이 학 석 사 학 위 논 문

획의 방향 코드 조합에 의한 오프라인 필기체 숫자 인식에 관한 연구

2002년 8월

부 경 대 학 교 대 학 원

전 자 계 산 학 과

이 찬 희

이 학 석 사 학 위 논 문

획의 방향 코드 조합에 의한 오프라인 필기체 숫자 인식에 관한 연구

지도교수 정 순 호

이 논문을 이학석사 학위논문으로 제출함

2002년 8월

부 경 대 학 교 대 학 원

전 자 계 산 학 과

이 찬 희

이찬희의 이학석사 학위논문을 인준함

2002년 6월 29일

주　　심　공학박사　김 영 봉　(인)

위　　원　공학박사　박 지 환　(인)

위　　원　공학박사　정 순 호　(인)

목 차

표 목차

그림목차

A Study on Off-line Handwritten Digit Recognition Using Combination of Stroke Direction Codes

Chan-Hee Lee

Dept. of Computer Science
Graduate School Pukyong National University

Abstract

Off-line recognition of handwritten digit recognition consists of two steps; the feature extraction step and the recognition step.

We propose the improved methods for both steps. In the former step, the improved Self-Organizing Graph(SOG*) is applied for extracting key features of digits. The SOG* thinning method is a variant of Self Organizing Graph(SOG) with incremental learning algorithm.

In the latter step, the feature points and several feature stroke information of digits are suggested to apply to the resultant pixels(neurons) of SOG*, a multi-layer neural networks learn the knowledge of their combination for each digit, and the final learned neural network system correctly classifies an off-line handwritten

digit into the corresponding one.

We experiment these recognition system by using the digit database of KAIST, YONSEI university, and Canada Concordia university as the training handwritten digits. From these experiments, we confirm that these proposed system is at correct recognition rate 98.85% to 99.25% but at error rate 0.55% to 0.71% and at rejection rate 0.2% to 0.45%. These results are equal or superior to the existing ones in terms of correct recognition. Therefore the proposed methods enhance the correctness and the speed of recognition of off-line handwritten digits.

1. 서론

오프라인 필기체 문자 인식의 기본이 되는 오프라인 필기체 숫자 인식은 우편 봉투의 자동 분류, 전표나 민원 서류와 같은 각종 서식의 입력 자동화 등 그 응용 분야가 다양하다. 필기체뿐만 아니라 문자 인식에 있어 인식 성능 향상을 위해서 일반적으로 두 가지 측면을 고려하게 된다. 그 첫 번째는 실제 인식 단계에서 입력으로 사용될 인식 문자 패턴에 대한 전처리 및 특징 추출에서 인식에 필요한 정보를 최소화하는 것이다. 두 번째는 특정 패턴에 국한되지 않는 일반화 및 객관적 적응 능력을 가지면서 유사 패턴에 대한 인식률을 높일 수 있는 분류기 적용에 관한 것이다[1-7].

이 두 가지 측면 중에서 전자에서 행하는 전처리 단계에서는 인식 성능의 향상을 위해 패턴의 형태를 다양하게 변형하는 방법이 사용되며 이 변형 방법은 크게 선형 형태 변형(Linear Shape Variation)과 비선형 형태 변형(Non-linear Shape Variation)으로 분류할 수 있다. 선형 형태 변형 방법으로는 이동, 신축, 회전, 밀림 등의 정규화 과정이 포함되며, 비선형 형태 변형 방법으로는 점밀도, 선밀도, 획밀도 등의 정규화가 포함된다[5-7]. 또한 전자의 측면에서는 전처리 단계의 세선화 과정을 통하여 방대한 이미지 정보를 축소시켜 이 축소된 정보가 인식을 위한 다음 단계로 사용됨으로써 효율을 높이게 된다. 따라서 세선화에 있어 가장 중요한 것은 원래 이미지의 구조적 특징을 유지하는 것이다[8].

지금까지의 최적 세선화 알고리즘은 크게 직접 기법과 간접 기법으로 나눌 수 있다. 직접 기법은 원래 이미지로부터 픽셀을 하나씩 제거해 나

감으로써 세선화를 수행하는 것이고, 간접 기법은 이미지의 특성을 반영하는 골격선을 생성하는 것이다[8,9]. 두 가지 기법 중에서 이미지의 특성을 반영하여 세선화를 수행하는 간접 기법이 직접 기법에 비해 우수한 성능을 갖는다[10].

전처리에서 세선화를 포함한 많은 변형을 사용하는 방법에 비해 신경회로망이 갖는 적응적 특성으로 인하여 신경회로망을 이용한 필기체 숫자 인식에 관한 연구가 활발히 진행되고 있다. 그러나 기존 신경회로망을 이용한 방식들은 특징 추출의 미비로 인해 다양한 형태의 왜곡을 흡수하지 못하는 문제점이 있거나 신경회로망이 갖는 적응적 특성을 무시한 채 인식률을 높이기 위하여 전처리 과정에서 많은 형태의 변형들을 사용함으로써 전처리 과정이 복잡해지며, 인식기의 입력도 많아지게 되는 문제점을 갖는다.

따라서, 본 논문에서는 필기체 숫자가 지니고 있는 특징에서 특징 점을 추출하여 특징 점을 기준으로 획으로 분리한 후 각 획들을 8방향 코드로 추출하여 이들 획의 조합으로 숫자를 인식하는 필기체 숫자 인식 시스템을 제안한다.

전처리 과정에서는 고속 세선화 기법인 SOG*(Improved SOG(Self-Organizing Graph)) 세선화를 이용하여 세선화를 수행한 후 그 결과로 나온 세선화 이미지에서 숫자의 특성을 파악하여 특징 점을 선택하고 특징 점을 기준으로 각 획을 분리하게 된다. 분리된 각각의 획은 8방향 코드로써 획 인식기의 입력으로 사용되어 그 결과로 나온 각 획들을 조합하여 최종적으로 숫자를 인식하는 오프라인 필기체 숫자 인식 시스템을 구현한다.

본 논문의 2장에서는 전처리 단계의 세선화 기법에서 SOM, SOG 그리고 SOG* 세선화 기법을 살펴보고, 3장에서는 SOG*에 이은 전처리 과정에서 특징 추출을 소개하며, 4장에서는 오프라인 필기체 숫자 인식 시스템을 설계 및 구현과 실험을 통해 여러 숫자 이미지 데이터베이스에 대한 인식 결과를 제시하며, 마지막 5장에서는 결론 및 향후 연구 과제를 언급하고자 한다.

2. 세선화

이 장에서는 오프라인 필기체 숫자 인식 시스템을 구현하는데 필요한 전처리 과정 중에서 세선화 기법에 관해 설명한다. 신경회로망을 이용한 간접 세선화 기법들을 소개하고 고속 세선화 기법인 SOG* 세선화 기법에 대해 설명한다.

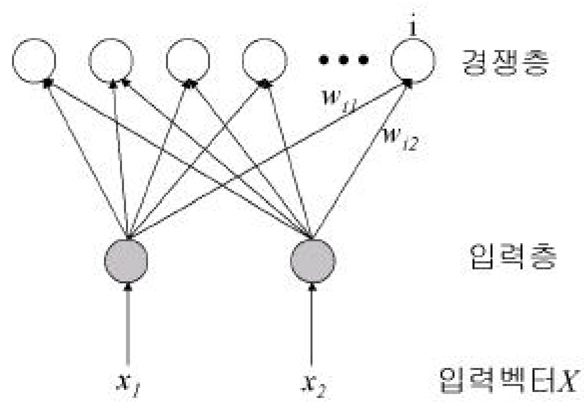
간접 세선화 기법 가운데 신경회로망을 이용한 방법으로 이미지 구조에 따라서 자신을 조직하는 특성을 갖는 Kohonen의 자기 구성 지도 (Self-Organizing feature Map : SOM)를 이용한 세선화 연구가 진행되었고, 이 방법의 단점을 개선한 자기 구성 특징 그래프(Self-Organizing feature Graph : SOG)가 제안되었다[10,11]. 이 SOG 방법은 템플릿 윈도우를 이용한 세선화 기법[12]과 ART2 신경회로망을 이용한 클러스터링 세선화 기법[13]에 비하여 그 과정이 단순하고 세선화 결과가 우수하다. 그러나 학습 단계에서 뉴런들을 재 정돈하는 단계로 인하여 학습 시간이 오래 걸리는 단점을 갖는다[14].

2.1 관련 연구

2.1.1 자기 구성 특징 지도(Self-Organizing feature Map : SOM)

선형 네트워크에서 SOM의 경쟁층의 각 뉴런들은 두 개의 인접 뉴런들을 가지며 그 인접성은 항상 고정되어 있다. SOM은 입력층과 경쟁층의 두 개의 층으로 구성되어 있으며, 두 개의 층은 완전 연결되어 있다.

<그림 1>은 이차원 입력 벡터와 일차원 경쟁층을 갖는 Kohonen Network을 보여준다.



<그림 1> 이차원 입력 벡터와 일차원 경쟁층을 갖는 Kohonen Network

<Figure 1> Kohonen Network in two-dimensional input layer and one-dimensional competitive layer

이차원 입력 벡터와 일차원 경쟁층을 갖는 SOM에서 입력층에 입력 패턴이 들어오면 경쟁층의 각 뉴런들은 식 (1)에 의해 승자 뉴런을 찾게 된다[15,16].

$$winning \quad unit \quad c = \min_{1 \le i \le n} \left\| X - W_i \right\| \tag{1}$$

여기서 X는 입력 벡터이고 W_i는 경쟁층의 i 뉴런에 연결된 가중치 벡터를 의미한다. 위의 <그림 1>에서 볼 때 $X=(x_1, x_2)$이고, $W_i=(w_{i1}, w_{i2})$

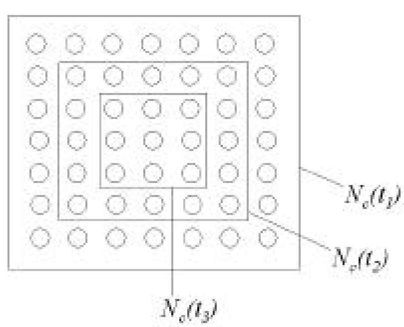
를 의미한다. 승자 뉴런이 결정되면 승자 뉴런과 그 이웃에 속하는 뉴런들의 가중치들은 식 (2)에 따라 학습된다.

$$W_i(t+1) = \begin{cases} W_i + \alpha(t)(X - W_i(t)) , & i \in N_c(t) \\ W_i(t) , & otherwise \end{cases} \quad (2)$$

식 (2)에서 N_c(t)는 t 시간에서 뉴런 c의 이웃을 말하며 α(t)는 학습률을 나타낸다[15,16]. 이 학습률은 학습이 진행되는 동안 계속 변하게 되며 변화되는 방법은 다음 식 (3)과 같다.

$$\alpha(t) = \alpha(0)(1 - \frac{t}{T}) \quad (3)$$

식 (3)에서 t는 현재 학습 iteration을 의미하며 T는 수행될 전체 학습 iteration을 의미한다. 이웃 뉴런들의 개수의 변화는 <그림 2>와 같이 나타낼 수 있으며 그 방법은 식 (4)와 같이 표현할 수 있다.



<그림 2> 이웃 뉴런의 개수(반경) 크기 변화

<Figure 2> Change of number(radius) size of neighborhood neuron

$$c - d < x < c + d$$
$$c - d < y < c + d \qquad (4)$$

승자 뉴런 c로부터 다른 뉴런으로의 거리를 d라 하면, c의 이웃 뉴런들은 위 식 (4)와 같이 모든 (x, y)가 된다. 식 (4)에서 d의 값은 다음 식 (5)와 같이 변한다[15].

$$d(t) = \left\lceil 1 - \frac{t}{T} \right\rceil \qquad (5)$$

2.1.2 자기 구성 특징 그래프(Self-Organizing feature Graph : SOG)

SOG는 SOM에서 학습되는 동안 고정된 인접성을 입력 벡터로부터의 거리에 따라 뉴런들을 재 정렬함으로써 수정하게 된다. 재 정렬된 뉴런들은 네트워크의 재조직에 있어 향상된 능력을 가지며 이 효과를 학습 알고리즘에 적용하기 위하여 SOM의 가중치 수정 방법인 식 (2)를 다음의 식 (6)과 같이 수정하여 사용하게 된다[10].

$$w_i(t+1) = w_i(t) + \zeta(t, K_i) \times (X - w_i(t))$$
$$\zeta(t, K_i) = \exp - \left[\frac{\ln(K_i)}{\alpha(t)} \right]^2 \qquad (6)$$

식 (6)에서 X는 입력 벡터이고, K_i는 정돈된 뉴런들의 초기 인덱스 값

이며, $\alpha(t)$는 학습률을 나타낸다.

