슬라브 번호 인식 장치의 개발

홍기상, 장정훈, 양종렬, 김승진, 김태원

포항공과대학교

1. 서 론

제철소에서 연주 및 분괴 공정 등을 통하여 출하되는 슬라브(slab)의 표면에는 서로의 구분을 위하여 고유번호가 기록되어 있으며, 슬라브가 열연 공정에 들어오면 조업원이 호스트 컴퓨터에서 보내 온 번호와 슬라브에 기록되어 있는 번호가 일치하는가를 검사함으로써 물류관리를 하고 있다. 지금까지는 수동으로 이를 확인하였기 때문에 효율적인 관리가 어려웠다. 이 문제를 해결하기 위하여 슬라브 번호를 자동으로 인식하는 장치의 개발이 필요하게 되었다. 이러한 시스템을 개발하므로써 물류관리의 완벽한 전산화를 이룩하고, 이를 바탕으로 생산성의 증대를 꾀할 수 있다.

문자 인식은 문자 영상 정보를 얻는 방식에 따라 온라인(on-line) 문자 인식과 오프라인(off-line) 문자 인식으로 분류할 수 있고, 인식 대상에 따라 인쇄체 문자 인식과 필기체 문자 인식으로 분류할 수 있다. 또한 문자 인식이 시행되는 환경에 따라 옥내 환경 하에서의 인식과 옥외 환경 하에서의 인식으로 나눌 수 있다.

슬라브 번호 인식은 위의 분류에 따르자면 오프라인 - 필기체/인쇄체 - 옥외환경 문자인식에 속한다. 입력 영상은 카메라를 통하여 얻어진다. 슬라브 번호는 필기체로 기록되어 있는 경우와 인쇄체로 기록되어 있는 경우가 모두 존재한다. 필기체의 경우에는 이웃하는 숫자끼리 집촉하여 있는 경우가 상당수 존재한다. 그리고 인쇄체의 경우에는 사용되는 폰트의 종류가 두가지가 존재한다. 일반적으로 슬라브 표면은 불균일하고 이물질등이 묻어 있으며, 여러 공정을 거치면서 발생하는 접촉에 의해 숫자들이 손상된다. 그리고 낮과 밤에 따라 조명이 변화하고 주위에 기중기와 같은 대형 장비가 이동함에 따라 그림자가 슬라브 주위에 발생한다. 이렇듯 슬라브 번호 인식에서는 인식을 어렵게 만드는 여러 가지 요인들이 복합적으로 존재하고 있다. 이러한 요인들을 통칭 노이즈(noise)라고 한다면, 슬라브 번호 인식 장치을 개발하는데 있어서, 기존의 인식 시스템에서 채용한 알고리즘보다 노이즈에 훨씬 강건한 알고리즘이 필요하다. 따라서 본 시스템을 구성하는데 있어서 중점을 둔 것은 전체 알고리즘을 구성하는 각 부분이 모두 노이즈에 강한 특성을 갖도록 하는 것이었다.

본 인식 시스템에서는 숫자 영역과 배경 영역을 분리하기 위하여 기존의 이진화 방법과 더불어 개념적으로 새로운 방법을 개발하여 적용하였다. 또한 인식부에서는 입력 영상으로부터 특징을 추출할 필요가 없고, 노이즈에 강건한 KLT를 이용한 인식 방법을 채용하였다. 그리고 사전 정보를 적절하게 활용하여 인식률을 향상시켰다.

그림 1에서 대표적인 슬라브 영상들을 보이고 있다. 일반적으로 번호가 슬라브의 앞면에 기록되어 있는 경우에는 필기체로, 옆면에 기록되어 있는 경우에는 인쇄체로 되어 있

그림 1-1. 슬라브 앞면 영상의 예.

