트루타입의 합성 글립을 이용한 새로운 한글 폰트 생성법

정 근 호　김 은 희　최 재 영
숭실대학교 컴퓨터학부

A Generating Method of Hangul Fonts using Composite Glyph of TrueType

Geunho Jeong　Eunhwe Kim　Jaeyoung Choi
School of Computing, Soongsil University

요 약

한글 폰트는 조합형 폰트와 완성형 폰트로 구분된다. 조합형 폰트는 폰트를 제작하는 시간과 노력이 적게 필요하지만 폰트의 품질이 완성형 폰트보다 뒤떨어진다. 완성형 폰트는 조합형과 비교하여 우수한 품질을 가지지만 폰트 제작에 더 많은 시간과 노력을 요구한다. 특히 완성형 폰트는 폰트내의 중복된 자소들의 정보를 중복해서 저장하므로 폰트 저장에 필요한 공간이 더 많이 필요하다. 본 논문에서는 트루타입의 합성 글립(Composite Glyph)을 이용하여 중복된 자소를 최소화한 완성형 폰트를 구성하였다. 중복성을 최소화한 완성형 폰트는 기존 완성형 폰트와 유사한 고수준의 품질을 유지하면서 조합형 폰트와 유사하게 폰트 저장 공간의 크기를 절약할 수 있다.

1. 서론

[1]컴퓨터에서 문자를 처리하기 위한 기술이 연구되면서 각 나라의 사용 언어 환경에 맞는 코드 체계와 입력기 그리고 입력한 문자들을 스크린이나 프린터로 출력하기 위해 필요한 폰트(font)와 관련된 연구가 계속되고 있다. 특히 한글은 영문권의 알파벳에 비해 자소가 다양하고 글자의 모양이 복잡하다. 따라서 한글을 컴퓨터에서 입력, 처리, 출력하는 원리는 알파벳과 같으나, 실제 방법은 훨씬 어렵고 복잡하다.

한글 폰트는 서로 상반된 장단점을 가진 조합형과 완성형 한글 폰트로 구분할 수 있다. 조합형 한글 폰트는 한글을 초성, 중성, 종성으로 구분하고 각 자소별로 폰트를 만든 다음, 이것을 조합하여 완성된 폰트를 만드는 것으로, 대개 화면용으로는 200-300자, 24핀 프린터나 레이저 프린터용으로는 400-800자 정도면 볼만한 글자 모양을 얻을 수 있다. 이에 비해 완성형 한글 폰트는 한글 각 글자의 폰트를 완성된 형태로 따로 따로 만드는 방식이다. KS C 5601 완성형 표준 코드가 발표된 이후에는 KS에 정의된 한글 2350자만 지원하는 완성형 폰트 제작 방식이 정착되었다 [1, 2].

조합형 폰트를 완성형 폰트와 비교해 보면 폰트를 만드는데 필요한 시간과 노력, 그리고 폰트 저장에 필요한 공간은 절약되지만 글자의

1) 본 연구는 한국과학재단 핵심전문연구(과제번호: KOSEF 981-0928-151-1) 지원으로 수행되었음.

품질은 완성형보다 뒤떨어진다. 반대로 완성형 폰트는 글자의 품질은 우수하지만 폰트를 만드는데 소요되는 시간과 노력 그리고 저장 공간이 많이 필요하다 [1]. 특히 한글 완성형 폰트는 폰트의 중복성이란 단점이 있다. 폰트의 중복성이란 자소가 모여서 한 음절을 이루는 한글의 특성상 완성형 폰트 하나의 음절을 도안하고 저장하면서 불필요하게 동일한 자소를 다시 디자인하고 이를 중복하여 저장하는 것을 말한다. 이러한 폰트의 중복성은 완성형 폰트 저장에 많은 공간을 필요로 한다. 만약 이들 중복된 자소를 최소화하여 완성형 폰트를 구성할 수 있다면, 완성형 폰트의 품질을 최대한 유지하면서 조합형 폰트와 유사하게 폰트 생성에 필요한 시간과 노력 그리고 저장 공간을 모두 절약할 수 있다.

