좌측에서 우측으로 이동(left-to-right scanning)하면서 현재의 X 좌표상의 모든 Y 좌표 ((x_i, y_0), (x_i, y_1), (x_i, y_2), ..., (x_i, y_{n-1}))의 black pixel의 수를 합한다(식 (2.9)).

$$Hist[x_i].density = \sum_{j=lower}^{upper} (color(x_i, y_j)) \qquad (2.9)$$

$$color(x_i, y_j) = \begin{cases} 1 : black\ pixel \\ 0 : non\text{-}black\ pixel \end{cases}$$

수직 히스토그램(vertical histogram)은 일정 영역내에서 상에서 하로 이동(upper-to-lower scanning)하면서 현재의 Y 좌표에서 모든 X 좌표((x_0, y_j), (x_1, y_j), (x_2, y_j), ..., (x_{n-1}, y_j))의 black pixel의 수를 합한다(식 (2.10)).

$$Hist[y_j].density = \sum_{i=left}^{right} (color(x_i, y_j)) \qquad (2.10)$$

$$color(x_i, y_j) = \begin{cases} 1 : black\ pixel \\ 0 : non\text{-}black\ pixel \end{cases}$$

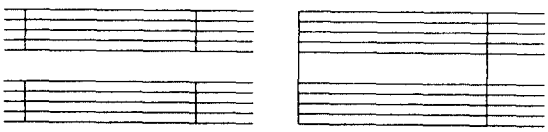
2.2.5 악상 기호 추출

2.2.4에서 설명한 수평 히스토그램과 수직 히스토그램을 이용하여 보표상에 나타나는 여러가지 악상 기호의 특징을 추출할 수 있으며, 추출된 특징을 일반적인 악보에서 나타나는 특징과 비교하여 마디선 부분, 음표 부분, 쉼표 부분을 인식할 수 있고 여기에 위치 정보를 첨가하여 다른 몇가지의 악상 기호 인식을 할 수 있다.

2.2.5.1 마디선 인식

먼저 악상 기호 인식에 있어서 기본이 되는 마디선(bar line)을 수평 히스토그램을 이용하여 인식함으로써 한 마디내의 박자수를 알 수 있다. 마디선의 구분은 인식의 오류로 인한 한 마디내의 박자수의 오류(박자수가 모자라거나 초과하는 경우)를 인식후 단계의 수정 과정(music syntax error correction level)에서 사용될 수 있다. 악보상에 나타나는 마디선의 특징은, 전체 보표의 높이(식 (2.11))에 걸쳐서 전 구간에 black pixel로 나타나고 그 상하단에는(그림 7 (b))처럼 연속적으로 연결될 수도 있다.

$$Height = MB\_y - MT\_y \tag{2.11}$$



(a) 오선 높이의 마디선 (b) 오선 상하에 걸친 마디선

그림 7. 마디선 부분의 패턴

그리고 마디선은 읽어들일때의 스캐너의 해상도에 따라 두께(thickness)가 다를 수 있으며, 마디선을 중심으로 좌우측의 픽셀 강도(식 (2.12), 식 (2.13))는 거의 비슷한 정도를 나타낸다.

$$\text{좌측 픽셀 강도} = \sum_{j=lower}^{upper} (color(x_{i-d}, y_j)) \tag{2.12}$$

$$\text{우측 픽셀 강도} = \sum_{j=lower}^{upper} (color(x_{i+d}, y_j)) \tag{2.13}$$

$$d : \lceil Stave\_Width/4 \rceil$$

2.2.5.2 음표 인식

(1) 음표의 대 인식

계산된 수평 히스토그램에서 마디선 부분과 음표의 대 부분은 그 강도(black pixel의 빈도수)가 비슷하지만 그 주위의 히스토그램 패턴은 다르게 나타난다. 즉 음표의 대를 중심으로 좌우측 영역에는 음표의 머리로 픽셀 강도가 마디선의 그것보다 강하게 나타난다. 마디선 부분과 음표 부분에 대한 수평 히스토그램을 보면(그림 8)과 같다.

그림 8. 수평 히스토그램

위에서 구한 음표의 대 부분의 X 좌표를 알고, 음표의 대의 수직 연결성을 3*3 윈도우를 이용하여 선탐색을 함으로써 Y 좌표의 상한과 하한을 구한다. 여기에 앞서 구한 scale factor Stave_Width를 이용하여 음표 부분의 영역을 설정할 수 있다. 설정된 영역은 (그림 9)와 같다. 이것은 X, Y좌표의 상한과 하한을 최소화 시킴으로써 논문[6]에서 설정되는 심볼 윈도우의 필요없는 영역을 제거할 수 있다.