그림 1-2. 슬라브 옆면 영상의 예.
(가는 폰트로 번호가 인쇄되어 있는 경우)

그림 1-3. 슬라브 옆면 영상의 예
(굵은 폰트로 번호가 인쇄되어 있는 경우)

다. 인쇄체의 경우에는 가는 폰트와 굵은 폰트의 두가지 종류가 있다. 이웃하는 문자들의 접촉은 앞면 필기체 번호의 경우 빈번하게 발생하는 반면, 옆면 인쇄체 번호의 경우에는 드물다. 문자와 배경의 밝기 차이가 적다던가 배경의 노이즈 또는 글자의 훼손 정도가 가장 심한 것은 일반적으로 옆면의 가는 폰트로 인쇄되어 있는 번호이고, 앞면은 중간 정도이며, 옆면의 굵은 폰트로 인쇄되어 있는 것은 대부분 선명하다. 그러나 굵은 폰트로 인쇄된 번호의 경우, 일단 문자가 손상되면 그 정도가 심한 것이 특징이다.

본 논문이 작성되고 있는 현재, 인식 시스템의 개발이 계속 진행 중인 상태이다. 현재 상태에서 완성도가 가장 낮은 부분이 슬라브 앞면의 필기체 번호 인식 쪽이므로 본 논문에서는 필기체의 인식에 대해서는 언급하지 않고 앞면의 번호 영역을 찾는 알고리즘까지만을 언급할 것이다. 또한, 슬라브 옆면에 존재하는 두가지 형태의 번호 중, 가는 폰트로 인쇄되어 있는 쪽이 인식하는데 있어서 한결 어려움이 따르므로 본 논문에서는 가는 폰트의 번호 인식을 중심으로 설명하고 굵은 폰트에 대해서는 차이점을 중심으로 간략하게 언급하기로 한다.

본 논문의 구성을 보면, 제 2장에서는 슬라브 번호 인식 장치의 하드웨어 및 소프트웨어의 구성에 대해 설명하고, 제 3장에서는 각 구성 요소별 주요 알고리즘을 설명한다. 제 4장에서는 실험 및 결과를, 제 5장에서는 결론 및 향후 개발 방향에 대하여 언급한다.

2. 슬라브 번호 인식 장치의 구성

2.1. 하드웨어의 구성

슬라브 번호 인식 장치의 하드웨어 구성을 그림 2에서 보이고 있다. 슬라브는 다른 장소에서부터 기중기에 의해 옮겨져서 푸쉬어(pusher) 앞에 쌓이게 된다. 조업원이 스윗치를 누르면 푸쉬어가 작동하여 슬라브를 하나씩 롤러(roller) 위로 밀어낸다. 푸쉬어가 끝까지 도달하였을 때, 센서가 PC에 신호를 보내고, PC는 내장되어 있는 프레임 그레버(frame grabber)를 구동시켜서 슬라브의 앞면과 옆면에 장치되어 있는 네 대의 카메라로부터 영상을 획득한다. 슬라브의 옆면에는 세 대의 카메라가 장착되어 있는데, 넓은 시야를 확보하기 위하여 광각 렌즈가 장착되어 있으며, 이웃하는 카메라의 시야는 약간씩 중첩되어 있다. 프레임 그레버 앞에서는 비디오 멀티플렉서 (video multiplexer)가 있어서, 네 대의 카메라로부터 영상을 순차적으로 획득한다. 획득한 영상의 해상도는 624×468이다. 카메라는 모두 고정이 되어 있으므로 영상에서 슬라브가 나타나는 범위는 거의 일정하다. 따라서 획득한 영상의 저장 시, 전체를 저장하는 것이 아니고 슬라브가 포함된 일부 영역만을 저장하게 된다.

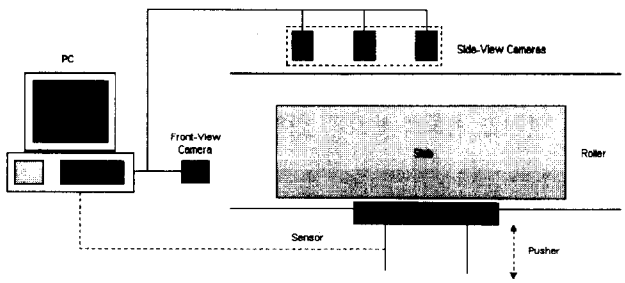


그림 2. 슬라브 번호 인식 장치의 하드웨어 구성.

현재는 실험 단계이므로 아직 장치는 되어 있지 않지만, PC는 호스트 컴퓨터와 연결되어 인식 전에 슬라브 번호에 대한 사전 정보를 넘겨 받고, 인식 결과를 사전 정보와 비교하여 그 결과를 다시 호스트 컴퓨터에 전송할 계획이다.