최근에는 문서 작성자가 사용한 폰트를 문서 수신자가 동일하게 사용할 수 있도록 문서에 폰트를 임베딩(embedding)하는 폰트 임베딩 기술이 인터넷 환경에서 쓰이고 있다 [3]. 영문의 경우는 폰트의 크기가 한글에 비해 매우 작기 때문에 폰트를 임베딩한 문서의 크기가 문제되지 않지만, 한글의 경우 수 Mbyte이상 되는 한글 폰트를 임베딩해야 하므로 문서의 크기가 문제될 수 있다. 그러므로 완성형 폰트의 품질을 최대한 유지하며 폰트 저장에 필요한 저장 공간을 줄일 수 있는 연구가 반드시 필요하다.

본 논문에서는 현재 마이크로소프트사의 Windows 98, Windows NT와 애플사의 Mac OS에서 지원하는 폰트 기술인 트루타입(TrueType)의 합성 글립(composite glyph)을 이용하여 폰트의 중복성을 최소화한 새로운 한글 폰트를 제시한다.

2. 트루타입 폰트

트루타입은 애플 컴퓨터사에 의해 개발된 디지털 폰트 기술로서 폰트 처리 방법상 윤곽선 폰트에 해당한다. 현재 사용하는 윤곽선 폰트로는 아도브사의 타입1(Type1), 마이크로소프트사의 트루타입(TrueType), URW사의 NimbusQ, 비트스트림(Bitstream)사의 스피도(Speedo), 그리고 이카루스(Ikarus)사의 IK 또는 DI 포맷 등이 있다. 이들 중 특히 트루타입은 Windows 98과 Windows NT, Mac OS, Solaris(version 2.6이상)와 같은 운영체제에서 지원하며 프린터 호환성과 문서 호환성 등의 특성이 있고 프린터와 스크린에서도 같은 글자꼴을 사용할 수 있다. 또한 글자꼴 데이터와 함께 힌팅 정보가 프로그램 형태로 저장되어 있기 때문에 폰트 제작자가 더 융통성 있게 폰트를 제작할 수 있다. 이러한 장점으로 인하여 최근에는 다른 윤곽선 폰트에 비해 트루타입 폰트가 많이 사용된다 [4, 5, 6].

트루타입은 트루타입 폰트와 트루타입 래스터라이저(Rasterizer)로 구성되어 있다. 트루타입 폰트 파일에는 저작권, 폰트 이름, licencing permission, 문자에 대한 기술 정보, 힌팅 명령어, 문자 집합(character set)과 매핑(mapping) 등의 정보가 있다. 이러한 정보는 테이블 형식으로 저장되어 있는데 트루타입 버전 1.66 폰트 파일은 최대 24개의 테이블로 구성되어 있으며, 이중 10개는 반드시 필요한 테이블들이고 나머지 14개 테이블은 선택적으로 필요한 테이블이다. 이들 테이블을 정리하면 표 1, 2와 같다 [7].

폰트에서 글립(glyph)이란 한 폰트 내의 한 자 한 자의 이미지(image)로 트루타입에서는 각 글립마다 고유한 인덱스를 부여한다. 반드시 필요한 테이블 중 cmap 테이블은 문자 코드와 글립 인덱스(glyph index)의 매핑(mapping) 정보를 유지하고 있는 테이블로 여러 부테이블을 가질 수 있다 [7].

glyf 테이블은 실제로 문자 정보가 들어가는 테이블로 각 글립에 대한 윤곽선 개수, x, y축 상의 lower left corner와 upper right corner의 좌표값, 힌팅 프로그램, 글립을 구성하는 좌표점들 그리고 각 좌표점들에 대한 플래그(flag)들로 구성되어 있다 [7]. 따라서 모든 글립에 대한 실제 정보를 유지하고 있으므로 glyf 테이블에 의해서 트루타입 폰트 저장 공간의 크기가 좌우된다.