그림 9. 음표 부분 영역 설정

(2) 음표의 머리 인식

음의 높낮이를 인식할 때 논문[6]에서는 단순한 점탐색을 사용하므로써 잡음에 의해 오류가 발생할 수 있는데 이를 없애기 위해 음표머리를 가질 수 있는 가능성이 있는 위치에서 패턴 매칭을 실시한다.

(그림 9)에서 설정된 영역내에서 수직 히스토그램(그림 10)을 생성하며 이것을 이용하여 음표의 머리 부분을 대략적으로 추출할 수 있고, 명확한 인식을 위해 보표의 위치정보와 음표 모리의 위치 정보를 이용하여 패턴 매칭을 실시한다. (그림 11)과 같이 음표 머리의 한 영역에 대해 패턴 매칭을 실시함으로써 머리의 종류를 인식할 수 있다.

그림 10. 수직 히스토그램

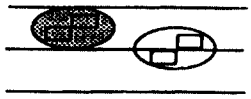


그림 11. 패턴 매칭을 이용한 음표 머리의 인식

기존의 연구는 단순음을 인식하는데 그쳤으나 본 논문에서는 한 음표의 대에 음표의 머리가 두 개 이상 있을 수 있는 경우(화음을 이루는)에도 인식가능하도록 하였으며 단, 음표머리의 위치는 음표대의 좌측 또는 우측에 같이 존재하는 경우로 제한한다. 이의 인식은 음표대의 아래에서 위로(음표의 머리가 대의 좌측에 있을 경우) 또는 위에서 아래로(음표의 머리가 대의 우측에 있을 경우) 탐색해 가면서 머리 부분에 대해서 같은 방법으로 패턴 매칭을 실시함으로써 가능하다.

(3) 음표의 꼬리 인식

음표의 꼬리 종류는 두 가지로 나눌 수 있는데 곡선 형태(flag)와 직선 형태(beam)가 그것이다. 먼저 곡선 형태의 음표 꼬리를 인식하기 위해서 방향성 에지(edge)를 추출하는 Sobel edge detection 알고리즘[12]을 사용한다. Sobel edge detection 알고리즘은 (그림 12)에서 처럼 음표 영역의 우측 상단 또는 우측 하단의 일정 영역을 설정하여 우측에서 좌측으로 탐색하다가 black pixel을 찾으면 그 픽셀을 중심으로 3*3 윈도우를 설정하고 에지의 방향을 계산한다. 음표의 꼬리가 인식되면 꼬리의 갯수를 위한 black run length를 계산한다.

$$Ey = (A2 + 2*A3 + A4) - (A0 + 2*A7 + A6)$$
$$Ex = (A6 + 2*A5 + A4) - (A0 + 2*A1 + A2)$$

A0	A1	A2
A7	(X,Y)	A3
A6	A5	A4

Edge 크기 : E = Ex + Ey
Edge 방향 : θ = arctan(Ey/Ex)

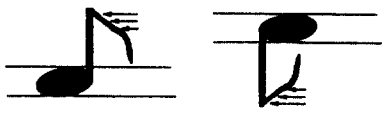


그림 12. Sobel edge detection 알고리즘을 이용한 곡선 꼬리 인식

또다른 꼬리 형태인 직선 형태의 꼬리는 black run length를 이용한다. (그림 13)에서 처럼 위에서 아래로(또는 아래에서 위로) 탐색하면서 연속된 black pixel의 수(run length)를 계산하며 이는 보표의 두께보다 굵기때문에(보통 Stave_Width의 1/3 이상) 꼬리로써 인식이 가능하다.

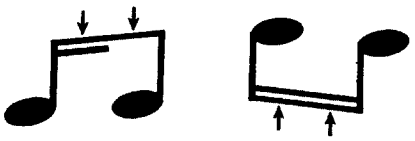


그림 13. black run length를 이용한 직선 꼬리 인식

위의 두가지 형태의 음표꼬리를 인식할 때 두 형태의 분류는 꼬리의 위치 및 기울기, 두께에 의해 이루어진다.