2.2. 소프트웨어의 구성

슬라브 번호 인식 장치의 소프트웨어 구성 및 흐름을 그

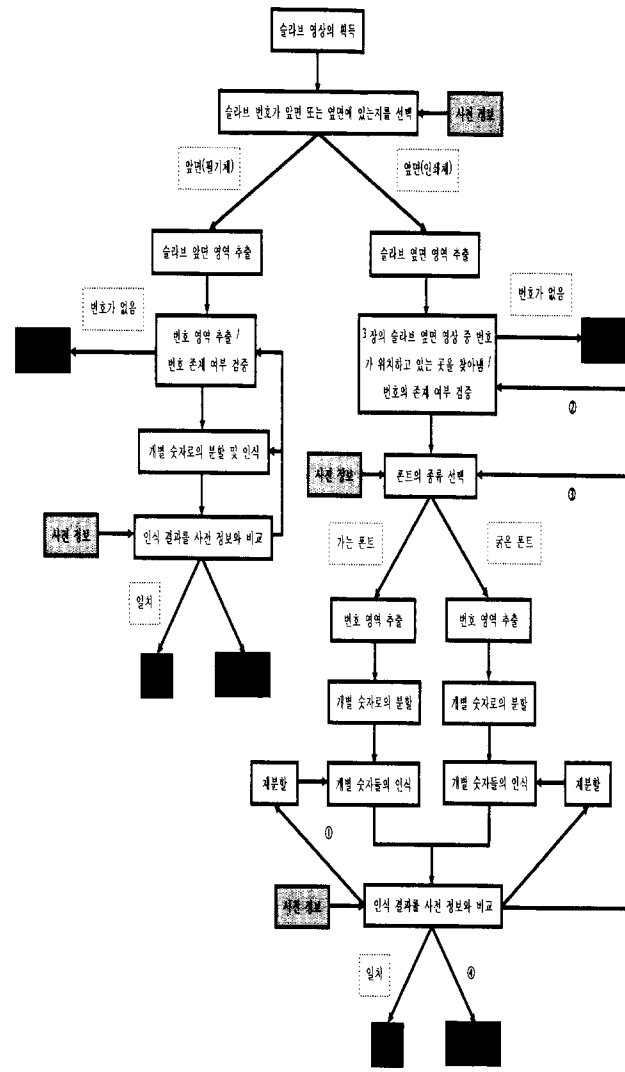


그림 3. 슬라브 번호 인식 장치의 전체 흐름도.

림 3에서 보이고 있다.

본 인식 장치의 목적은 모르는 대상에 대한 인식 결과를 알려주는 것이 아니고, 인식 대상에 대한 사전 정보가 있을 때, 그 대상이 예정대로 온 것인지를 판단하는, 이른바 확인을 목적으로 하는 시스템임을 염두해 둘 필요가 있다. 즉, 인식 결과를 사전 정보와 비교하여 같으면 OK를, 다르면 WARNING을 출력하는 것이다. 그리하여 WARNING이 발생하면 예정된 슬라브가 온 것이 아니기 때문에 조업원은 적절한 조치를 취하게 된다. 그림 3에서 OK 또는 WARNING이 발생하는 곳들을 표시하였다.

인식 대상에 대한 사전 정보가 주어지므로 인식 시스템에서는 그것과 완전히 독립적으로 인식을 한 후 최후에 사전 정보와 비교를 하는 것보다는 인식 과정에서 사전 정보를 활용하는 것이 타당하다. 그림 3에서 사전 정보라고 표시되어 있는 곳에서 그것을 활용한다. 그런데, 사전 정보를 무조건적으로 이용할 수는 없고, 검증하는 부분이 포함되어 있어야 할 것이다.

사전 정보로는 슬라브의 폭, 두께, 깊이, 슬라브 번호, 슬라브 번호를 인쇄한 기계(marking machine)의 ID이다. 여기에서 인식할 때 유용하게 이용할 수 있는 것이 슬라브 번호와 인쇄 기계의 ID이다. 번호 인쇄 기계의 ID가 없는 경우에는 사람에 의해 슬라브의 앞면에 번호가 기록되어 있음을 의미한다. ID가 있는 경우에는 번호가 옆면에 인쇄되어 있음을 뜻하게 되고, 그 값에 따라 인쇄 시 사용된 폰트의 종류를 알 수 있게 된다. 따라서 번호가 앞면 또는 옆면에 있는가와 사용된 폰트의 종류가 무엇인가에 대한 판단은 사전 정보를 그대로 이용하면 된다.