트루타입 래스터라이저는 폰트 테이블에 기술되어 있는 폰트 정보를 읽고 비트맵을 생성하는 프로그램으로 운영체제나 프린터 제어 소프트웨어의 한 부분이다. 래스터라이저는 문자

표 1 트루타입 폰트 파일에 필요한 테이블

table name	table contents
cmap	character to glyph mapping
glyf	glyph data
head	font header
hhea	horizontal header
hmtx	horizontal metrics
loca	index to location
maxp	maximum profile
name	naming table
post	PostScript information
OS/2	OS/2 and Windows specific metrics

표 2 선택적으로 필요한 테이블

table name	table contents
cvt	Control Value Table
EBDT	Embedded bitmap data
EBLC	Embedded bitmap location data
EBSC	Embedded bitmap scaling data
fpgm	font program
gasp	grid-fitting and scan conversion procedure(grayscale)
hdmx	horizontal device metrics
kern	kerning
LTSH	linear threshold table
prep	CVT program
PCLT	PCL5
VDMX	Vertical device metrics table
vhea	Vertical device Metrics header
vmtx	Vertical Metrics

비트맵을 생성하기 위해서 먼저 트루타입 폰트 파일로부터 문자에 대한 윤곽선 기술 정보를 읽어서 요청된 크기와 디바이스 해상도에 맞게 문자의 윤곽선 기술 정보를 확장한다. 그리고 힌팅정보를 사용해서 윤곽선을 조정하고 조절된 윤곽선을 픽셀로 채우는 일을 한다 [3].

3. 합성 글립(Composite glyph)

트루타입에서의 글립은 윤곽선과 명령어들로 구성된 단순(simple) 글립과 자신이 참조하는 글립의 인덱스들과 명령어들로 구성된 합성(composite) 글립이 있다. 단순 글립과 합성 글립의 정보는 모두 glyf 테이블에 저장된다. 래스터라이저가 단순 글립과 합성 글립을 모두 인식할 수 있도록 하기 위하여 합성 글립은 윤곽선의 개수를 저장하는 numberOfContours라는 변수에 음수값을 저장하고 있다 [7].

그림 1은 Windows 98에서 제공하는 Bookman Old Style 폰트에서 문자 이름이 onehalf인 윈도우로 폰토그라퍼의 글자 윈도우를 사용하여 나타낸 것이다. 폰토그라퍼(Fontographer)는 미국의 Altsys사가 만든 전문 그래픽 편집 프로그램이다 [8]. 그림 2는 glyf 테이블 내의 onehalf에 대한 정보를 나타낸다. Bookman Old Style 폰트의 glyf 테이블 내용을 보기 위하여 마이크로소프트사가 제공하는 트루타입 폰트 파일 덤핑(dumping) 도구인 TTFDump v.1.60을 사용하였다 [9]. 그림 2에서 보는 바와 같이 합성 글립은 글자 형태를 만들어 내는데 필요한 명령어들과 윤곽선을 나타내는 좌표점들을 복제하는 것이 아니라, 단순히 참조하는 글립의 인덱스만 저장하므로 저장 공간을 줄일 수 있다. 그리고 실제로 글립에 대한 이미지가 만들어질 때는 Offset 값을 사용하여 위치 조정을 하고, flags에 설정된 값에 따라 회전과 확대 등이 실행된다. 폰트 디자이너 입장에서도 이미 디자인된 글립을 다시 재사용하여 새로운 글자를 만드므로 새로운 글립을 만드는 시간과 노력을 절약할 수 있다.

실제로 onehalf를 저장하려면 1(one superior)을 저장하는데 142 bytes, /(fraction)를 저장하는데 102 bytes, 2(two superior)를 저장하는데 240 bytes가 소요되므로 반복된 글립 테이블의 머리글(header) 부분만 제외시킨다고 생각해도 거의 482 bytes 정도가 필요하게 된다. 그러나 합성 글립을 이용하면 52 bytes 정도가 필요하므로 많은 저장 공간을 절약할 수 있다.

4. 한글 완성형 폰트에 합성 글립을 적용

영문 알파벳은 음소문자이며 각 문자가 독립된 모양을 가지므로 합성 글립을 적용한다는 것이 불가능하며 단지 특수 문자 제작에만 사용한다. 하지만 한글은 음소문자이면서 음절문자라는 특성을 가지므로 각 음소에 해당하는

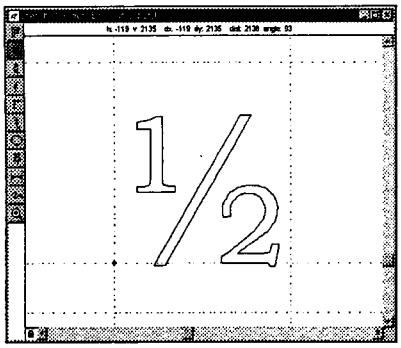


그림 1 Bookman Old style트루타입 폰트에서 'onehalf'

초·중·종성의 자소를 먼저 디자인하여 단순 글립으로 저장하고 각 자소를 합성하여 합성 글립으로 음절들을 만들 수 있다.