(4) 점 음표 인식

음표의 일부로써 점(dot)은 음표의 종류와 위치에 따라 그 위치가 다른데 다음과 같이 네 가지로 나눌 수 있다. 음표에 꼬리 성분이 있는가 없는가에 따라 X축상에서의 이동이 있고, 음표의 머리 부분이 보표상에 있는가 아니면 보표 사이에 있는가에 따라 각각 그 보표의 바로 우측위 또는 우측에 위치하게 된다. 형태별로 (그림 14)에 나타낸다. 점 음표는 앞서 인식한 음표의 1/2을 더함으로써 그 음의 길이가 된다. 그러나 직선 형태의 꼬리는 꼬리가 없는 음, 즉 4분 음표와 같이 X축 이동이 없다.

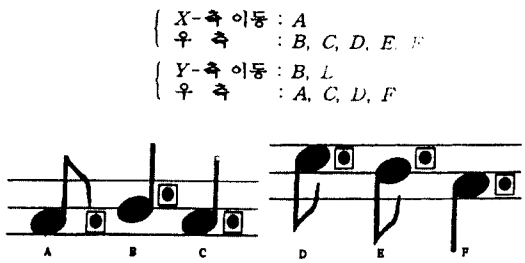


그림 14. 위치 정보를 이용한 점의 인식

2.2.5.3 쉼표 인식

쉼표(rest symbol)는 보표상에서 일정한 위치에 나타나며, 음표의 대 부분과 같이 강한 픽셀 강도를 나타내지는 않는다. 이와같이 수평 히스토그램과 보표상의 위치 정보를 이용하여 일정 영역이 설정되며 그 영역내의 패턴의 특징에 의해 쉼표의 종류를 구분할 수 있다.

2.2.5.4 조표 인식

조표(key signature)는 대개 보표(stave)의 좌측에 치우쳐서 보표상의 정해진 위치에 나타나며, 같은 종류의 조표가 한 개 이상 연속적으로 고유의 자리에 위치한다. 이것도 쉼표와 마찬가지로 위치 정보와 패턴의 특징 및 출현 횟수에 의해 인식이 가능하다.

2.2.5.5 음자리표 인식

음자리표(clef)는 보표의 맨 좌측에 나타나며 높은 음자리표(G-clef)의 경우 그 폭이 다른 기호들 보다 훨씬 넓으며 높이도 또한 높게 나타난다. 낮은 음자리표(F-clef)는 대개 높은 음자리표와 쌍을 이루어 나타나며 같은 X 좌표를 갖는다.

2.2.5.6 박자 표시 인식

음자리표, 조표 다음에 나타나는 것이 박자 표시(time signature) 기호(숫자)이다. 가운데 보표(오선의 세번째)를 중심으로 분자와 분모로 나누어지고 2, 3, 4, 6과 4, 8이 각각 있을 수 있으며, 숫자가 지닌 특성(hole을 갖는가의 여부 및 그 갯수)에 따라 분류된다.

2.2.5.7 기타 특수 기호

위의 인식 과정으로 복잡하지 않은 일반적인 악보의 인식이 가능하며 좀 더 복잡한 특수 기호의 인식을 위해서는 더 많은 분류 및 인식 알고리즘이 첨가되어야 하고, 이로 인해 인식에 필요한 시간은 악상기호의 빈도수에 비례하여 증가할 것이며 차후에 계속 연구되어야 할 것이다.

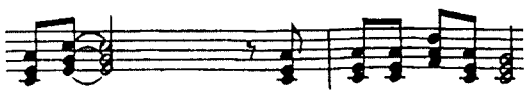
2.2.6 인식 결과

본 논문에서는 IBM PC 호환 386-DX 상에서 수행하였으며 그 결과는 평균 수행시간(보표삭제 수행시간 포함)이 25초이고 평균 인식률은 94%이다. 즉 수행 시간면에서 아주 뛰어남을 확인할 수 있다. 데이타 압축률은 영상 데이타(PCX 파일 형식)에 대한 인식 결과의 중간 형태 아스키(ASCII) 데이타의 비율(byte수)로서 약 6%로 데이타량을 줄일 수 있다.