SOG의 기본 구조는 SOM과 동일하며, 단지 가중치의 수정에 있어 식 (6)을 사용하여 각 뉴런들의 정렬 순서도 가중치 수정의 우선 순위에 영향을 미치도록 한다.

2.1.3 L* 학습 알고리즘

L* 학습 알고리즘은 기존 SOM에서 경쟁층의 승자 뉴런을 찾는데 소요되는 시간을 줄여 전체 학습 시간을 감소시키는 알고리즘이다[17].

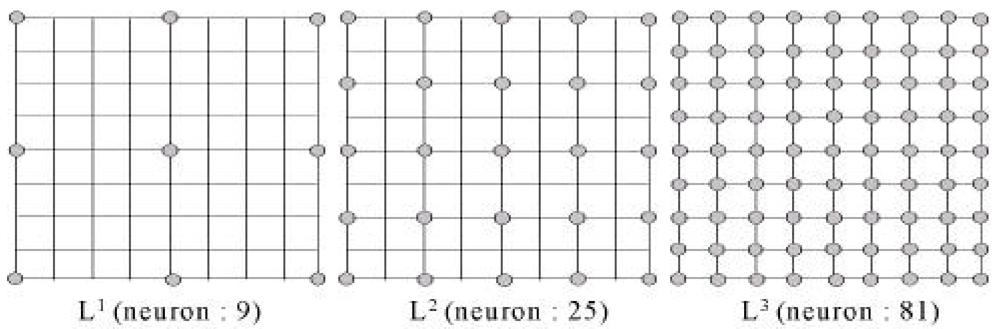
SOM이 승자 뉴런을 찾는 방식은 모든 뉴런을 대상으로 하지만, L* 알고리즘은 승자 뉴런을 찾을 때 모든 뉴런을 대상으로 하는 것이 아니라 단계별로 뉴런들의 수를 점차 늘여가면서 승자 뉴런을 찾기 때문에 기존의 SOM 방식보다 계산 시간이 단축되는 효과를 얻을 수 있다.

L* 알고리즘은 1단계부터 $\log_2(M-1)$ 단계까지 학습을 한다. 여기서 M은 경쟁층 맵의 크기를 의미한다. 이때 각 단계별로 학습할 뉴런의 개수는 1차원 맵의 경우 $2^{level}+1$이고, 2차원 맵의 경우는 $(2^{level}+1)^2$이 된다. level은 각 단계를 의미한다. 단, 각 단계별로 학습할 뉴런의 개수에 실험치로 결정되는 학습 계수를 곱한 만큼을 학습 회수로 결정한다. 학습 계수가 20 이상이면 완전한 자기 구성 특징 맵을 구성할 수 있다[17].

한 단계가 끝나면 다음 단계에 학습할 뉴런을 추가해야 하는데, 이때 추가하는 뉴런들은 이전 단계에서 학습된 뉴런들을 이용한다.

다음 <그림 3>에서 이차원 9×9 맵, 즉 81개의 경쟁층을 가지는 맵에서 각 단계별로 학습해야 할 뉴런들을 나타내고 있다. 1단계에서는

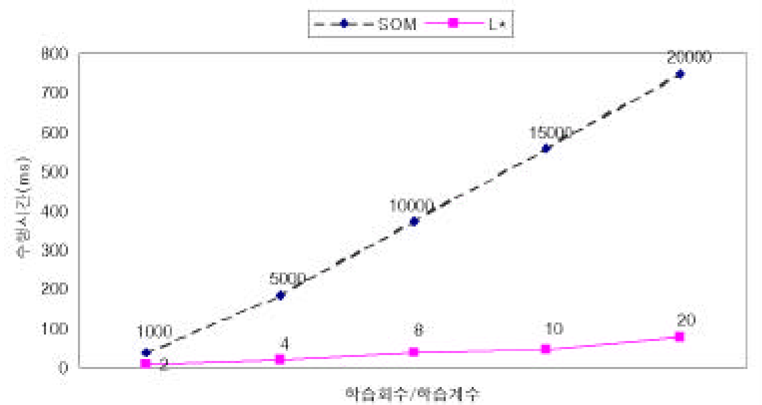
$(2^1+1)^2=9$의 수만큼 뉴런을 선택하는데, 맵에서 코너에 해당하는 뉴런과 그 중심 뉴런이 선택된다. 2단계 이후부터는 이전 단계에서 선택된 뉴런들 사이에서 선택되며, 최종 단계에서는 전체 뉴런이 선택된다.



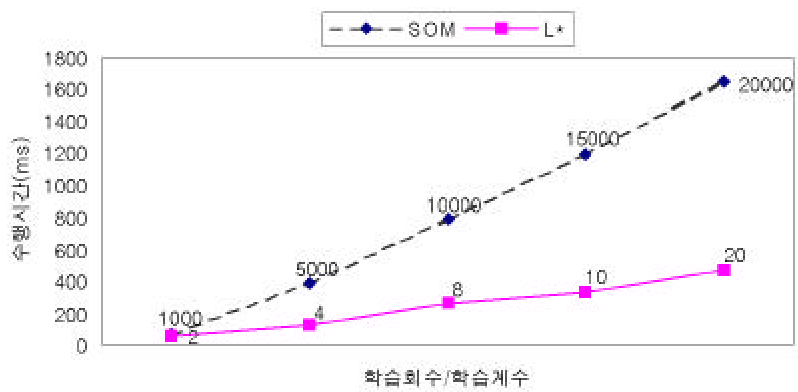
<그림 3> 9×9 맵에서 단계별로 학습 할 뉴런

<Figure 3> Neuron to learn by levels in the 9×9 map

<그림 4>에서는 각각 기존의 SOM 방식과 L^* 방식에 대한 이차원 9×9, 17×17 맵에 대한 시간 측정의 결과를 나타내었다. SOM은 학습 회수가 20,000번일 때 완전한 자기 구성 맵을 구성하였고, L^* 알고리즘은 학습 계수(Λ)가 20일 때, 즉 학습 회수가 $\sum_{level=0}^{\log_2(M-1)} \Lambda \times (2^{level}+1)^2$이 될 때 완전한 자기 구성 맵을 구성하였다. 즉, 9×9 맵에서 SOM은 20,000번의 학습으로 완전한 맵을 구성하고, L^* 알고리즘은 2,300번의 학습으로 완전한 맵을 구성할 수 있는 것이다. 실험은 P-Ⅲ 1GHz 시스템에서 테스트하였다.



(a) 9×9 맵의 수행 시간 비교



(b) 17×17 맵의 수행 시간 비교

<그림 4> SOM과 L^* 수행 시간 비교

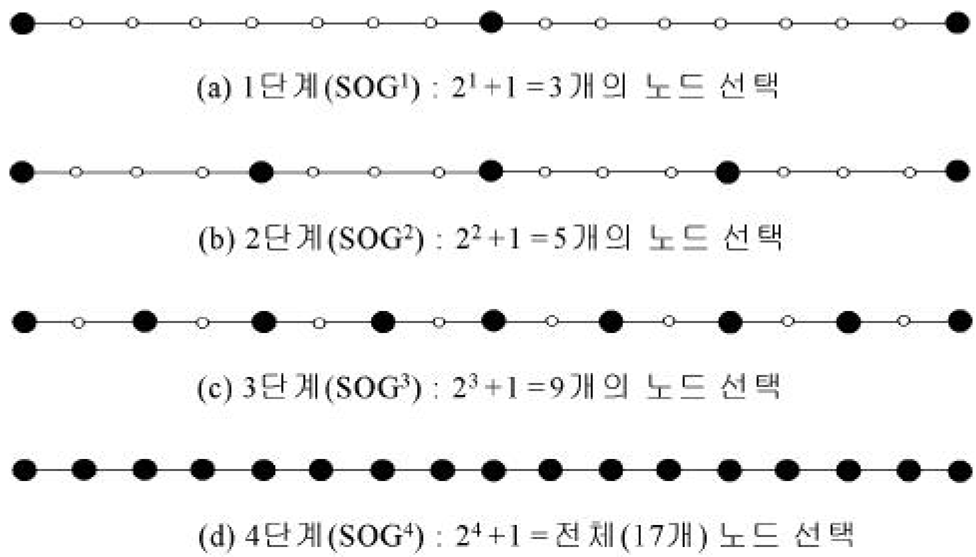
<Figure 4> Comparison of execution time of SOM and L^*

2.2 개선된 자기 구성 그래프(Improved SOG : SOG*)

2.2.1. SOG* 학습 알고리즘

SOG 알고리즘은 학습이 끝날 때까지 각 단계에서 입력이 들어올 때마다 입력 패턴과 경쟁층의 뉴런들과의 거리를 정렬해야 한다[10]. 이 학습 과정에서 뉴런들이 각 입력 패턴의 구조를 나타내는 위치를 찾아 안정화되는데, 이렇게 안정화를 이루게 되면 더 이상 학습할 필요가 없음에도 불구하고 기존 SOG는 안정화 후에도 계속된 학습으로 많은 시간을 소비하는 단점을 가지게 된다. 이러한 단점을 개선하기 위하여 SOG의 변형을 사용하는데, 이는 SOG에서 안정화되는 시점을 찾아 이후 학습에서는 기존 정렬을 그대로 사용하는 방법이다. 또한, 세선화를 위한 학습 과정에서 경쟁층의 뉴런을 재 정돈하는 과정을 거쳐서 승자 뉴런을 선택할 때 모든 뉴런을 한번에 모두 재 정돈하는 것이 아니라 재 정돈할 뉴런의 수를 단계별로 증가 시켜가며 학습시키는 점증적 학습 알고리즘인 L* 알고리즘을 사용하여 세선화를 수행하는 방법이다[14].

이 과정은 <그림 5>와 같이 나타낼 수 있다. 첫 번째 단계인 SOG^1은 <그림 5>의 (a)와 같이 검은색의 노드로 표현되며 제일 끝의 두 노드와 그들 중심의 노드가 선택되어 학습을 하게되며, 두 번째 이후 단계부터는 이전 단계에서 선택되어진 n개의 노드들 사이에서 선형 보간법[22]을 이용하여 n-1 개의 새로운 노드를 선택하게 된다. <그림 5>에서 검은색으로 표시된 굵은 노드는 현재 단계 또는 이전 단계에서 선택되어진 노드를 나타내며 흰색으로 표현된 노드는 현 단계에서 학습에 참여하지 않는 노드를 나타내고 있다.



<그림 5> SOG*의 학습 과정

<Figure 5> Learning process of SOG*

이러한 SOG* **알고리즘을 아래에 기술하였고**, **이는** L* **알고리즘을** SOG **변형에 적용한 방식이다**.

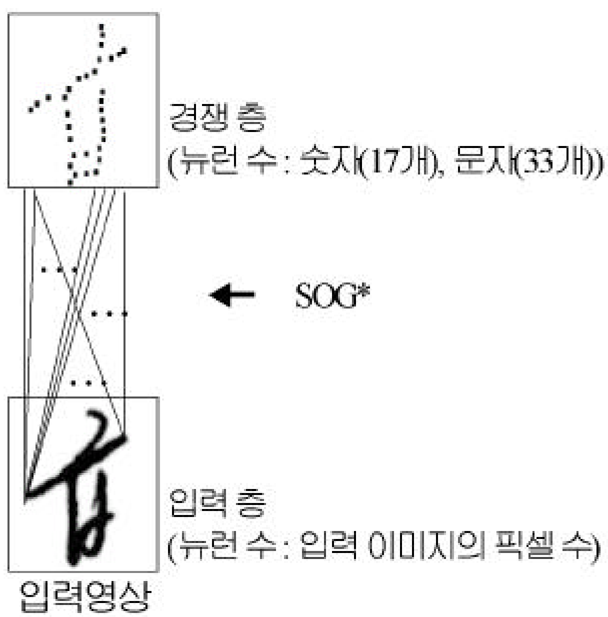
SOG* 알고리즘

```
for level := 1 to log2(M-1) do
  begin
      /* 학습할 뉴런 선택 */
      node := 2^level + 1;
      /* 가중치 부여 */
      give the weight of step;                                    ㉮

      for repeat := 1 to delta*level do    // delta : 학습 계수(Λ)
        begin
             for input := 1 to level do
               begin
                    /* 입력 패턴과 경쟁층 뉴런의 거리 정렬 */
                    if sorted orders = previous orders
                    then Ki := previous orders;
                    else 입력과 노드의 거리에 의한 정렬;           ㉯
                    /* 학습 */
                    update weight of winning unit;
                    update weight of neighborhood units;         ㉰
               end;                            // end of input
        end;                                   // end of repeat
  end;                                         // end of level  ㉱
```

이 알고리즘의 ㉮는 가중치를 수정하는 부분이며, 이 가중치는 최초 단계에서는 랜덤 값으로 부여하게 되지만, 두 번째 단계 이후부터는 이전 단계에서 선택된 뉴런들을 기반으로 가중치를 부여하게 된다. ㉯ 부분은 경쟁층 뉴런들을 재 정렬하는 부분이며, 입력 패턴이 들어오게 되면 승자 뉴런이 선택되어 지고 나머지는 입력 패턴과의 거리에 따라 정렬되는데 이때 이전의 정렬 순서와 현재의 정렬 순서가 같다면 가중치들이 안정화된 것을 의미하므로 더 이상의 정렬을 수행하지 않는다. ㉰는 실제 학습하는 부분이며 승자 뉴런과 이웃 뉴런의 가중치를 갱신하며 학습을 수행하게 된다.