한글의 초·중·종성 글립을 가지고 있는 조합형 폰트의 경우 대체로 보기 좋은 글자꼴을 얻기 위해서는 초성 10벌, 중성 4벌, 종성 4벌 또는 초성 8벌, 중성 4벌, 종성 4벌 등으로 구성하는데, 각각 모두 382개나 344개의 자소를 디자인해야 한다. 실제 명조체는 900여 개의 자소가 있어야 이를 조합하여 제대로 서체를 표현할 수 있으므로 기존의 조합형 폰트의 품질이 크게 떨어진다는 것을 알 수 있다. 예를 들면 휴먼 명조체의 경우 초성의 'ㄱ'은 26벌, 'ㅏ'는 5벌이 필요하며, 전체 자소는 909개의 자소가 필요하다. 하지만 조합형의 특성상 각 초성 중성 종성에 대한 정해진 조합 틀을 가지고 있으므로, 벌수를 늘린다고 하더라도 조합 형태에 따라 자소의 모양도 변하는 한글을 제대로 표현하는 폰트를 만들기는 어렵다 [10].

따라서 완성형 폰트에서 중복된 자소들을 뽑아내고 그 자소를 단순 글립으로 구성한 후, 중복된 자소를 사용하는 글립이 단순 글립들을 참조하도록 완성형 폰트를 구성한다. 그러면 완성형 폰트의 품질을 그대로 유지하면서 폰트의 크기는 단순 글립을 저장하기 위한 기억장소만 소모하므로 기억장소는 조합형과 근사하게 된다.

5. 실험 결과

```
Glyph 244: off = 0x0000CC00, len = 52
  numberOfContours:-1  (Composite)
  xMin:                          243
  yMin:                          -25
  xMax:                          1776
  yMax:                          1419
    0: Flags:                    0x0026
         Glyf Index: 241
         X BOffset: 47
         Y BOffset: 0
         Other:                  Round X,Y to Grid

    1: Flags:                    0x0027
         Glyf Index: 188
         X WOffset: 809
         Y WOffset: 0
         Other:                  Round X,Y to Grid

    2: Flags:                    0x0107
         Glyf Index: 242
         X WOffset: 1096
         Y WOffset: -680
         Other:                  Round X,Y to Grid

  Instructions:
  -------------
  00000: NPUSHB       (9):     2   50   28    1
                               25    3    0    7   24
  00011: SVTCA[y-axis]
  00012: MIAP[rd+ci]
  00013: SHC[rp1,zp0]
  00014: MIAP[rd+ci]
  00015: SHC[rp1,zp0]
  00016: MIAP[rd+ci]
  00017: SHC[rp1,zp0]
```

그림 2 Bookman Old Style 폰트에서 합성 글립으로 만들어진 'onehalf'의 glyf 테이블

본 논문에서는 한글 완성형 폰트에 합성 글립을 적용하기 위하여 휴먼 중간 샘체 트루타입 폰트와 휴먼 명조체 트루타입 폰트를 사용하였다. 각 폰트들은 KS C 5601에서 정의된 한글 2350자가 단순 글립으로 저장되어 있다. 실험에서는 트루타입 폰트 파일의 내용을 분석하기 위해서 TTFDump v.1.60을 사용하였고, 합성 글립으로 폰트를 재구성하기 위해서 폰토그라퍼를 사용하였다.