Ⅲ. 음악 데이타 생성

3.1 중간 형태 파일 생성

인식한 악상 기호는 악보를 연주하기 위해 필요로 하는 음악 데이타 즉, 음의 길이 및 높낮이, 화음, 쉼표의 길이, 조의 종류를 문자 데이타로 변환, 저장되어야 한다(그림 15).



```
(57) 60 12  (57) 64 12  (57) 69 12
(102) 64 12  (102) 67 12  (102) 72 12
(145) 64 48  (145) 67 48  (145) 71 48
(310) 250 12
(357) 60 12  (357) 64 12  (357) 69 12
(400) -1 -1
(440) 60 12  (440) 64 12  (440) 69 12
(484) 60 12  (484) 64 12  (484) 69 12
(530) 65 12  (530) 69 12  (530) 74 12
(571) 60 12  (571) 64 12  (571) 69 12
(614) 60 48  (614) 64 48  (614) 67 48
(14) -1 -1
```

- 순서대로 (X좌표), 음의 높이, 음의 길이를 표시
- 같은 좌표의 음은 화음, 음의 높이가 250인 경우는 쉼표, -1인 경우는 마디 구분을 표시

그림 15. 중간 형태 파일의 생성 결과

3.2 오류 검증 및 수정

생성된 음악 데이타 파일을 음악 문법(한 마디내의 박자수가 정당한가)에 맞는가 분석하고 맞지 않으면 오류가 생길 수 있는 경우를 생각하여 경험적 방법에 의해 오류를 수정한다. 대개 오류가 발생할 수 있는 경우는 점 음표의 미인식, 온음표의 미인식, 그리고 쉼표의 미인식등이다. 또한 기보법[13]에 의해 음표간의 거리는 음표 자체의 음길이에 비례하므로 이를 이용한다.

3.3 미디 구현

미디[14, 15]는 신세사이저(synthesizers)와 다른 전자 음악 장비간의 상호 접속을 목적으로 만들어진 통신 규약(communications protocol)이다. 이러한 미디 규격을 사용하는 기기에서 음악 데이타를 연주하기 위해 앞서 생성한 중간 형태의 파일을 미디 표준 파일 형식[16]으로 변환시켜야 한다.

3.3.1 미디 표준 파일 형식

미디 데이타 스트림은 일련의 바이트로 구성되며 크게 상태 바이트(status bytes)와 데이타 바이트(data bytes)로 이루어진다. 각 메세지는 상태 바이트와 그 뒤를 따르는 데이타 바이트(없거나 하나 이상일 수 있다)로 구성된다. 미디 파일은 헤더 청크(header chunk)와 트랙 청크(track chunk(s))로 나누어지는데 헤더 청크는 헤더를 표시하는 문자('MThd' : 4D 54 68 64)와 헤더 부분의 길이, 파일의 전체 구조, 파일의 트랙 청크 번호, 그리고 델타 시간(delta time)의 목적을 명시하는 부분으로 구성되고, 트랙 청크는 실질적인 음악 데이타를 표시하는 부분으로써 하나 이상의 트랙 이벤트(track event(s))로 이루어져 있다.

3.3.2 중간 형태 파일의 미디 파일 변환

중간 형태 파일은 순수한 악보에 대한 데이타밖에 없으므로 우선 헤더 부분을 표준 파일 형식에 맞도록 기록하고, 그 다음에 트랙 부분을 나타내는 문자('MTrk' : 4D 54 72 6B)와 기타 필요로 하는 몇가지 정보를 함께 기록(그림 16) 하는데 예를들어 악기번호같은 것은 임의로 선정해 준다. 그리고 음의 세기는 일정한 크기(40h)로 고정시키고 화음을 이루는 음들의 길이는 동일하다고 가정하여 동시에 음이 발생하도록 하고 또한 음의 소멸도 동시에 일어나도록 한다.

```
4D 54 68 64 00 00 00 06 00 01 00 02 00 78 4D 54
72 6B 00 00 00 19 00 FF 59 02 00 00 00 FF 51 03
0A 67 5A 00 FF 58 04 04 02 18 08 00 FF 2F 00
4D 54 72 6B 00 00 01 CB 00 C1
                ( 중략 )
00 00 91 30 40 00 91 34 40 00 91 39 40 3C 91 30 00
00 91 34 00 00 00 91 39 00 00 91 34 40 00 91 37 40
00 91 3C 40 3C 91 34 00 00 91 37 00 00 91 3C 00
00 91 34 40 00 91 37 40 00 91 3B 40 81 70 91 34
00 00 91 37 00 00 91 3B 00 00 91 00 00 3C 91 30
40 00 91 34 40 00 91 39 40 3C 91 30 00 00 00 91 34
00 00 91 39 00 00 91 30 40 00 91 34 40 00 91 39
40 3C 91 30 00 00 91 34 00 00 91 39 00 00 91 30
40 00 91 34 40 00 91 39 40 3C 91 30 00 00 00 91 34
00 00 91 39 00 00 91 35 40 00 91 39 40 00 91 3E
40 3C 91 35 00 00 91 39 00 00 91 3E 00 00 91 30
40 00 91 34 40 00 91 39 40 3C 91 30 00 00 91 34
00 00 91 39 00 00 91 30 40 00 91 34 40 00 91 37
40 81 70 91 30 00 00 91 34 00 00 91 37
                ( 중략 )
00 00 FF 2F 00 00 91 3C 40 5A 91 30 00 00 91
3C 00 00 91 F4 40 05 91 F4 00 00 FF 2F 00
```