사전 정보로 주어진 슬라브 번호는 인식 시 혼동되는 숫자들의 검증이나 개별 숫자로의 재분할 시 사용하게 된다.

사전 정보에 대한 검증 방법으로는 두가지를 들 수가 있다. 첫번째는 사전 정보를 이용한 부분의 다음 단계에서 검증하는 것이다. 이전 단계의 가정이 틀리다면 대부분의 결과가 예상되는 결과보다 못할 것이다. 두번째는 인식 결과를 이용하는 것이다. 결국 인식 전에 사용된 사전 정보는 인식 결과에 모두 영향을 끼치므로, 사전 정보가 인식 대상과 불일치하면 엉뚱한 인식 결과가 나올 것이다. 본 시스템에서는 두가지 방법을 모두 이용하였다.

위에서 언급한 내용을 바탕으로 슬라브 번호 인식 장치의 흐름을 살펴보면 다음과 같다.

우선 슬라브의 앞면에 설치되어 있는 한 대의 카메라와 옆면에 설치되어 있는 세 대의 카메라로부터 총 네 장의 영상을 얻는다. 그러면, 사전 정보에 의해 슬라브 번호가 앞면에 있는지 또는 옆면에 있는지를 선택한다. 만약 앞면이 선택되면 앞면에 설치되어 있는 카메라로부터 얻은 영상에서 슬라브 앞면 영역만을 추출한다. 만약 옆면이 선택되면 세 장의 영상에서 슬라브의 옆면에 해당되는 영역만을 추출하게 된다. 슬라브 영역 추출 알고리즘은 3.1절에서 설명한다.

앞면이 선택된 경우, 그 다음으로 번호 영역을 추출한다. 즉, 번호를 둘러싸는 상자(bounding box, 이하 BB)를 찾는 것이다. 이 때, 번호의 존재 여부도 검증하게 된다. 그것은 BB의 기하학적인 모양에 의해 어느 정도 판단이 가능하다. 따라서 번호가 없다고 판단되면 WARNING을 출력하고 종료한다. 슬라브 앞면 번호 영역 추출 알고리즘은 3.2절에서 설명하고, 본 논문에서는 앞면에 대해서는 여기까지만을 언급한다.

옆면이 선택된 경우, 추출한 세 장의 옆면 영상에서 번호가 존재하고 있는 영상을 찾아낸다. 이 때, 사전 정보가 틀렸다면 번호가 존재하지 않을 수도 있으므로 이것에 대한 검증이 필요하다. 번호가 있는 영상을 선택하는 방법으로, 각 영상마다 번호가 존재하는 정도를 점수화하여 그 점수가 가장 큰 영상을 선택하게 된다. 만약 가장 큰 점수가 일정량 이하가 되면 세장의 영상 모두에 대하여 번호가 없다고 보고 WARNING을 출력한 다음 종료한다. 자세한 알고리즘은 3.3절에서 설명한다.

번호가 포함된 영상을 찾으면, 사전 정보에 의해 인쇄되어 있는 번호의 폰트의 종류를 선택하게 된다. 이 정보에 대한 검증으로는 인식 결과를 이용한다.