먼저 폰토그라퍼의 폰트 윈도우와 glyf 테이블의 정보를 사용하여 단순 글립으로 구성되어 있는 각 폰트의 문자들을 초·중·종성별로 각 자소의 포인트 수와 성격(직선, 곡선)에 따라 분류하였다. 동일한 그룹으로 분류된 자소들은 일정한 오류 내에서 중복된 글립 정보를 검색하여 삭제하고, 중복성이 제거된 자소들을 사용하지 않는 코드 영역에 단순 글립으로 저장하였다. 이렇게 단순 글립으로 저장된 자소들을 원래 문자 이미지대로 해당되는 자소들을 합성하여 폰트를 재구성하였다. 재구성된 폰트들은

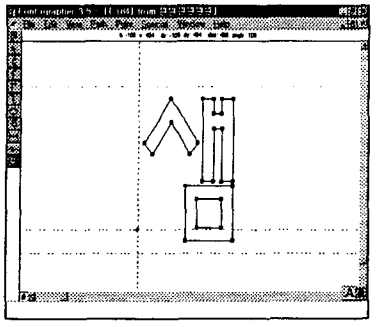


그림 3 단순 글립으로
구성된 '샘' 윈도우

중복성이 제거된 자소들이 단순 글립으로 저장되어 있고, 이들을 합성하여 구성된 한글 2350자가 합성 글립으로 저장되어 있다.

그림 3과 4는 단순 글립으로 구성되어 있는 '샘'을 나타내는 윈도우와 폰트내의 그 내용을 보여준다. 그림 5와 6는 합성 글립으로 구성하였을 때, 샘체의 '샘'과 폰트 내의 내용을 보여준다. 단순 글립으로 구성하였을 경우는 244 bytes가 소요되는 반면 합성 글립으로 구성하였을 때는 30 bytes만이 소요된다. 샘체 전체에서 중복된 글립을 제거하였을 때 원래의 크기 408,044 bytes에서 173,020 bytes로 줄어들었으며, 이는 원래 크기의 42.4%에 해당한다. glyf 테이블의 크기는 316,524 bytes에서 81,734 bytes로 줄어들었으며 이것은 원래 글립의 25.8%에 해당한다. 샘체는 예상보다 적은 공간

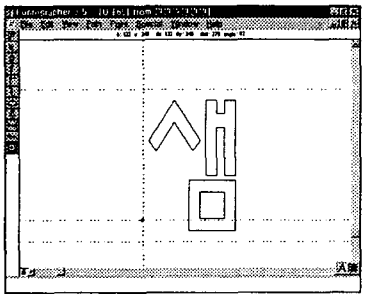


그림 5 합성 글립으로
구성된 '샘' 윈도우

```
Glyph  55: off = 0x000019C2, len = 244
    numberOfContours:          4
    xMin:                      22
    yMin:                      -34
    xMax:                      336
    yMax:                      403
  EndPoints
  ---------
    0:  5
    1:  17
    2:  21
    3:  25
  Length of Instructions:      :  161
  00000: NPUSHB       (78):    4   3   11   5
                      <중략>
                        ⋮
  Flags
  -----
    0:        XDual                X-Short On
                      <중략>
                        ⋮
  Coordinates
  -----------
  0: Rel (  116,   403)  ->  Abs (  116,   403)
  1: Rel (-94,  -138)  ->  Abs (   22,   265)
                      <중략>
                        ⋮
  25: Rel (   0,   168)  ->  Abs (  336,   134)
```

그림 4 단순 글립으로 저장된
'샘'의 glyf 테이블 데이터

이 절약되었는데 이는 샘체의 글립이 매우 단순하여 글립 정보가 다른 글꼴에 비해 전체 폰트 파일에서 차지하는 비율이 적기 때문이고, 샘체의 글립이 매우 단순한 모양이어서 줄어드는 글립 정보와 늘어나는 합성 글립 정보와의 상대적인 비율 때문이라고 생각된다. 샘체 폰트

```
Glyph  56: off = 0x00001AB6, len = 30
  numberOfContours:          -1  (Composite)
  xMin:                      22
  yMin:                      -34
  xMax:                      336
  yMax:                      404
    0: Flags:                0x0026
       Glyf Index:           13
       X BOffset:            0
       Y BOffset:            0
       Other:    Round X,Y to Grid
    1: Flags:                0x0026
       Glyf Index:           24
       X BOffset:            0
       Y BOffset:            2
       Other:    Round X,Y to Grid
    2: Flags:                0x0007
       Glyf Index:           10
       X WOffset:            136
       Y WOffset:            -268
       Other:    Round X,Y to Grid
```

그림 6 합성 글립으로 저장된
'샘'의 glyf 테이블 데이터

표 3 트루타입 폰트 테이블별 크기 비교

테이블	명조체.ttf	중복성을 제거한 명조체.ttf
OS/2	78	78
cmap	260	260
cvt	4	4
fpgm	6	6
glyf	791,736	196,776
head	54	54
hhea	36	36
hmtx	8,724	8,724
loca	13,428	13,428
maxp	32	32
name	286	286
post	418	418
prep	10	10

의 품질은 단순 글립으로 구성한 샘체와 중복성을 제거한 샘체와 동일하였다.