이러한 SOG* 방법에 사용되는 신경망의 구조는 <그림 6>과 같은 이차원 입력 벡터와 일차원 경쟁층을 갖는 Kohonen Network이다.



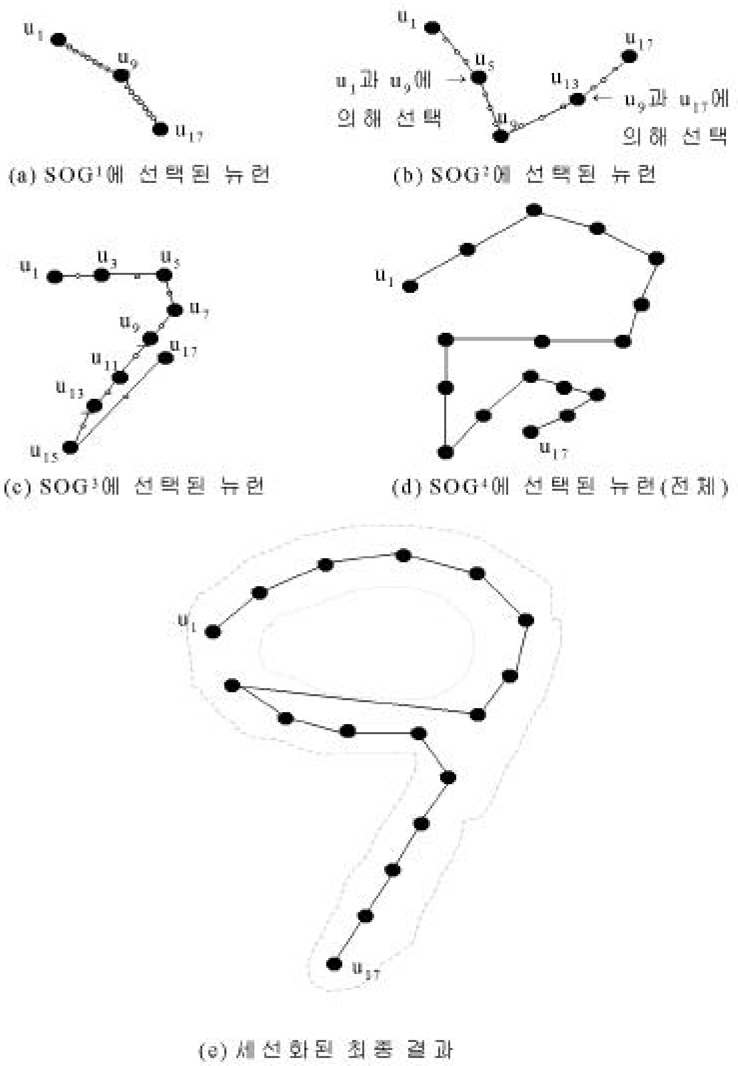
<그림 6> SOG* 구조

<Figure 6> Structure of SOG*

SOG*에서 입력 영상은 이차원 입력 구조로서 0과 1사이의 이차원 좌표로 변환되어 사용되며, 경쟁층은 실험을 통하여 숫자 이미지는 17개, 문자 이미지는 33개의 고정된 수의 뉴런들이 일차원적으로 이루어진다.

학습을 위해서는 각 단계별로 뉴런의 수를 증가시켜 가면서 학습을 하며 다음으로 뉴런과 입력과의 거리를 계산하고 이 값들을 다시 정돈한다. 마지막으로 이 정돈된 값들의 순서를 반영하여 가중치를 수정한다. 가중치를 수정하는 방법은 식 (6)과 같다.

숫자 이미지를 예로 들면, 17개의 뉴런으로 세선화를 수행하므로 총 4단계로 이루어진다. 각 단계별로 학습할 뉴런의 개수는 3, 5, 9, 17이 되며, 최초에 학습할 3개의 뉴런은 랜덤 가중치로서 결정하여 학습을 한다. 다음 단계에서는 5개의 뉴런이 학습에 참가하게 되는데, 이때는 이전에 학습한 3개의 뉴런과 각 뉴런들 사이에 연결되어 있는 뉴런들이 추가되어 학습하게 된다. 두 번째 단계부터는 선형 보간법을 이용하여 새로 추가되는 경쟁층 뉴런들의 가중치를 결정한다. 선형 보간법을 사용하는 이유는 이전 단계에서 학습된 뉴런들의 가중치를 그대로 유지함으로서 전체 뉴런들의 가중치들이 안정화되는 시점을 빨리 찾아 학습에 소요되는 시간을 단축하기 위해서이다. 세 번째 이후 단계도 두 번째 단계와 동일한 방법으로 수행하게 된다.



<그림 7> 숫자 9가 학습되는 과정

<Figure 7> Process that a digit of 9 gets to be learned

<그림 7>에서는 숫자 이미지 9에 대해 학습하는 단계를 설명하고 있으며, 그림에 표시된 u_n은 뉴런의 번호이다. 최초 학습 단계에서는 <그림 7>의 (a)와 같이 3개의 뉴런이 선택되어 학습하게 되는데, 이때 선택

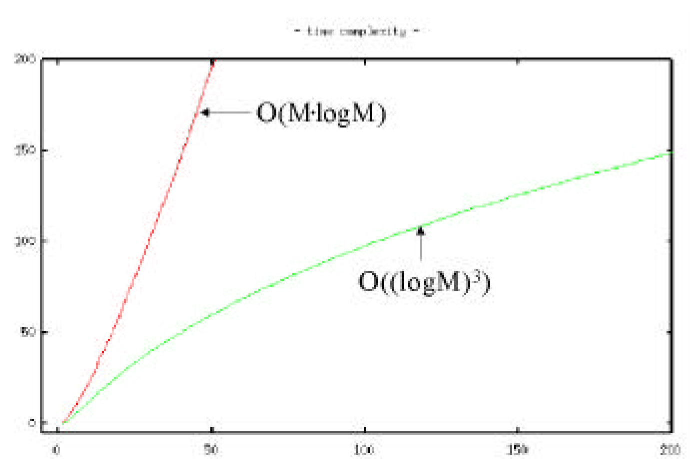
되는 뉴런은 외곽의 두 뉴런과 그들의 중심 뉴런인 u_1, u_9, u_{17}이 선택되게 된다. 두 번째 단계는 <그림 7>의 (b)와 같이 이전 단계에서 선택되어진 뉴런들 사이에 있는 뉴런들을 선택하여 학습에 참여하게 된다. 이후 단계에서도 두 번째 단계와 동일한 방법으로 <그림 7>의 (c), (d)와 같이 수행하게 된다. <그림 7>의 (e)는 4단계까지 모든 과정이 끝난 최종 세선화 결과를 나타내고 있다. <그림 7>에서 검은색으로 표시된 뉴런은 현 단계에서 선택되어 학습에 참여하는 뉴런이고, 작은 점으로 표시된 뉴런은 현 단계에서 선택되지 않은 뉴런을 나타내고 있다.

2.2.2 SOG* 알고리즘의 분석

기존의 SOG는 학습 시에 항상 모든 뉴런들을 참가시키기 때문에 O(MlogM)의 시간 복잡도를 나타낸다. 반면, SOG* 세선화 기법은 학습 시에 점증적으로 뉴런들을 참여시킨다. 단계는 1단계부터 $\log_2(M-1)$ 단계까지고, 각 단계별로 학습 계수를 곱한 만큼을 학습 회수로 수행하게 된다. 따라서 식 (7)과 같은 수식을 유도할 수 있다. 여기서 level은 단계를 말하며, C는 단계 당 학습 회수를 의미한다. 유도된 수식에 의해서 SOG* 세선화 기법의 시간 복잡도가 $O((\log M)^3)$임을 알 수 있다.

$$
\begin{aligned}
& C \cdot \sum_{level=1}^{\log(M-1)} (2^{level}+1) \cdot \log(2^{level}+1) \\
&= C \cdot (3 \cdot \log 3 + 5 \cdot \log 5 + \Lambda + M \cdot \log M) \\
&\le C \cdot (\sum_{k=1}^{\log(M-1)} k \cdot \log k) \\
&\le C \cdot (\sum_{k=1}^{\log(M-1)} k^2) \\
&= O((\log(M-1))^3) = O((\log M)^3)
\end{aligned} \tag{7}
$$

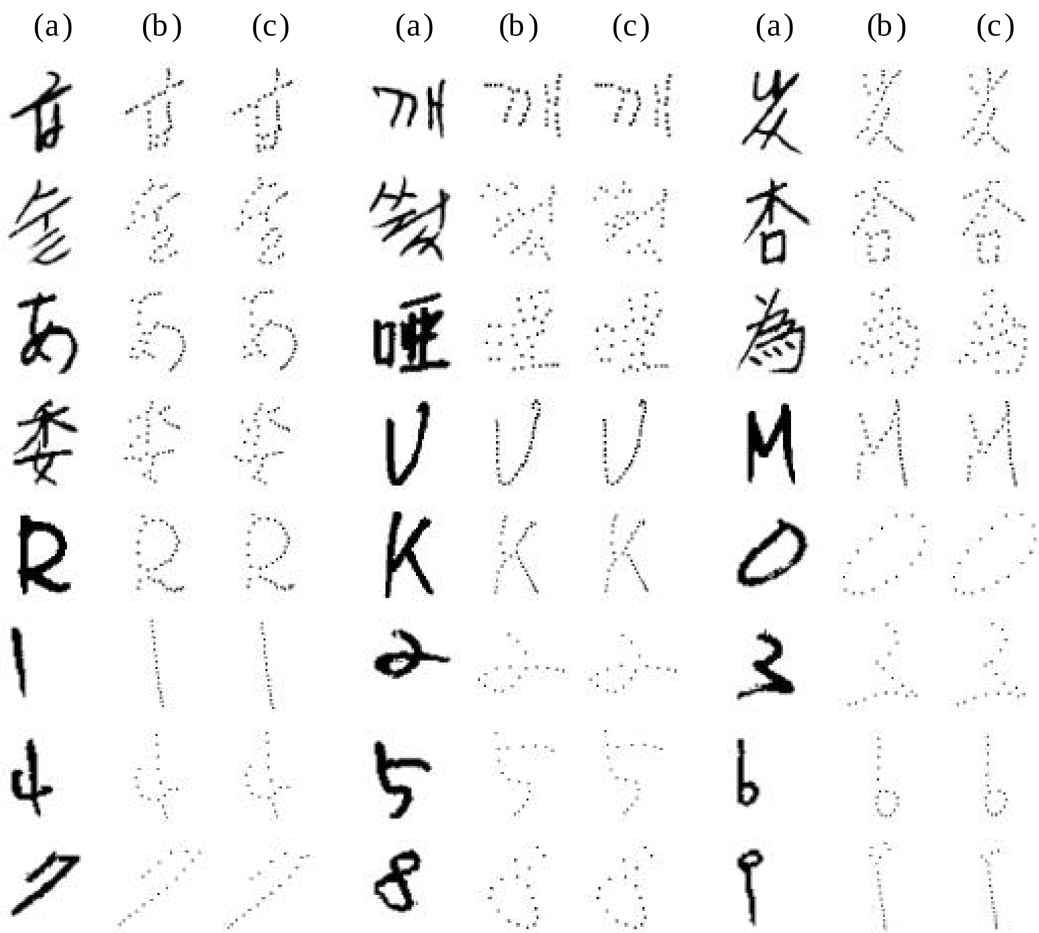
또한 기존 SOG와 SOG* 세션화 기법의 시간 복잡도를 비교한 그래프를 <그림 8>에 나타내었다. SOG는 뉴런의 수가 증가함에 따라 기하급수적으로 증가하는 반면 SOG* 세션화 기법은 증가량이 아주 적음을 알 수 있다.