그림 16. 미디 파일 생성 결과

Ⅳ. 메쉬 컴퓨터 구조 및 병렬 알고리즘

4.1 병렬 수행의 필요성

영상 처리와 컴퓨터비젼에서는 일반적으로 그 데이타의 방대함과 처리 함수의 복잡함에 의해 처리 시간이 길어지므로 실시간 처리가 힘들어지고 지루함을 느끼게 한다. 이를 해결하기 위해서는 빠른 처리 속도를 갖는 프로세서가 필요하고 다중 프로세서를 갖는 병렬 컴퓨터 구조[11, 17-19]를 사용한다. 많은 저수준의 영상 처리 연산(low-level image-processing operations)은 공간적 특성에 의해 대개 높은 병렬성을 가지므로 공간지향적인 컴퓨터 구조(spatially oriented computer architectures)에서 대량 병렬(massively parallel) 구현이 가능하다.

4.2 SIMD 메쉬 컴퓨터 구조

본 논문에서 시스템 구성이 비교적 간단하며 널리 사용되고 있는 SIMD 메쉬 컴퓨터(Mesh-Connected Computer : MCC) 상에서의 병렬 알고리즘을 기술하고자 하므로 SIMD MCC에 대해 논하고자 한다. 실제로 구현된 병렬 처리 시스템으로는 Connection Machine(CM)을 사용하여 인쇄 악보를 인식한 것이 있는데 CM은 16K 프로세서와 8K 메모리를 갖는 SIMD 컴퓨터이다[10].

고체 전자소자(solid state electronic devices) 기술의 발전에 의해 컴퓨터 하드웨어의 비용 측면보다 문제 해결을 위한 시간 단축이 더 큰 관심 사항이 되기때문에 비용보다 수행 시간(execution time)을 감소시키는데 중점을 두어 순차 알고리즘을 병렬 알고리즘으로 변환하고자 한다.

이차원적인 디지탈화된 영상은 SIMD MCC 구조상에서 처리되기에 적합하며, 이 경우 각 프로세서는 영상의 각 픽셀에 대해 지역적(동일한)연산을 동시에 수행할 책임을 갖게된다. 여기에서의 가정은 $n*n$ 픽셀 영상은 pixel(i,j)가 PE(i,j)의 RAM에 기억된다.

4.3 이차원 메쉬 컴퓨터상에서의 병렬 알고리즘

순차적으로 수행되는 알고리즘을 SIMD MCC 상에서 병렬로 수행하기 위해서, 연산의 제약성을 고려하여 먼저 병렬 처리 가능한 알고리즘을 분류하면 보표의 위치 인식(각 보표의 중간점 및 좌우 끝점), 보표의 삭제, 보표의 히스토그램 생성, 수직선 성분의

추출, 음표의 대를 공유한 음표 머리 부분의 인식, 음표의 (곡선 형태의)꼬리 인식, 점 음표의 인식이다. 이 중 시간 복잡도를 낮출 수 있는 알고리즘을 기술한다.

4.3.1 보표의 삭제

구역내의 모든 PE에서 동시에 보표 삭제 연산을 주변 정보를 이용하여 상수 시간내에 처리할 수 있다.

(단계1)-(단계4)에서 해당 PE의 위에 있는 PE들(PE(x-1, y), PE(x-2, y), PE(x-3, y), PE(x-4, y))이 병렬로 각각 아래로 있는 PE로 픽셀값을 전송하여 해당 PE의 A4, A3, A2, A1 변수에 순서대로 저장한다. 같은 방법으로 (단계5)-(단계8)을 수행하여 A5, A6, A7, A8에 저장하고 (단계9)에서 해당 PE의 픽셀값을 수정(삭제)한다.