명조체는 단순 글립으로 구성된 명조체 트루타입 폰트의 크기가 815,328 bytes이고 중복된 글립들을 합성 글립으로 구성하였을 때는 220,368 bytes이므로, 원래 크기의 27.0%에 해당되는 크기로 저장 공간의 크기가 줄어들었다. 표 3은 단순 글립으로 구성된 명조체 트루타입 폰트와 중복성을 제거한 명조체 트루타입 폰트의 테이블별 크기를 비교한 것이다. 표에서 보듯이 실제 폰트의 윤곽선 정보를 담고 있는 glyf 테이블의 크기는 791,736 bytes에서 중복성을 제거하였을 때 원래 크기의 24.9%에 해당하는 96,776 bytes로 줄어들었다. 표 4는 중복성을 제거한 명조체 폰트의 초·중·종성별 글립들의 개수를 나타낸다. 초성이 중성과 종성에 비해 비교적 많은 벌수가 필요하며, 전체 글립의 개수는 760개이다.

그림 7의 (a), (b)는 단순 글립으로 구성된 명조체와 중복성을 제거한 명조체를 비교한 결과물이다. 출력 결과에서 알 수 있듯이 원래의 명조체 폰트와 중복성을 제거하고 합성 글립으로 폰트를 재구성한 폰트는 품질의 차이를 거의 느낄 수 없다.