<그림 8> 두 알고리즘의 시간 복잡도

<Figure 8> Time complexity of two algorithms

다음 <그림 9>에는 기존의 SOG와 SOG* 세선화 기법의 실험 결과를 보여주고 있다. SOG* 세선화 기법은 기존의 SOG와 같이 우수한 세선화 결과를 냄을 알 수 있다.



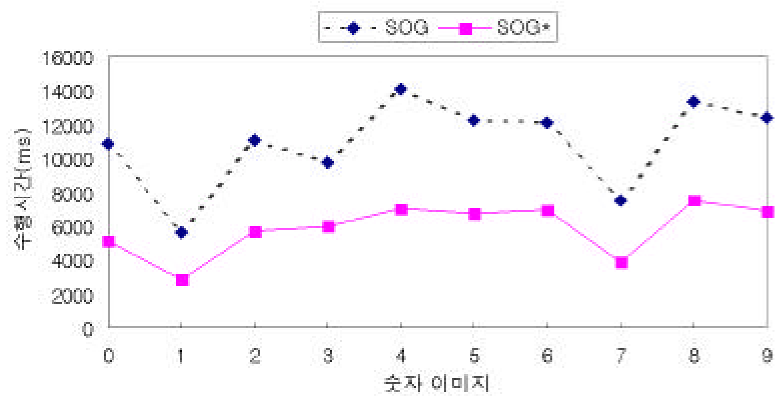
(a) 원 이미지, (b) SOG* 세선화 결과, (c) SOG 세선화 결과

(a) source image, (b) results of SOG*, (c) results of SOG

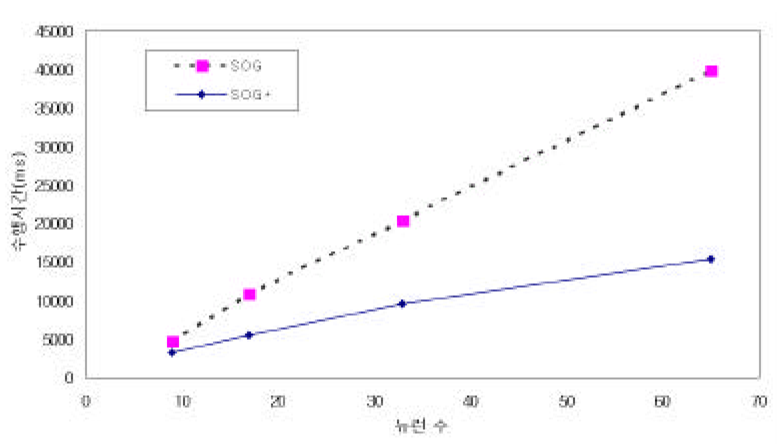
<그림 9> 세선화 결과

<Figure 9> Thinning results

<그림 10>에서는 SOG와 SOG* 각각에 대해 각 숫자별 알고리즘 수행 시간과 뉴런의 개수에 따르는 수행 시간을 비교하여 세선화 결과뿐 아니라 그 속도면 에서도 SOG*가 우수함을 보여주고 있다. 실험은 역시 P-Ⅲ 1GHz의 시스템에서 실험하였다.



(a) 숫자별 알고리즘 수행 시간



(b) 뉴런 수에 따른 수행 시간

<그림 10> SOG와 SOG*의 수행 시간 비교

<Figure 10> Comparison of execution time between SOG, SOG*

3. 특징 추출과 획 인식

이번 장에서는 필기체 문자 및 숫자를 인식하는 다른 방법들과 본 논문에서 사용한 방법을 살펴보도록 하겠다.

필기체 숫자는 필기자의 습관이나 필기 도구 등에 의해 많은 변형을 가지게 된다. 특히 오프라인 필기체 숫자 인식의 경우에는 앞의 변형들과 함께 스캐너와 같은 입력 장치에서 발생하는 잡음 등도 고려를 해야 한다. 이로 인해 기존 논문의 경우에는 전처리 과정에서 하나의 특징만이 아닌 복합 특징을 이용한 단일 신경회로망 또는 다중 구조의 신경회로망을 이용하여 인식하는 방법을 사용하였다[3-5,19,20,23-26]. 그러나 이러한 방법들은 전처리에서 많은 시간을 소비하며 인식기가 커지는 문제점을 가지게 된다. 따라서 본 논문에서는 이러한 전처리의 복잡성을 단순화시키는 방법으로 세선화 된 결과를 이용하여 먼저 특징 점을 추출하고 획을 분리하여 각 획들의 방향 코드 특징을 추출하여 획 인식기의 입력으로 사용하여 전처리의 단순화, 속도 향상과 인식기의 축소, 인식률의 향상을 이룬다.

3.1 관련 연구

기존 논문에서 필기체 문자 및 숫자를 인식하기 위하여 많은 특징들을 사용하였다. 문자 및 숫자의 원 이미지와 변형된 이미지를 함께 특징으로 사용하여 인식하게 되는데 그 방법으로 국부 특징과 전역 특징, 선형 형태 변환 및 비선형 형태 변환 등의 방법을 주로 사용하였다. 그 방법

들 중에 몇 가지를 살펴보도록 하겠다.

임길택, 남윤석, 진성일[19]은 회전과 이동 영상을 이용하여 모듈 구조의 신경망 기반 인식 방법을 사용하였다. 전체 숫자 패턴 공간을 서로 중첩되는 여러 개의 작은 군집으로 나누어 신경망으로 구현하였다. 학습 단계에서 입력 숫자 영상의 회전과 디더링 연산을 한 후 이 새로운 숫자 영상을 이용하여 학습하고 인식 단계에서는 원 영상과 회전 및 이동된 영상들을 함께 사용하여 인식하는 방법을 사용하여 여러 개의 출력 값이 나오면 이들을 다시 결합하여 최종 인식을 하는 방법을 사용하였다. 이 방법에서 사용한 특징은 원 이미지, 회전 이미지, 이동 이미지 등의 세 가지다. 이미지 자체를 특징으로 사용하게 되므로 그 특징 정보의 양이 많아질 수밖에 없다.

김백섭, 송혜정[20]은 전처리에서 잡음 제거, 기울기 교정, 윤곽선에 대한 평활화를 거친 후 특징 추출에서 지역적 특징으로 윤곽 화소간의 방향을 사용하였고, 구조적 특징으로 화소간의 방향 정보를 긴 거리로 확장한 작은 획의 구조를 이용하였으며 전역적 특징으로 면적과 폭과 높이의 비율을 이용하였다. 지역적 특징과 구조적 특징을 체인 코드로 변환하여 테이블을 만든 후 이후 인식에서는 거리함수를 이용하여 최소거리 분류 규칙을 따르는 k-NN(k-Nearest Neighbor) 알고리즘으로 인식하는 방법을 사용하였다. 이 방법에서는 전처리에서의 계산을 거친 후에 특징 추출에서는 전역적 특징과 지역적 특징 등을 모두 사용하였고 이를 다시 체인 코드로 변환하여 특징을 추출하였다. 이 역시 많은 특징을 사용하여 그 정보의 양이 많다.

김현돈, 조성배[21]는 구조 적응형 자기 구성 지도(Self-Organizing feature Map : SOM) 모델에서 유전자 알고리즘을 이용하여 분화된 노드의 가중치를 결정하여 이차원 패턴 공간 데이터에 대한 실험과 오프라인 필기체 숫자 데이터에 대해 실험을 하였다. 필기체 숫자에 대한 실험에서 위상 보존을 측정하여 입력 벡터로부터 출력 공간에서 n개의 승리자를 구하고, 출력 노드로부터 일정 거리 안에서 n개의 승리자를 구하여 각각의 n개의 승리자의 값을 비교하여 위상 보존 정도를 평가하였다. 이 방법은 특징의 양을 최소화하기 위해 유전자 알고리즘과 신경망의 결합을 이용하는 방법으로 가중치를 결정하여 인식하는 방법을 제안하였다.

김종석, 홍연찬[22]은 클러스터 신경망을 이용하여 우편번호 인식 시스템을 설계하였는데 우편봉투에서 숫자 분리 후 각 숫자를 클러스터 신경망을 이용하여 인식하는 방법을 사용하였다. 사용된 특징으로는 망 특징과 Kirsch 마스크를 이용한 4방향 특징을 사용하였다. 이 방법은 특징을 두 가지만을 사용하였지만 망 특징과 마스크를 이용한 방향 특징을 추출하는데 소요되는 계산량이 많다.

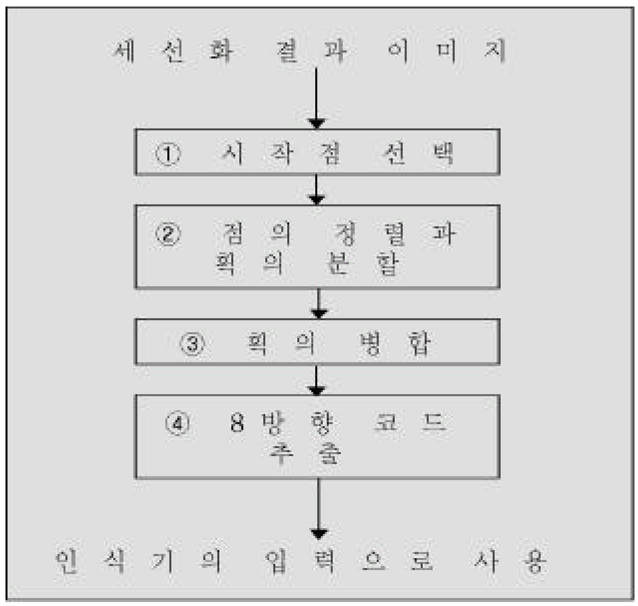
3.2 특징 추출과 획 인식

3.2.1 특징 점의 추출과 획의 분리

특징 점이란 세선화 된 이미지에서 어느 한 점에서 인접한 점이 여러 개가 존재하고 이 점 위치에 해당하는 원 이미지 상에 여러 개의 획이 존재하는 분기점으로 정의한다[26].

특징 점 추출과 획의 분리는 동시에 이루어지며 각 과정은 다음 <그

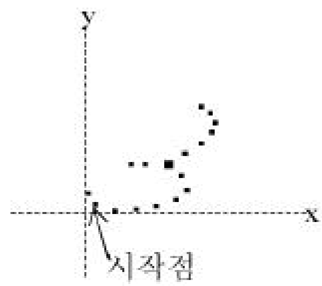
림 11>과 같다. 이 과정은 세선화의 결과로 나온 17개의 점으로 각 점과 점 사이의 거리와 방향, 원 이미지와의 관계성으로 획을 특징짓기 위한 과정이다.



<그림 11> 특징 점 추출과 획의 분리 과정

<Figure 11> Process of extract of feature point and separate strokes

첫 번째 과정은 <그림 11>의 ① 과정으로 세선화의 결과로 나온 이미지에서 시작점을 선택하게 되는데, 이때의 시작점은 이미지의 사각 절단면을 x, y 축의 이차원 공간으로 생각해서 이차원 평면상에서 최소점(원점)을 기준으로 가장 가까운 점을 시작점으로 선택하게 된다. 이는 다음 <그림 12>와 같이 결정한다.