가는 폰트나 굵은 폰트에 대하여 그 이후의 처리 과정은 동일하다. 그러나 알고리즘 측면에 있어서는 약간 다르다. 결정적으로 다른 부분은 번호 영역을 추출하기에 앞서 수행되는 문자 영상과 배경 영상의 분리이다. 기존의 문자 인식 시스템에서는 입력 영상에는 노이즈가 거의 포함되어 있지 않다는 가정를 두고 있기 때문에 간단한 이진화 알고리즘을 이용하였다. 그러나 서론에서도 언급하였듯이 슬라브 영상에는 배경과 숫자의 분리를 방해하는 요소가 많이 있기 때문에 기존의 이진화 알고리즘만을 이용할 경우에는 만족스러운 결과를 얻기가 어렵다. 본 인식 시스템에서는 굵은 폰트로 인쇄되어 있는 번호의 경우에는 배경과의 밝기 차이가 대부분 뚜렷하게 나기 때문에 계산량이 적은 기존의 이진화 알고리즘을 이용하였으나 가는 폰트로 인쇄되어 있는 번호의 경우에는 그 특성에 맞는 이진화 알고리즘을 자체적으로 개발하여 적용하였다. 자세한 알고리즘은 3.4절에서 설명한다. 이 후의 알고리즘은 가는 폰트의 경우와 굵은 폰트의 경우가 동일하나 사용되는 파라메타들은 서로 다르다.

이진화된 슬라브 번호 영상은 번호 영역의 추출과 개별 숫자로의 분할에서 사용하게 된다. 번호 영역의 추출은 번호의 BB를 찾는 것을 의미한다. 이 알고리즘에 관해서는 3.5절에서 설명한다. 번호 영역이 추출되면, 그 안에 있는 개별 숫자들을 분리해내야 한다. 이 때, 이웃하는 숫자 사이의 경계를 정확하게 찾아내는 것이 중요하다. 경계를 찾는 알고리즘은 3.6절에서 설명한다. 경계가 결정되면, 경계 사이에 존재하는 숫자 영역을 좀 더 정확하게 찾은 후, 원래의

명도 영상에서 그 영역에 해당되는 부분을 가져와 인식에 이용하게 된다.

기존의 인식 알고리즘에서는 대부분 원래의 명도 영상을 이진화한 후, 그것으로부터 특징(feature)을 추출하여 인식기의 입력값으로 사용하였다. 본 인식 시스템에서도 입력 영상의 이진화된 결과를 얻지만, 그것은 개별 숫자로의 분할까지만 사용되고, 인식에는 원래의 명도 영상이 직접 사용된다. 그 이유는 이진화된 영상을 보면 배경에 속하는 부분도 나타나거나 문자의 일부가 훼손되어 있는 경우가 자주 발생하기 때문이다. 이러한 경우 신뢰할만한 특징을 추출하기가 어렵다.

본 시스템에서 채용하고 있는 인식기는KLT(Karhunen-Lo ve transform)를 이용한 부분공간 분류기(subspace classifier)이다. 이 인식기의 특징은 명도 영상을 하나의 다차원 벡터로 보고 입력받는다는 것이다. 즉, 인식부와 특징 추출부가 특별히 구분되어 있지 않다. 또한 지금까지 발표된 여러 실험 결과들에 따르면 노이즈에 상당히 강건하다는 특성을 가지고 있다. 따라서 슬라브 번호와 같은 부류의 문자를 인식하기에 적합하다고 말할 수 있다. 대략적인 알고리즘은 3.7절에서 설명한다.

인식 결과는 사전 정보에서 주어진 번호와 비교하게 된다. 만약 일치하면 OK를 출력하고 종료하게 되지만 일치하지 않는 경우에는 조건에 따라 다른 피드백이 동작하게 된다. 그 중에서 대표적인 피드백이 재분할이다. 재분할 루틴이 동작할 조건은 인식 결과의 일부가 사전 정보와 일치하지 않는 경우이다.

일부가 일치하지 않는 경우를 크게 두가지로 나누어보면 인식기로부터 나온 번호의 문자열 길이와 사전 정보에서 주어진 번호의 문자열 길이가 일치하는 경우와 일치하지 않는 경우로 나눌 수 있다.