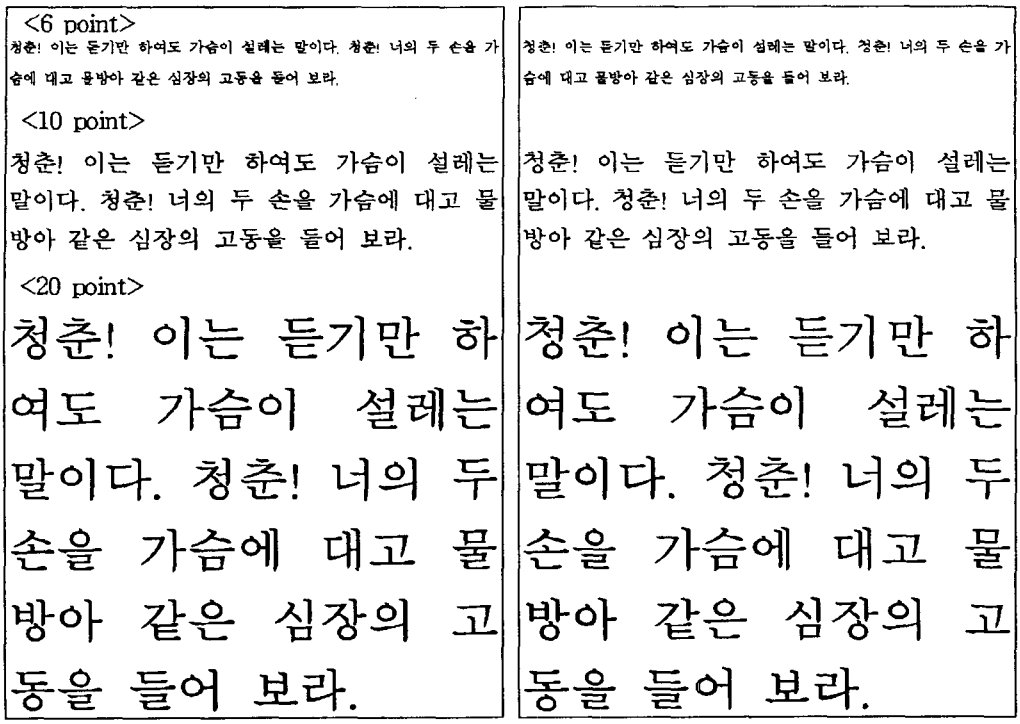
표 4 중복성을 제거한 명조체의 초·중·종성별 글립 갯수

초성	394	중성	273	종성	93
ㄱ	21	ㅏ	5	ㄱ	5
ㄲ	20	ㅐ	10	ㄲ	3
ㄴ	24	ㅑ	5	ㄳ	3
ㄷ	26	ㅒ	5	ㄴ	5
ㄸ	24	ㅓ	25	ㄵ	2
ㄹ	25	ㅔ	25	ㄶ	2
ㅁ	18	ㅕ	23	ㄷ	3
ㅂ	20	ㅖ	16	ㄹ	8
ㅃ	11	ㅗ	17	ㄺ	4
ㅅ	21	ㅘ	16	ㄻ	4
ㅆ	20	ㅙ	16	ㄼ	2
ㅇ	21	ㅚ	16	ㄽ	1
ㅈ	19	ㅛ	14	ㄾ	2
ㅉ	18	ㅜ	11	ㄿ	1
ㅊ	19	ㅝ	11	ㅀ	2
ㅋ	20	ㅞ	11	ㅁ	5
ㅌ	24	ㅟ	11	ㅂ	7
ㅍ	22	ㅠ	10	ㅄ	2
ㅎ	21	ㅡ	13	ㅅ	5
		ㅢ	8	ㅆ	4
		ㅣ	5	ㅇ	5
				ㅈ	3
				ㅊ	3
				ㅋ	2
				ㅌ	4
				ㅍ	3
				ㅎ	3

6. 결론 및 향후 연구 방향

완성형 폰트에서 중복된 글립을 추출하여 합성 글립으로 구성함으로써 폰트의 품질을 최대한 유지하면서 폰트의 저장 공간을 절약하고, 폰트 작성시 수고와 노력을 줄일 수 있는 중복성을 최소화한 한글 폰트를 얻을 수 있었다. 실험 결과 샘체 트루타입 폰트는 기존 폰트가 요구하는 저장 공간의 42.4%로 줄어들었으며, 명조체 트루타입 폰트는 기존 폰트의 27.0%로 그 크기가 줄어들었다. 중복성을 최소화한 폰트는 향후 인터넷 출판 환경에서 연구되고 있는 임베디드 폰트(embeded font)에 유용하게 사용될 것이다.

현재 기존의 완성형 폰트에서 중복된 글립을 자동 추출하여 합성 글립으로 구성하는 도구에 대한 연구를 진행하고 있다. 또한 한글 폰트 편집기에서 합성 글립을 쉽게 작성할 수 있는 편집기에 대한 연구가 이루어져야 한다.



(a) 명조체 (b) 중복성을 제거한 명조체

그림 7 명조체와 중복성을 제거한 명조체 품질 비교

7. 참고문헌

[1] 이준희, 정내권, "컴퓨터 속의 한글," 정보시대, pp.457-473, 1991.

[2] 변정용, "훈민정흠 원리의 공학화에 기반한 한글 부호계의 발전 방향," 정보과학회지, 제12권, 제8호, pp.73-76, 1994.

[3] Microsoft Typograhpy, available http://www.microsoft.com/typography/

[4] 정의훈, *트루타입 글자꼴을 위한 하드웨어 글자 생성기의 설계*, 한국과학기술원, 석사 학위논문, 1994.

[5] Roger D. Hersch, Ecole Polytechnique Federale, and Lausanne, *Visual and Technical Aspect of Type*, pp.110-125, Cambridge University Press, 1993.

[6] Peter Karow, *Digital Typefaces Description and Formats*, Springer-Verlag, 1994.

[7] Microsoft Corp., *TrueType Font Files*, Microsoft Corp., Nov. 1995.

[8] 안상수, "폰토그라퍼," 안그라픽스, 1993.

[9] Microsoft Typography, "TTFDump Quick Reference Guide," 1995.

[10] 안은영, 조형제, "기본 획 합성에 의한 한글글꼴 생성", 정보과학회 논문지, 제21권, 제4 호, pp.649-651, 1994.