<그림 12> 시작점의 선택

<Figure 12> Select of start point

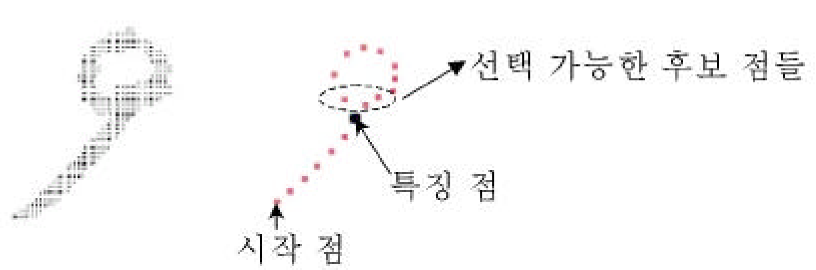
두 번째로 <그림 11>의 ② 과정은 시작점을 기준으로 다음 점들을 정렬하는 과정이다. 정렬을 수행하는 과정은 먼저 현재 점을 기준으로 가장 가까운 3개의 점을 선택하며 현재 점이 시작점이라면 선택된 각 점들과 현재 점 사이에 원 이미지에서 글자 부분이 존재하는지의 여부를 확인하여 다음 점을 선택하고, 둘 이상의 점이 선택 가능하다면 가장 가까운 점을 선택한다. 이때 원 이미지를 검색하는 이유는 거리가 가깝다고 하여 획이 아님에도 선택되어 지는 경우를 방지하기 위해서이다. 반면에 현재 점이 시작점이 아니라면 시작점과 동일한 방법을 거치되 이전 점과의 기울기까지 고려하여 인접 점을 선택한다. 이는 최적의 획을 구해내기 위한 과정이다. 이때 현재 점에서 둘 이상의 인접 점이 세 가지의 조건을 모두 만족하여 선택 가능하다면 여러 개의 획이 분산되어 존재함을 의미하므로 바로 이 점이 특징 점이 되는 것이다. 또한 점과 점 사이의 거리가 실험적인 임계치 이상이고, 원 이미지 상에 글자 부분이 존재하지 않는다면 획을 분리하게 된다.

<그림 13> SOG* 세선화 결과 이미지에서의 특징점 추출

<Figure 13> Feature point by result of SOG* thinning image

<그림 13>에 원 이미지와 세선화 결과 이미지에서 선택된 특징 점을 보여 주고 있다. 다른 점보다 굵게 표시된 점이 바로 특징 점이다. 이러한 특징 점들은 일반적으로 0, 1 등은 존재하지 않는다. 나머지 숫자들은 존재할 수도 있고 그렇지 않을 수도 있다. 실험의 용이성을 위해 특징 점을 최소화하였다.

<그림 14>에서는 정렬 과정과 특징 점을 추출하는 과정을 보여주고 있다.



<그림 14> 정렬과 특징 점 추출

<Figure 14> Sorting and extract of feature point

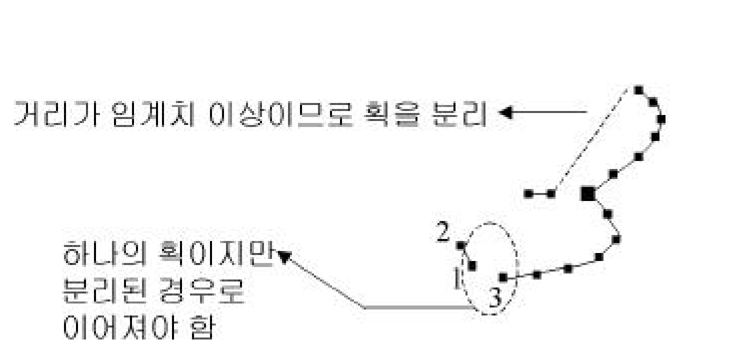
<그림 14>에서 특징 점으로 표시된 부분은 선택될 수 있는 점이 원으로 표시한 곳에 3개의 점 모두가 가능하다. 원 이미지에서 픽셀이 모두 존재하며 거리도 모두 임계치를 벗어나지 않으나 이전 점과의 기울기를 고려한다면 우측의 두 개 점이 가능하다. 우측 두 개의 점이 세 개의 조건인 원 이미지의 픽셀 존재 여부, 거리, 기울기를 모두 만족하기 때문에 이때는 가장 거리가 가까운 점을 선택하여 이전의 과정을 반복하게 된다. 이러한 과정을 반복하게 되면 <그림 14>의 세선화 이미지는 내부적으로 <그림 15>와 같은 정렬 순서를 가지게 된다. <그림 14>의 경우

특징 점이 존재하여 특징 점을 기준으로 획을 분리하여 각각을 구성할 수 있으나 하나의 획으로 이어서 표현할 수 있으므로 강제로 분리하지 않고 <그림 15>와 같이 하나의 이어진 획으로 생성한다. 하나의 이어진 획으로 구성하기 힘든 경우에만 특징 점을 기준으로 분리를 하여 여러 개의 획으로 구성한다.

<그림 15> 세선화 이미지의 정렬 순서

<Figure 15> Sorting order of thinning image

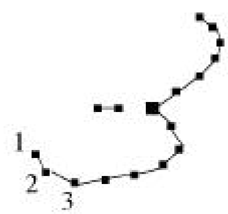
세 번째 ③의 과정은 획의 병합인데 이 과정이 필요한 이유는 <그림 16>과 같이 획 분리와 특징 점을 추출한 후에 획의 분포에 있어 시작점이 획의 끝이 아닌 경우가 발생하기 때문이다. 이는 하나로 이어져야 하는 획이 불필요하게 분리된 경우이다. <그림 16>에서 표시된 번호는 정렬의 순서를 의미하며 1로 표시된 점이 시작점이다. 획을 병합하는 방법은 각 획의 처음과 끝점을 각각 세 가지 조건을 만족하는지를 검사하여 그 조건을 만족하는지의 여부로 병합을 결정한다.



<그림 16> 획 분리와 특징 점을 추출한 후 획의 분포

<Figure 16> Distribution of strokes after separate stroke and extract feature point

<그림 16>에서는 획 분리와 특징 점을 추출한 후의 획의 분포를 알아보기 쉽도록 선으로 연결하여 표현하였다. 이 그림에서 3이라는 숫자가 모두 세 개의 획으로 구성되어 있음을 알 수 있는데 각 획들의 시작점과 끝점 각각의 관계를 조사하면 원으로 표시된 부분이 거리와 기울기 그리고 원 이미지의 숫자 존재 여부 모두를 만족하므로 하나의 획으로 이어지는 것이 바람직하다는 것을 알 수 있다. 획을 병합한 후의 분포는 <그림 17>과 같이 짧은 획의 정렬 순서는 반대로 바뀌게 되어 15개의 점이 하나의 획으로 구성되었다.

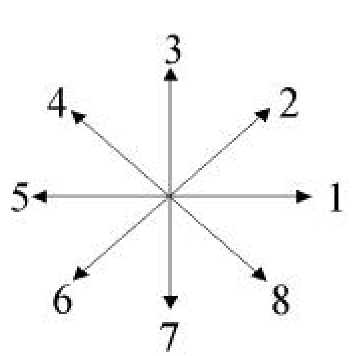


<그림 17> 병합한 후의 획 분포와 정렬 순서

<Figure 17> Distribution of strokes and sorting order after merging

3.2.2 방향 코드의 추출과 공간 정보 저장

3.1의 각 단계를 거친 후 마지막으로 ④의 과정은 각 점들간의 방향 코드를 추출하는 것이다. 세선화 된 이미지를 8개의 절대 방향을 이용하여 생성한다. 8방향 코드는 <그림 18>과 같은 2차원 공간상의 8개의 절대 방향을 의미한다[14,27].



<그림 18> 8방향 코드

<Figure 18> Codes of eight directions

<그림 17>의 숫자 3을 예로 들어 각 획의 방향 코드를 추출하면 두 개의 획이 생성되어 첫 번째 획은 8 8 1 1 1 2 3 4 2 2 2 3 3 4, 두 번째 획은 5가 생성이 된다. 여기서 동일한 코드들이 반복적으로 생성된 것을 알 수 있는데 동일한 코드가 반복적으로 나오는 것은 동일 방향으로 획이 계속 이어짐을 의미한다. 따라서 획 인식기의 입력을 최소화시키기 위하여 이들 동일 코드들은 하나의 코드로 축소가 가능하다. 생성된 방향 코드에서 동일 코드를 축소시키게 되면 최종적으로 첫 번째 획은 8 1 2 3 4 2 3 4, 두 번째 획은 5가 생성이 된다. 반복되는 코드들을 제거하여 코드를 생성한 결과 하나의 획을 구성하는데 최대 12개의 코드만이 필요하였다.

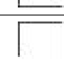
이렇게 생성된 방향 코드는 인식기의 입력으로 들어가게 되는데 분리된 획들의 공간 정보(위치) 또한 중요한 특징으로 사용되기 때문에 공간 정보를 저장해 둔다. 공간 정보는 <그림 17>과 같이 하나의 숫자 이미지를 획으로 생성한 최종 결과가 두 개 이상의 획으로 분리된 경우에는 각 획들이 어느 위치에 존재하며 다른 획과 어떻게 연관되어 있는지가 아주 중요하게 된다. 획이 분리가 되는 경우에는 특징 점을 기준으로 분리가 되므로 특징 점을 포함하는 획의 특징 점 부분과 그 외 획의 점들의 관계를 공간 정보로서 활용할 수 있는데 <그림 17>을 예로 들면 특징 점을 포함하는 획에서 특징 점의 좌측에 하나의 획이 붙어 있는 경우로 판단할 수 있다. 따라서 이러한 공간 정보를 저장해 두었다가 획을 인식한 후 최종적으로 획의 조합으로 숫자를 인식할 때 정보로 사용하게 된다.

획의 분리와 방향 코드를 추출한 결과 <표 1>과 같이 stroke 0부터

stroke 14까지 총 15개의 획으로 집약되었으며 각 획들이 의미하는 획의 모양을 설명하였다. 각 획들은 각각 획에서 자체적으로 하나의 숫자를 구성하는 것(stroke 0 6)과 하나의 획으로는 숫자를 구성하지 못하고 두 개 이상의 획이 함께 존재하여 하나의 숫자를 구성하는 것(stroke 7 14)으로 구분할 수 있다. 예를 들면 숫자 4는 stroke 8과 stroke 13이 조합되어 구성되며, stroke 7과 stroke 11이 조합되어 숫자 2를 구성하는 경우이다.

<표 1> 각 획의 모양

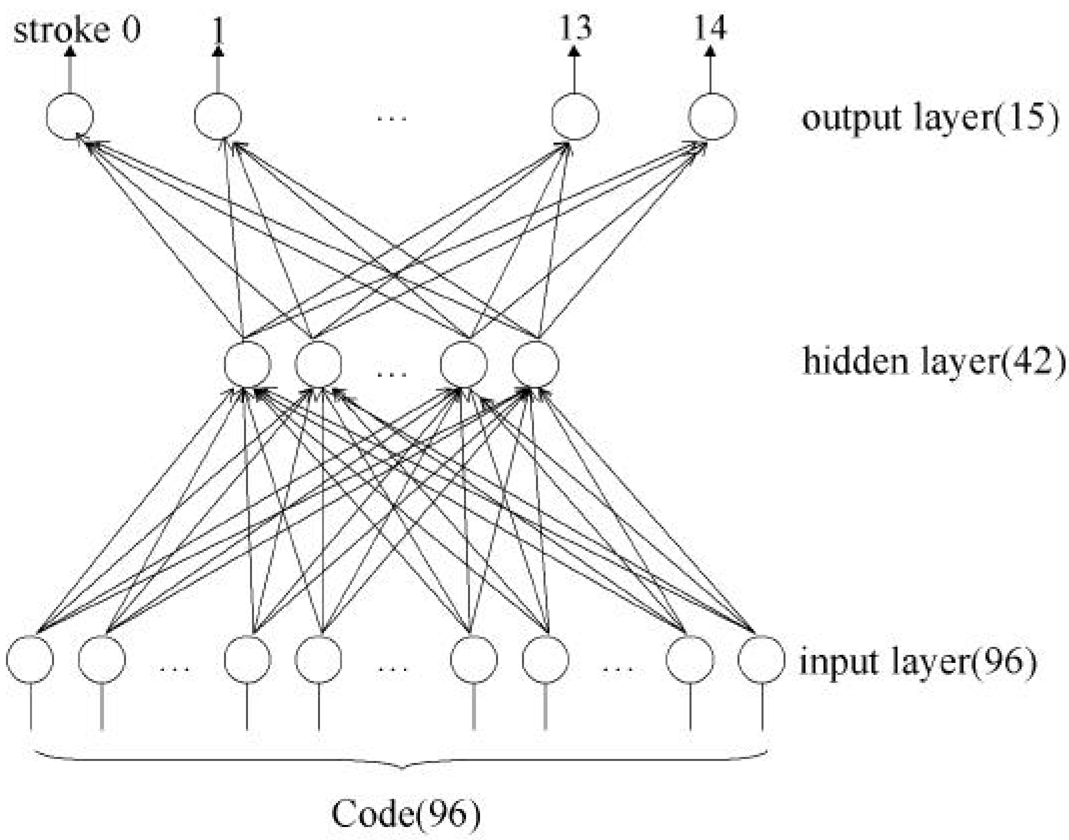
<Table 1> Form of each strokes

획	모양
stroke 0	0
stroke 1	3
stroke 2	4
stroke 3	5
stroke 4	6
stroke 5	7
stroke 6	9
stroke 7	-
stroke 8	|
stroke 9	\
stroke 10	/
stroke 11	
stroke 12	
stroke 13	
stroke 14	

3.2.3 획 인식

3.1부터 3.2의 각 단계를 모두 거치게 되면 각 획들의 방향 코드와 함께 공간 정보가 생성이 되었다. 이렇게 생성되면 생성된 방향 코드를 입력으로 획을 인식하는 과정을 거친다.