단계1 : PE(x-1, y), PE(x-2, y), PE(x-3, y), PE(x-4, y)의 픽셀값을 각각 PE(x, y), PE(x-1, y), PE(x-2, y), PE(x-3, y)로 전송
단계2 : PE(x-1, y), PE(x-2, y), PE(x-3, y)의 픽셀값을 각각 PE(x, y), PE(x-1, y), PE(x-2, y)로 전송
단계3 : PE(x-1, y), PE(x-2, y)의 픽셀값을 각각 PE(x, y), PE(x-1, y)로 전송
단계4 : PE(x-1, y)의 픽셀값을 각각 PE(x, y)로 전송
단계5 : PE(x+1, y), PE(x+2, y), PE(x+3, y), PE(x+4, y)의 픽셀값을 각각 PE(x, y), PE(x+1, y), PE(x+2, y)로 전송
단계7 : PE(x+1, y), PE(x+2, y)의 픽셀값을 각각 PE(x, y), PE(x+1, y)로 전송
단계8 : PE(x+1, y)의 픽셀값을 각각 PE(x, y)로 전송
단계9 : PE(x, y)의 픽셀값 삭제 조건에 따라 픽셀값 조정

순차 알고리즘에서는 각 행에 대해 각 열의 수 만큼 수행함으로 $O(m*n)$의 시간 복잡도를 가지며, 병렬 알고리즘은 모든 PE가 (단계1)에서 (단계8)까지의 연산을 동시에 수행함으로써 $O(1)$의 시간 복잡도를 갖는다(m: 보표 영역의 폭, n : 보표 영역의 높이).

4.3.2 보표 히스토그램 생성

보표의 좌우 및 상하 끝점(LT_x, LT_y), (LB_x, LB y), (RT_x, RT_y), (RB_x, RB_y))에서 scale factor Stave_Width를 이용하여 히스토그램을 생성할 구역을 설정한다. 같은 X 좌표상의 모든 Y 좌표값에 대해 병렬 수행하여 결과값을 마지막 행에 각 열의 누적 결과값을 저장하도록 한다. 즉 (단계1)과 (단계2)가 같은 행에서 보표 영역의 높이 만큼 병렬로 수행되므로 순차 알고리즘의 $O(m*n)$보다 나은 $O(n)$의 시간 복잡도를 갖는다(m: 보표 영역의 폭, n : 보표 영역의 높이, $m > n$).

단계1 : PE(x, y_i) 픽셀값을 PE(x+1, y_i)로 전송
단계2 : 전송받은 값을 현재의 픽셀값과 더한다

4.3.3 수직선 성분 추출

생성된 히스토그램을 이용하여 마디선, 음표의 대와 같은 수직선 성분을 추출할 수 있으며 이는 히스토그램 결과값을 가지고 있는 마지막 행의 모든 PE에서 병렬로 수행할 수 있다.

단계1 : 마지막 행의 모든 PE에서 병렬로 마디선인가 음표의 대인가를 구분
단계2 : 마디선의 확인을 위해 수행
단계3 : 음표의 대를 가진 PE에서 자세한 음표 분류를 위한 연산을 수행

(단계1)에서 마지막 행의 모든 열에 해당하는 PE는 상수 시간에 처리할 수 있고, (단계2)에서도 마디선 확인을 위해 상수 시간이 필요하다. (단계3)에서는 음표의 대를 갖는 PE가 동시에 수행될 수 있어서 역시 상수 시간에 처리할 수 있다. 그러므로 $O(m)$의 순차 알고리즘을 $O(1)$로 개선한다(m: 보표 영역의 폭).

4.3.4 음표 꼬리 인식

설정된 영역(상우 또는 하우 영역)에서 곡선 형태의 꼬리(flag)를 인식하기 위해서 좌우로의 병렬 탐색과 활성화된 PE에서의 Sobel 연산을 각 PE에서 병렬로 수행한다.

단계1 : 해당 영역의 우측 끝열(right-most column)에 위치한 모든 PE들은 병렬로 차례대로 좌측으로 탐색해 나가다가 검은 픽셀을 찾으면 (단계2)를 위해 변수를 set

단계2 : (단계1)에서 변수가 set된 PE에 대해 Sobel 연산 수행
단계3 : 해당 영역의 좌측 끝열에 위치한 PE들은 병렬로 차례대로 우측으로 Sobel 연산 결과값은 전송
단계4 : 우측 끝열에서 마지막 행에 있는 PE로 결과값을 전송, 누적시킨다
단계5 : 평균 기울기 계산

(단계1)에서 $O(m)$, (단계2)에서는 $O(1)$, (단계3)은 순차 알고리즘에서는 불필요하지만 SIMD에서 연산의 동일성을 위해 Sobel 연산 결과값을 우측 마지막 열의 PE로 전송하기 위해 $O(m)$이 필요하다. (단계4), (단계5)도 역시 평균 기울기를 계산하기 위해 $O(n)$과 $O(1)$이 필요하여 전체적으로 $O(n^2)$가 $O(n)$으로 개선된다(m : 영역의 폭, n : 영역의 높이, $m \simeq n$).