인식기는 은닉층을 갖는 3층 퍼셉트론 구조로 구성하였다. <그림 19>에서 획 인식기의 구성도를 나타내었다.



<그림 19> 획 인식기의 구성도

<Figure 19> Diagram of stroke recognition

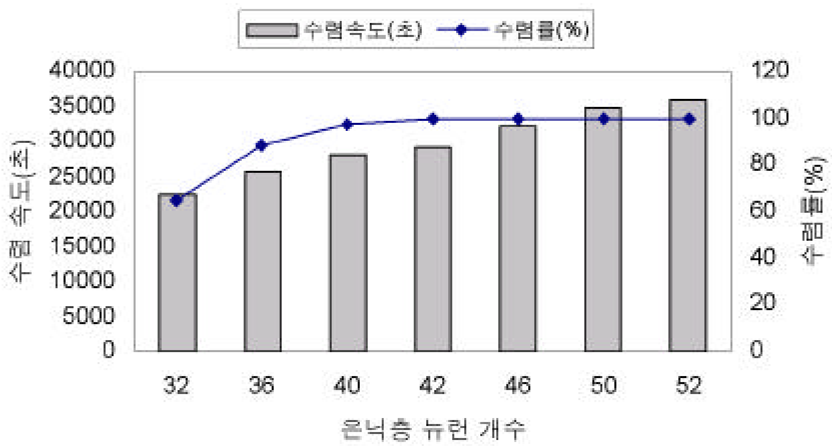
획 인식기의 입력은 전처리 과정에서 추출한 각 획의 12개 방향 코드를 사용하며, 각 코드당 8개의 방향이 존재하고 방향이 없는 것은 0으로 두었기 때문에 8개의 방향과 무 방향을 표현함에 있어 총 8개의 뉴런이 필요하다. 이들 방향 코드의 입력 값은 <표 2>에 나타내었다. 따라서 입력 뉴런의 개수는 8×12로 총 96개의 뉴런이 필요하며 출력은 stroke 0부터 stroke 14까지의 15개의 획 표현을 위해 15개, 은닉층은 <그림 20>과 같이 실험적으로 42개로 결정하였다.

<표 2> 방향 코드의 입력 값

<Table 2> Input value of direction codes

방향 코드	입력 값
0	00000000
1	10000000
2	01000000
3	00100000
4	00010000
5	00001000
6	00000100
7	00000010
8	00000001

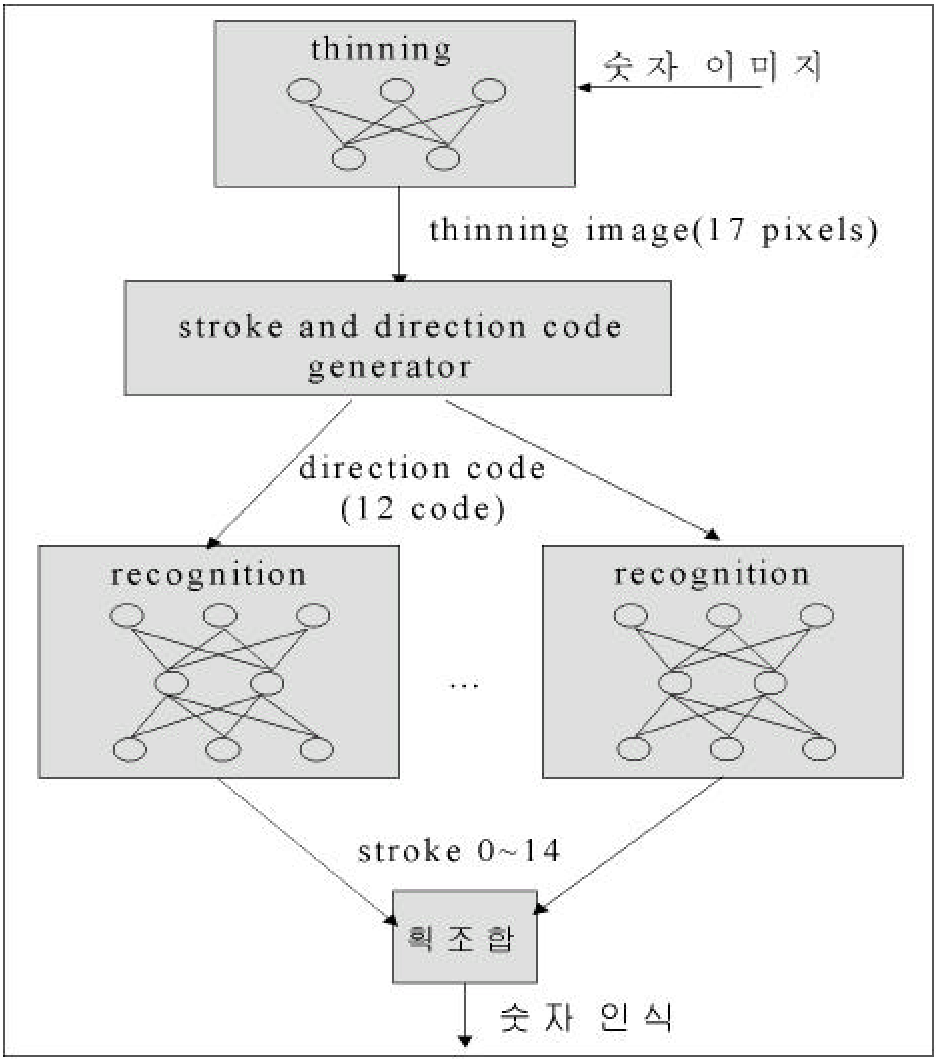
다음 <그림 20>과 같이 학습시 은닉층의 뉴런 개수가 42개 이상이 되면 100% 수렴하게 되고, 수렴 속도는 뉴런의 수가 늘어날수록 점점 느려짐을 알 수 있다. 수렴률이 100%에 도달하는 시점의 최소 뉴런의 개수가 42개이므로 42개의 은닉층의 뉴런이 가장 적절하다.



<그림 20> 은닉층 뉴런 수에 따른 수렴 속도

<Figure 20> Extortion speed by number of hidden neuron

지금까지의 모든 과정을 거친 후에 최종적으로 획들의 조합과 공간 정보를 이용하여 숫자를 인식하게 된다. 지금까지의 모든 과정을 다음 <그림 21>에 나타내었다. 전체 과정을 간단히 요약하면 숫자 이미지를 입력으로 받아 SOG* 세선화로 세선화를 수행하여 그 결과로 나온 17개의 점으로 구성된 세선화 이미지로 점들간의 관계로서 특징 점과 획을 추출하여 각 획별로 8방향 코드와 공간 정보를 생성한다. 추출된 각 획들의 방향 코드는 획 인식기에 입력으로 입력되어 각 획을 인식하여 그 결과로 나온 획들의 조합과 공간 정보로 최종 숫자를 인식하게 된다.



<그림 21> 숫자 인식을 위한 전체 시스템의 흐름도

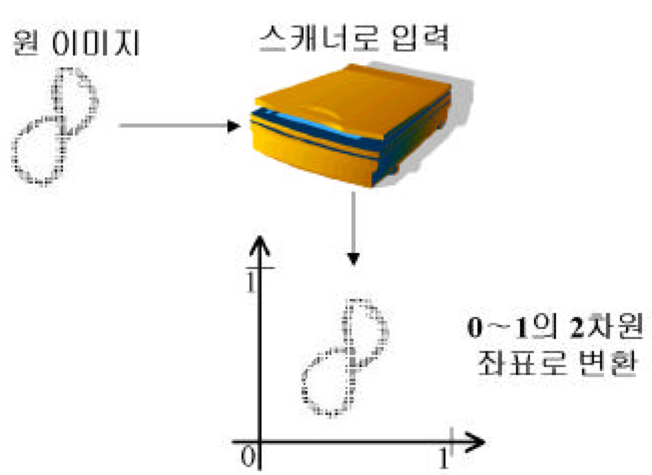
<Figure 21> System diagram of digit recognition

4. 구현 및 실험

이번 장에서는 방향 코드 조합에 의한 오프라인 필기체 숫자 인식을 위한 시스템 구현 및 실험에 관해 설명하고, 실험을 통하여 기존의 오프라인 숫자 인식 시스템과의 특징 추출 기법과 인식률에 대해 비교하여 보여준다.

4.1 오프라인 필기체 숫자 인식 시스템의 구현

오프라인 필기체 숫자 인식 시스템의 개발 소프트웨어는 Visual C++ 6.0을 사용하여 구현하였으며, P-Ⅲ 1GHz의 시스템에서 <그림 22>와 같이 원 숫자 이미지를 스캐너를 통하여 입력받아 0 1 사이의 2차원 좌표로 변환하여 데이터를 입력받아 이후 처리한다.



<그림 22> 데이터 입력 과정

<Figure 22> Process for data of input

4.2 오프라인 필기체 숫자 인식의 실험

본 실험에 사용된 데이터는 Concordia University, KAIST, 연세대학교의 필기체 숫자 데이터베이스이다. Concordia University의 데이터베이스는 우편번호에서 추출한 0부터 9까지 숫자 600자씩 총 6,000자로 구성되어 있고, KAIST 데이터베이스는 23명의 필기자가 쓴 5,786자이고, 연세대학교 데이터베이스는 세금계산서에 쓰여진 41,312자로 구성되어 있다.

본 논문의 실험은 다른 실험과 달리 데이터베이스의 일부를 학습하고 일부를 테스트하는 방식이 아니라, 숫자 이미지의 특성을 미리 파악하여 획을 추출하고 추출된 획들을 방향 코드로 변환하여 이를 획 인식기의 학습에 사용하여 인식된 획들의 조합으로 최종 숫자를 인식하는 방법을 사용하였다. 따라서 숫자 이미지 데이터베이스 전체를 인식에 사용할 수 있었다.

학습 단계에서 각 획별로 특성을 파악하여 미리 방향 코드를 작성하여 학습을 하였는데, 이때 유사 패턴으로 인한 인식의 저하가 발생하게 된다. 이런 경우에는 그 패턴에 대한 가중치를 더 높게 부여하여 학습을 시킨다. 예를 들어 1의 패턴 중에 방향 코드가 3인 경우는 숫자 7의 방향 코드 3 5와 유사하여 인식의 임계 값을 넘지 못하는 경우가 발생하는데 이런 현상을 없애기 위해 숫자 1의 패턴에 대해 가중치를 높게 부여하면 숫자 1과 7이 모두 임계 값을 넘어 정확한 인식을 하였다.

학습 단계의 신경회로망 구조에서는 최대 학습 회수를 20,000번으로 제한하고, 임계 값은 0.9, 학습률은 실험적으로 0.02로 부여하였고, 인식

단계에서는 역시 임계 값을 0.9로 부여하여 실험하였다.

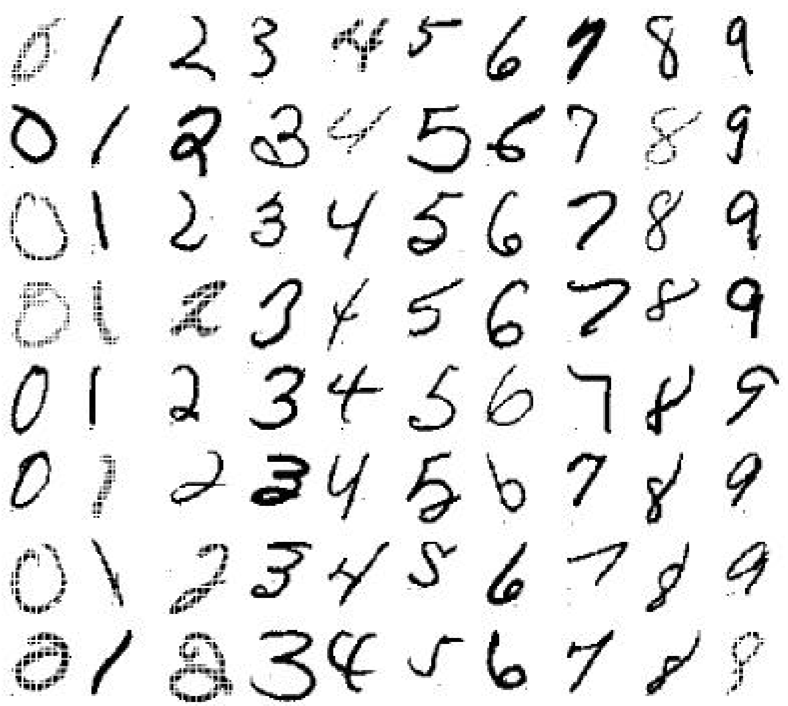
다음 <표 3>은 각 획별 패턴의 개수이다. 패턴의 수가 적은 획에서 패턴이 많은 획과 유사 패턴이 발생할 경우에는 앞에서 설명한 숫자 1의 코드 3처럼 가중치를 높게 부여한다.

<표 3> 각 획별 패턴의 수

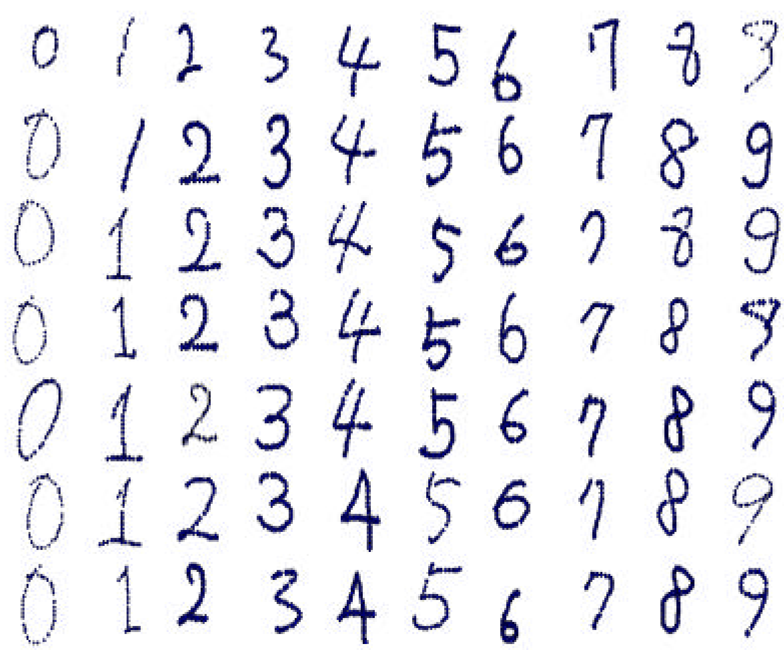
<Table 3> Pattern number of per strokes

획	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
개수	79	45	19	71	29	7	49	21	16	5	4	78	21	18	21

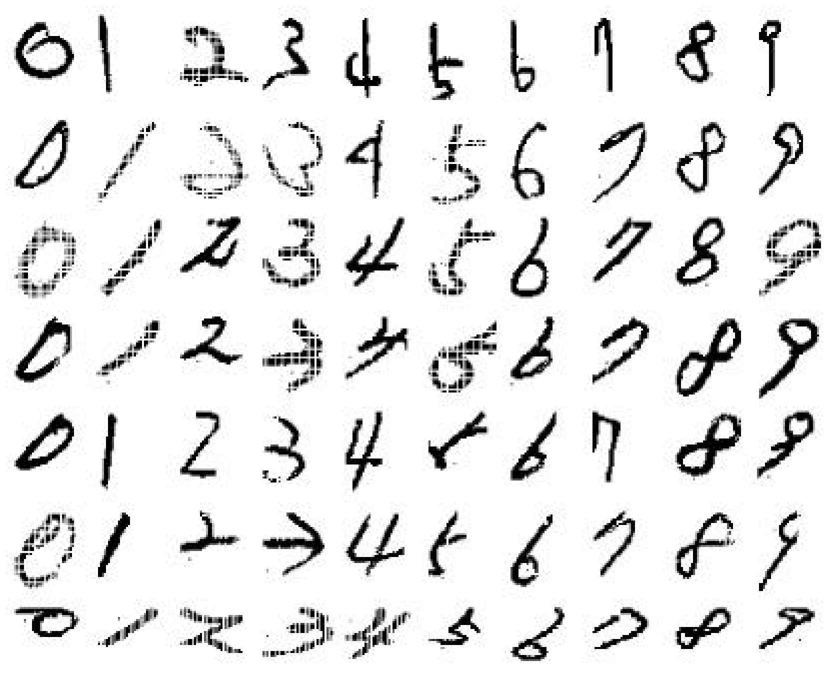
위의 <표 3>의 각 패턴들은 획의 특징을 고려해 만든 것들이며, 단순한 세로획(stroke 8)의 경우에도 패턴이 많은 이유는 필기체 숫자의 특성상 필기자의 필기 습관에 의한 변형이 발생하기 때문이다. 가장 기본적인 세로 획의 코드는 3 또는 7이지만 필기된 숫자의 기울어진 정도에 따라 2 6이 나올 수도 있고, 또한 숫자를 곧 바로 쓰지 않고 휘어서 필기한다면 단순하게 두 개의 코드 값만으로는 표현할 수 없기 때문이다. 다른 획의 경우에도 동일하다. 본 논문의 실험에 사용된 필기체 숫자 데이터베이스들은 다음 <그림 23>과 같다.



(a) Concordia University **숫자 데이터베이스**



(b) KAIST **숫자 데이터베이스**



(c) 연세대학교 숫자 데이터베이스

<그림 23> 실험에 사용된 숫자 데이터베이스

<Figure 23> Numeric database using experimentation

4.3 오프라인 필기체 숫자 인식의 실험 및 실험 결과

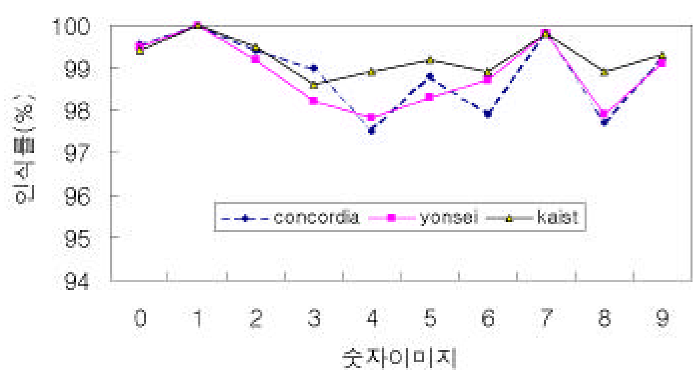
<그림 23>에 나타낸 세 개의 필기체 숫자 데이터베이스에 대해 실험한 결과는 다음 <표 4>와 같은 인식률을 얻었다.

<표 4> 전체 인식률

<Table 4> Recognition rate of total

숫자 DB	정인식	오인식	미인식
Concordia	98.89%	0.40%	0.71%
KAIST	99.25%	0.20%	0.55%
Yonsei	98.85%	0.45%	0.70%

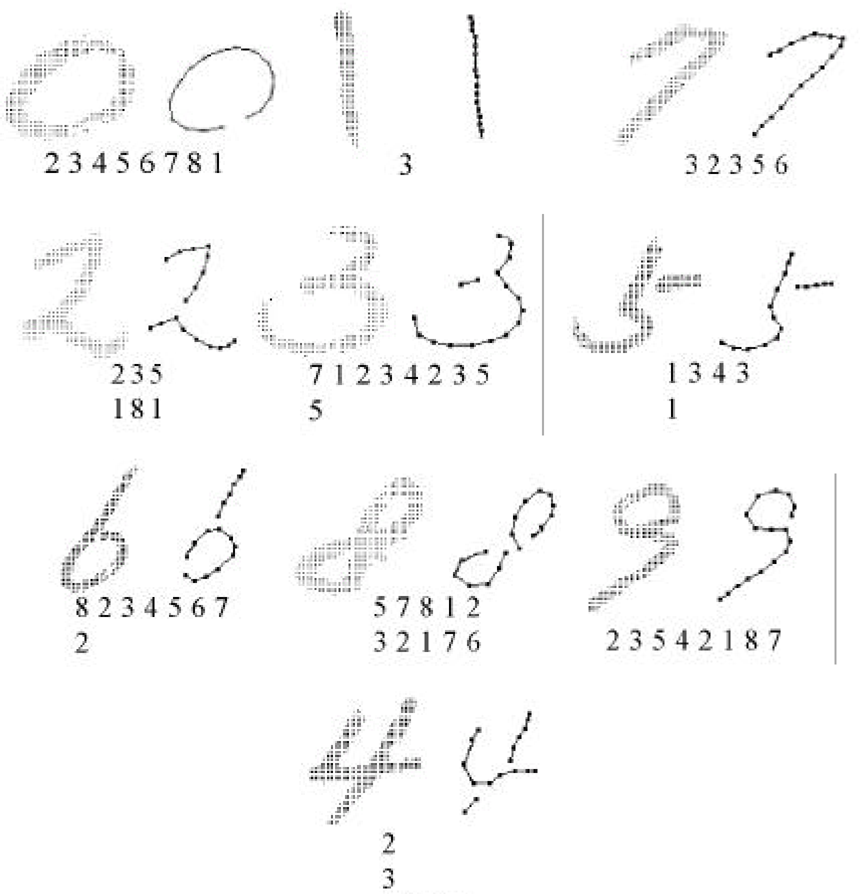
각 숫자별로 인식률을 측정한 결과는 <그림 24>에서처럼 단순한 1과 7은 다른 숫자에 비해 아주 높은 인식률을 갖는 반면 4, 8 등의 숫자는 다른 숫자에 비해 다소 낮게 나타난다. 그 이유는 1, 7 등의 숫자는 획이 단순하고 패턴의 개수도 작으며 다른 숫자들과의 패턴이 독립적이지만 다른 숫자들은 그 획의 패턴 수가 1, 7 등의 숫자보다 많으며 유사한 패턴이 일부 존재하는 이유이다.



<그림 24> 각 숫자별 인식률

<Figure 24> Recognition rate of each number

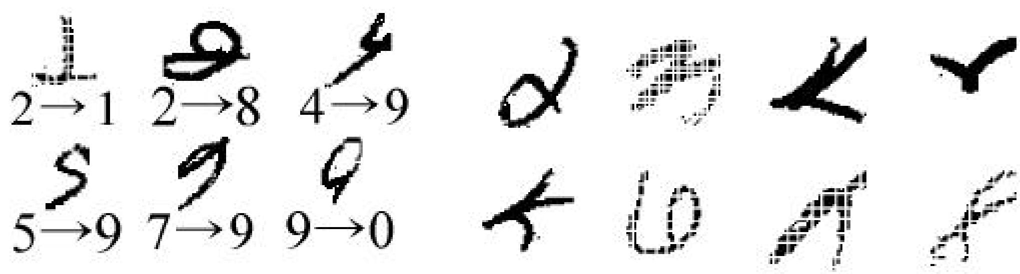
<그림 25>에는 각 숫자별로 원 이미지와 획을 분리하여 방향 코드를 생성한 최종 결과를 보여주고 있다. 0, 1, 7, 9와 같은 숫자들은 하나의 획으로 각 숫자를 표현할 수 있으며 나머지 숫자들은 둘 이상의 획의 조합으로 각각 표현할 수 있었고, 각 획의 분리가 정확히 이루어 졌음을 알 수 있다.



<그림 25> 획 분리와 방향코드 생성후의 결과

<Figure 25> Result of after divide strokes and create direction codes

다음 <그림 26>에는 본 논문의 실험 결과 오인식과 미인식된 숫자 이미지들을 나타내었는데, 오인식된 숫자 이미지의 대부분은 사람이 판단하기에도 어려웠고 그 나머지는 방향 코드의 유사성 때문에 오인식이 발생하였다. 미인식된 숫자 이미지들도 비슷한 이유에서 발생하였다.



(a) 오인식된 이미지　　　(b) 미인식된 이미지

<그림 26> 오인식과 미인식된 숫자 이미지들

<Figure 26> Numeric Images of error and reject.