4.3.5 점 음표 인식

음표의 종류(음표 머리의 위치가 선 상에 있는가, 선 사이에 있는가, 음표 꼬리가 있는가, 있으면 음표대의 윗쪽 또는 아래쪽에 연결되어 있는가)에 따라 설정된 영역을 최좌측열(left-most column)의 PE에서 병렬로 우측으로 탐색한다. 이때 검지않은 픽셀에서 검은 픽셀로(white-to-black), 검은 픽셀에서 검지않은 픽셀로(black-to-white) 변하는 PE를 계산(count)하여 wtob=1, btow=1인 행을 활성화한 다음 마지막 열의 count를 합산(sum)하여 임계값(margin) 이상이면 구역내에 점이 존재하는 것으로 간주한다.

단계1 : 모든 행에서 병렬로, 좌측에서 우측으로 탐색하면서 white-to-black, black-to-white를 계산하면서 그 값을 전송
단계2 : 최우측열의 PE에서 white-to-black과 black-to-white를 검사
단계3 : 최우측열의 PE 결과값을 마지막 행의 PE로 전송, 누적

(단계1)에서 각 행에 대해서 동시에 우측으로 탐색함으로 $O(m)$, (단계2), (단계3)은 각각 $O(1)$, $O(n)$의 시간 복잡도를 갖는다. 그러므로 $O(n^2)$의 시간 복잡도를 갖는 순차 알고리즘이 $O(n)$으로 된다.($m \simeq n$)

4.4 시간 복잡도 분석

4.3절에서 기술한 알고리즘들은 인식 알고리즘의 주요 부분이면서 시간적으로도 전체중 많은 비중을 차지하는데 전체 인식 알고리즘 측면에서 볼 때 순차 알고리즘의 시간 복잡도는 보표의 삭제, 보표의 히스토그램 생성, 수직선 성분 추출, 음표의 꼬리 인식, 점 음표 인식 알고리즘에 의해 결정되며 $O(n^2)$의 시간 복잡도를 가진다. 이것을 병렬화함으로써 $O(n)$이 된다.

V. 결 론

본 논문에서의 인쇄 악보 인식을 기존의 연구에서 수행한 전처리 과정을 생략하였으며, 보표를 찾는 알고리즘이 기존의 수평 히스토그램을 사용한 것보다 보표의 특성을 충분히 활용한 점 탐색 및 선 탐색 알고리즘으로 인식 시간을 단축하였고 인식률을 향상시키기 위해 보표를 삭제하였다. 이로 인한 시간 부담이 많이 증가하였으나 2.2.6절의 인식 결과에서 처럼 여전히 인식 시간은 짧다. 음표 부분의 영역설정에 있어서 불필요한 영역을 없애고 처리영역을 최소화 하였고, 보표의 위치 정보를 이용한 음표의 위치 인식에 있어서도 잡음에 의한 오류를 없애기 위해 패턴 매칭을 사용하였다. 또한 기존의 연구에서 처리하지 못한 화음구성 음표도 처리하도록 하였다.

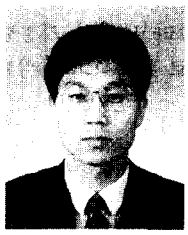
여러가지 악보를 실험한 결과를 평균 인식률은 94%, 인식 시간은 평균 25초로써 기존의 연구와 비교해서 속도면에서 빠른것을 알 수 있다. 인식된 결과를 미디 표준 파일형식으로 변환하여 미디 시스템에서 연주해 보았으며, 악보 인식에 적합한 메쉬 컴퓨터 구조를 설명하고 4.3절에서 이 구조상에서 병렬 수행 가능한 알고리즘들의 병렬 알고리즘을 기술하고 시간 복잡도를 낮추었다.

병렬 알고리즘을 구현하여 실제 수행 시간을 비교하는 것은 앞으로의 과제이다. 그리고 인식 가능한 기호의 범위를 특수 기호 및 문자로 확대하고 인식률을 높여야 할 것이다.