그 결과의 검증을 위해 타 논문의 인식률과 전처리 과정의 특징을 비교해 보면 <표 5>와 같다. <표 5>의 결과에서 보듯이 본 논문은 단순한 방향 코드라는 특징만을 사용하여 타 논문과 유사한 높은 인식률을 얻었다. 이로써 이 단순한 방향 코드 특징이 각 숫자의 특성을 잘 반영하고 있음을 의미한다. 또한 타 논문들이 특정 데이터베이스에 대해 일부를 학습하고 나머지를 테스트하는 방식을 사용하였지만 [3,4,17,20,24,28], 본 논문은 숫자의 획 특성을 파악하여 그 특징을 미리 생성한 후 학습을 시키는 방식을 사용하였다. 따라서 <표 4>에 나타낸 인식률이 다른 데이터베이스에서도 거의 유사하게 나올 것으로 기대한다.

<표 5> 논문별 인식률과 사용된 특징 비교

<Table 5> The treatise awareness rate and used characteristic comparison

참고문헌	숫자 DB	인식률(%)	사용된 특징
[3]	Concordia Univ.	97.80	국부특징(방향정보), 전역특징(영상압축)
	ETR	99.40	
[4]	Concordia Univ.	97.10	윤곽선의 4방향, 교차거리, 교차, 망, 투영
	Dong-A Univ.	98.00	
[5]	NIST	97.48	선형형태변환(세선화, 기울기 교정, 덧씌우기), 비선형형태변환(점밀도, 교차회수, 획간간격)
[19]	Concordia Univ.	98.00	원 이미지, 회전 이미지, 이동 이미지
[21]	Concordia Univ.	94.45	유전자 알고리즘+신경망→가중치 조절
[23]	Concordia Univ.	96.61	다수인식기의 확률적 결합
[24]	Concordia Univ.	98.20	망, 교차, 투영, 교차거리, 특정궤적
[25]	자체구성	98.00	망, 교차, 투영, 교차거리, 특정궤적, 방향벡터
[28]	Concordia Univ.	95.00	특징 벡터 값, 숫자 영상의 직선길이, 숫자 영상의 직선 방향 값
[29]	NIST	99.00	기울기, UDLRH 오목성, 매쉬, Haar
[30]	자체구성	94.00	외곽선 추출, 세선화(또는 골격선)
본 논문	Concordia Univ.	98.89	세선화를 통한 획의 방향 코드
	KAIST	99.25	
	Yonsei Univ.	98.85	

5. 결론 및 향후 연구과제

본 논문은 오프라인 필기체 숫자를 인식함에 있어 방향 코드만으로 획의 특징을 나타내어 그 획의 조합으로 숫자를 인식하여 인식률을 높이고자 하였다.

전처리 과정에서는 속도가 빠르고 그 결과가 우수한 세선화 방법인 SOG* 세선화 기법을 이용하였고, 이렇게 세선화된 이미지를 이용하여 획으로 분리하며, 분리된 획을 8방향 코드로 생성한 후 반복 코드를 제거하여 획 인식기의 입력으로 들어가게 된다. 획 인식기는 은닉층을 갖는 backpropagation을 이용하였고, 학습에 사용된 패턴들은 필기체 숫자 이미지의 특징을 미리 파악하여 방향 코드를 미리 생성하여 학습을 한 후에 숫자 이미지 데이터베이스 전체에 대해 인식 테스트를 실행하였다. 생성된 획과 공간 정보를 이용하여 최종적으로 숫자를 인식하는 방법을 사용하였다.

기본 논문이 인식률을 높이기 위하여 많은 특징들을 사용하고, 또한 특정 숫자 데이터베이스에 대해서 일부를 학습하고 일부를 테스트하는 방식을 사용한데 비해 단순한 특징만을 사용하여 전처리를 단순화 시켰으며 패턴을 미리 생성하여 학습에 사용하였기 때문에 전체 데이터베이스에 대해 인식 테스트를 할 수 있었다.

본 연구에서는 오프라인 필기체 문자 인식의 기본이 되는 오프라인 필기체 숫자 인식을 위한 시스템을 구현하였다. 본 논문의 실험 결과 기존의 방법보다 전처리에 소요되는 노력과 시간을 감소할 수 있었다. 이를 바탕으로 향후 연구과제로 오프라인 필기체 문자 인식 시스템을 구현하

는 것이 필요할 것이다. 또한 오프라인뿐만 아니라 온라인 상의 필기체와 인쇄체에 대한 문자 인식이 연구과제로 남아 있다.

참고문헌

[1] Keiji Yamada, "Inverse Recall Neural Network Model and Feedback Pattern Recognition", IEEE International Conference on Neural Networks, pp. 399-406, 1993.

[2] Lianwen JIN, Kwokping CHAN and Bingzheng XU, "Off-line Chinese Handwriting Recognition using Multi-Stage Neural Network Architecture", IEEE International Conference on Neural Networks, pp. 3083-3088, 1995.

[3] 김영준, 이성환, “유전자 알고리즘과 결합된 다층 클러스터 신경망을 이용한 무제약 필기체 숫자의 오프라인 인식”, 한국정보처리학회논문지, 제 21 권 제 8 호, pp. 1468-1479, 1994

[4] 박창순, 김두영, “오프라인 필기체 숫자 인식을 위한 다양한 특징들의 성능 비교 및 인식률 개선 방안”, 한국정보처리학회논문지, 제 3 권 제 4 호, pp. 915-925, 1996.

[5] 방극준, 조남신, 강창언, 홍대식, “필기체 숫자 인식을 위한 병렬 자구성 계층 신경회로망”, 전자공학회논문지, 제 33 권 B편 제 7 호, pp. 1193-1202, 1996.

[6] 이성환, “패턴 인식의 원리 II", 홍릉과학출판사, 1997.

[7] 이성환, “문자 인식 이론과 실제 II", 홍릉과학출판사, 1997.

[8] Holt. C. M., A. Stewart, "A Parallel Thinning Algorithm with Fine

Grain Subtasking", Parallel Comput. 10, pp. 329-334, 1989.

[9] L. Lam., S. Lee and C. Y. Suen, "Thinning Methodologics - a Comprehensive Survey", IEEE Transaction Pattern Analysis and Machine Intelligence, 14(9), pp. 869-885, 1992.

[10] P. Ahmed, "A Neural Network Based Dedicated Thinning Method", Pattern Recognition, 16, pp. 585-590, 1995.

[11] M. Altuwaijri, M. Bayoumi, "A New Thinning Algorithm for Arabic Characters Using Self-Organizing Network", Circuits and Systems, ISCAS '95, IEEE International Symposium on Vol. 3, pp. 1824-1827, 1995.

[12] Jhon Cowell, Fiaz Hussain, "Thinning Arabic Characters for Feature Extraction", IEEE Information Visualization, Proceedings, Fifth International Conference, pp. 181-185, 2001.

[13] Majid M. Altuwaijri, Magdy A. Bayoumi, "A Thinning Algorithm for Arabic Characters Using ART2 Neural Network", IEEE Transactions on Circuits and Systems-II, Vol. 45, No. 2, 1998.

[14] **이찬희**, **정순호**, "**개선된** SOG **기반 고속 세선화 알고리즘**(SOG*)", **한국정보처리학회논문지**, **제** 8-B **권 제** 6 **호**, pp. 651-656, 2001.

[15] J. E. Dayhoff, "Neural Netwok Architectures", Van Nostrand Reinhold, 1990.

[16] S. Haykin, "Neural Networks", Prentice Hall, 1994.

[17] Y. P. Jun, H. S. Yoon, J. W. Cho, "L* Learning : A Fast Self-Organizing Feature Map Learning Algorithm Based on

Incremental Ordering", IEICE Transaction, INF & SYST., Vol. E76-D, No. 6, June, 1993.
[18] I. Gyoun, P. Albrecht, Y. Le Cun, J. Denker, W. Hubbard, "Design of a Neural Network Character Recognizer for a Touch Terminal", Pattern Recognition, Vol. 24, No. 2, pp. 105-119, 1991.
[19] 임길택, 남윤석, 진성일, "회전 및 이동 영상을 이용하는 모듈 구조 신경망 기반 필기체 숫자 인식", 한국정보처리학회논문지, 제 7 권 제 6 호, pp. 1834-1843, 2000.
[20] 김백섭, 송혜정, "필기체 문자인식을 위한 체인코드에 기반한 효율적인 전처리 및 특징 추출", 한국정보과학회논문지(B), 제 25 권 제 12 호, pp. 1788-1796, 1998.
[21] 김현돈, 조성배, "유전자 알고리즘을 사용한 구조 적응 자기구성 지도의 최적화", 퍼지 및 지능시스템학회논문지, Vol. 11, No. 3, pp. 223-230, 2001.
[22] 김종석, 홍연찬, "클러스터 신경망을 이용한 우편번호 인식 시스템의 설계", 퍼지 및 지능시스템학회논문지, Vol. 11, No. 2, pp. 132-140, 2001.
[23] 강희중, 이성환, "무제약 필기 숫자를 인식하기 위한 다수 인식기를 결합하는 의존관계 기반의 프레임워크", 한국정보과학회논문지, 제 27 권 제 8 호, pp. 855-863, 2000.
[24] 류강수, 김우태, 진성일, "다중 특징과 모듈화된 신경회로망을 이용한 인쇄체 및 필기체 혼용 숫자 인식", 전자공학회논문지, 제 32 권 B 편 제 10 호, pp. 1347-1357, 1995.

[25] 류강수, 진성일, "모듈화된 신경회로망 중간층 출력의 재학습에 의한 필기체 숫자 인식", 한국정보과학회논문지(B), 제 23 권 제 9 호, pp. 931-640, 1996.

[26] 이찬희, 이상훈, 장수미, 정순호, "획의 방향 코드 조합에 의한 오프라인 필기체 숫자 인식", 한국정보과학회 2002 춘계 학술 발표논문집, 하권, pp.610-612, 2002.

[27] H. Freeman, "On the Encoding of Arbitrary Geometric Configuration", IEEE Transactions on Electronic Computers, Vol. EC-10, pp. 260-268, June 1961.

[28] 박주양, "문서 영상 이진화와 필기체 숫자 인식", 석사학위논문, KAIST, 1996.

[29] 윤종민, 신요안, 차형태, 정규식, "신경망 가지치기에 근거한 특징 선택의 필기체 문자 인식에의 응용", 한국정보과학회논문지(B), 제 24 권 제 10 호, pp. 1043-1052, 1997.

[30] Travieso, C. M., Morales, C. R., Alonso, I. G., Ferrer, M. A., "Handwritten Digits Parameterisation for HMM based Recognition", Image Processing and Its Applications, Seventh International Conference on (Conf. Publ. No. 465), Vol. 2, pp. 770-774, 1999.

감사의 글

한참 무더위로 지친 속에 2년이라는 길지도 짧지도 않은 대학원 생활을 돌이켜보며 지금 학위논문의 마지막에 이 글을 쓰고 있습니다. 한 권의 논문을 완성하기 위해 도와주셨던 고마운 분들의 마음을 글로써 간직하고픈 마음을 가집니다.

많이 부족한 저를 학문과 더불어 마음 자세까지 늘 지켜봐 주시고 충고와 칭찬의 말씀을 아끼지 않으신 정순호 지도교수님께 깊은 감사를 드립니다. 그리고 여러모로 바쁘신 중에도 이 논문을 심사해 주신 김영봉 교수님과 박지환 교수님께도 깊이 감사 드립니다. 또한 학부 때부터 대학원까지 학문과 인격을 지도해주신 박만곤 교수님, 윤성대 교수님, 여정모 교수님, 박승섭 교수님, 김창수 교수님, 박흥복 교수님, 이경현 교수님께도 깊이 감사 드립니다.

같이 살면서 충고와 격려를 해준 사랑스런 후배 형구와 학문적으로 많은 도움을 준 수길, 현기, 지웅 선배, 대학원 생활을 잘 할 수 있도록 도와준 김미은 조교에게도 감사의 말을 전하고 싶습니다. 같은 연구실에서 생활하면서도 도움을 주지 못해 늘 미안한 상수와 뒤에서 많은 도움을 주시는 강석호 교수님께도 감사 드립니다. 그리고 한 학기 먼저 졸업해 학교와 직장에서 열심히 생활하는 대학원 동기들과 변함 없는 우정으로 격려를 해주는 영원한 벗 정성, 형일과 그 외 많은 선후배들에게 감사의 마음을 전합니다.

늘 바쁘다는 핑계로 자주 찾아뵙지 못하지만 언제나 저의 뒷바라지에 혼신을 기울이시는 부모님께 죄송하고 감사하다는 말씀을 드리고 이 글을 바칩니다.