참 고 문 헌

1. Craig A. Lindley, Practical Image Processing in C, John Wiley and Sons, Inc., 1991.
2. Sing-Tze Bow, Pattern Recognition and Image Preprocessing, Marcel Dekker, Inc., 1992.

3. Zahid Hussain, Digital Image Processing : practical applications of parallel processing techniques, Ellis Horwood Limited, 1991.
4. Jae S. Lim, 2-Dimensional Signal and Image Processing, Prentice-Hall, 1990.
5. 이명우, 최종수, "컴퓨터비전 시스템에 의한 인쇄 악보의 인식과 연주," 전자공학회지, 22권 5호, pp.10-16, 1985.
6. 김완주, "인쇄 악보 인식시스템의 구현에 관한 연구," 한국과학기술원 석사학위 논문, 1986.
7. Yao Yong, "Handprinted Chinese Character Recognition via Neural Networks," Pattern Recognition Letters, Vol. 7, pp.19-25, Jan. 1988.
8. Takeshi Agui, Masayuki Nakajima, Tae K. Kim, Eduardo T. Takahashi, "A Method of Recognition and Representation of Korean Characters by Tree Grammars," IEEE Trans. PAMI, PAMI-1, No. 3, pp.245-251, Jul. 1979.
9. B. Duerr, W. Haettich, H. Tropf, G. Winkler, "A Combination of Statistical and Syntactical Pattern Recognition Applied to Classification of Unconstrained Handwritten Numerals," Pattern Recognition, Vol. 12, pp.189-199, 1980.
10. Alan Ruttenberg, "Optical Reading of Typeset Music," Master of Science in Visual Studies at the MIT, 1991.
11. Howard Jay Siegel, et al., "PASM : A Partitionable SIMD/MIMD System for Image Processing and Pattern Recognition," IEEE Trans. on Computers, Vol. C-30, No. 12, pp.934-947, 1981.
12. 김재희, 인공지능의 기법과 응용, 교학사, 1989.
13. 편집부, 기보법, 도서출판 다라, 1992.
14. 편집부, 컴퓨터 미디, 도서출판 세운, 1991.
15. 이택수, 컴퓨터 미디 이용과 활용, 크라운 출판사, 1991.
16. Steve De Furia and Joe Scacciaferro, The MIDI Programmer's Handbook, M & T Publishing, Inc., 1989.
17. Robert Cyper, Jorge L. C. Sanz, "SIMD Architecture and Algorithms for Image Processing and Computer Vision," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 37, No. 12, pp.2158-2173, 1989.
18. Azriel Rozenfeld, "Parallel Image Processing Using Cellular Arrays," IEEE Computer, Vol. 16, No. 1, pp.14-20, 1983.
19. Azriel Rozenfeld, Angela Y. Wu, "Parallel Computer for Region-Level Image Processing," Pattern Recognition Vol. 15, No. 1, pp.41-50, 1982.



黃 永 吉(Yeong-Gil Hwang) 정회원
1985년 3월 ~ 1989년 2월 : 경북대학교 전자공학과 학사
1991년 8월 ~ 1993년 8월 : 서강대학교 전자계산학과 석사
1993년 7월 ~ 현재 : 삼성전자 통신개발실 연구원

金 聖 天(Sung Chun Kim) 정회원
1975년 : 서울대학교 공과대학 공업교육학 (전기전공) 학사
1976년 ~ 1977년 : 동아컴퓨터(주) Sys. Eng.
1977년 ~ 1978년 : 스페리유니백 Sales Rep.
1979년 : Wayne State Univ. 컴퓨터공학 석사
1982년 : Wayne State Univ. 컴퓨터공학 박사
1982년 ~ 1984년 : 캘리포니아주립대 조교수
1984년 ~ 1985년 : 삼성반도체(주) 책임연구원
1986년 ~ 1989년 : 서강대학교 공과대학 전자계산소 부소장
1989년 ~ 1991년 : 서강대학교 공과대학 전자계산학과 학과장
1985년 ~ 현재 : 서강대학교 공과대학 전자계산학과 조교수 (1985. 8 ~ 1987. 8), 부교수(1987. 9 ~ 1992. 8), 교수(1992. 9 ~ 현재)
1989년 ~ 현재 : 한국정보과학회 병렬처리시스템 연구회 부위원장, 대한전자공학회 및 한국통신학회 논문지 편집위원(1991, 1993), 한국정보과학회 학회지 부위원장(1993)
※주관심분야 : 병렬처리시스템(Parallel Computer Architecture, Interconnection Network), Neural Network, Computer Network