문자열의 길이가 일치하는 경우를 다시 두가지로 나누어 보면 개별 숫자의 분할 위치는 정확하나 인식기 자체에서 인식 오류가 발생한 경우와 분할 갯수는 맞으나 분할 위치가 어긋나서 인식 오류가 발생한 경우로 나눌 수 있다. 전자의 경우, 대부분 인식기에서 유사한 모양의 숫자간에 혼동을 일으켜서 발생했을 가능성이 크다. 본 시스템에는 통계적으로 얻은 혼동 클래스들(confusion class - 혼동되는 숫자들이 하나의 클래스를 이룸)을 바탕으로 각 혼동 클래스 내의 숫자들을 전문적으로 구분해내는 루틴이 포함되어 있다. 예를 들어 입력된 숫자 영상에 대하여 인식기에서는 1로, 사전 정보에서는 7로 주어졌다고 가정하자. 마침 (1 ↔ 7)의 혼동 클래스가 등록되어 있다면 그것에 해당되는 루틴이 동작하여 인식 결과를 검증하게 된다. 검증 알고리즘은 3.8절에서 설명한다. 만약 (1 ↔ 7)의 혼동 클래스가 등록되어 있지 않다면 이것은 분할 위치가 어긋나서 발생한 오류로 본다. 따라서 이런 경우에는 재분할 루틴이 동작하여 분할 위치를 어느 정도 수정한 후 다시 인식을 시도하게 된다.

문자열의 길이가 일치하지 않는 경우에는 분할을 덜 했거나 더 한 경우인데, 인식기에서 나온 번호의 문자열과 사전 정보에 의한 번호의 문자열을 비교(string matching)하여 덜 한 곳 또는 더 한 곳을 추정해낼 수 있다. 재분할 루틴에서는 그곳에서 잘못된 점을 시정하여 다시 인식을 시도하게 된다. 구체적인 재분할 알고리즘은 3.10절에서 설명한다.

그림 3에서 피드백 ②와 ③은 인식 결과와 사전 정보의 불일치 정도가 매우 클 때 작동한다. 피드백 ②의 경우에는 다음으로 번호가 존재할 확률이 큰 영상이 선택되고, 피드백 ③의 경우에는 폰트의 종류가 바뀌게 된다. 만약 모든 피드백의 시도에도 불구하고 사전 정보와 인식 결과가 일치하지 않으면 WARNING을 출력하고 종료한다. 그림 3에서 표시된 피드백이 발생할 조건에 대해서는 3.9절에서 설명한다.

3. 인식 장치를 이루는 요소들의 알고리즘

3.1. 슬라브 앞면과 옆면 영역의 추출

카메라로부터 슬라브 영상이 입력되면 사전 정보에 따라 슬라브의 앞면 또는 옆면을 처리할지를 선택한 후, 슬라브에 해당되는 영역만을 추출하게 된다. 이렇게 함으로써 번호 영역을 찾을 때의 부담을 줄일 수 있다. 슬라브 앞면/옆면 영역을 찾는 알고리즘의 기본이 되는 것은 에지(edge)를 추출한 후, 슬라브 모서리에 해당되는 직선들을 Hough 변환을 이용하여 찾는 것이다. 그런데, 앞면 영역 추출의 경우 입력 영상에서 추출한 에지만을 이용해서는 원하는 선들을 만족스럽게 찾아낼 수 없었다. 그래서 함께 사용한 것이 슬라브가 진입하기 전에 획득한 배경 영상(I_{bg})과 슬라브가 진입한 후에 획득한 영상(I)과의 차영상(difference image)이다. 차영상 I_{dif}의 정의는 다음과 같다.

$$I_{dif}[i, j]=\begin{cases}255 & \text{if } |I_{bg}[i, j]-I[i, j]|>\delta \\ 0 & \text{otherwise}\end{cases} \quad (1)$$

여기에서 δ는 파라메타(parameter)이다. 그림 5에서 차영상을 보면 슬라브 앞면에 해당되는 영역이 대략적으로 나타나고 있음을 알 수 있다.

그림 4에서 구해야 할 선들을 보이고 있다. I_{disk}를 수평 투

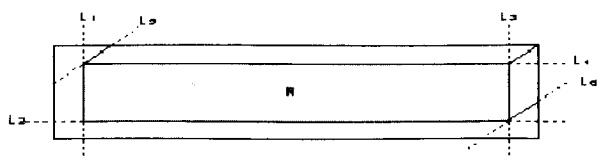


그림 4. 슬라브 앞면 영역의 추출 과정.

그림 5. 슬라브 앞면의 추출 예와 차영상.

영하여 영역 R을 대략적으로 찾은 후, 그 안에서 I의 에지를 Hough 변환하여 L_2와 L_4를 정확하게 찾아낸다. L_5는 기울기가 거의 일정하다는 조건을 이용하여 구한다. L_6는 거의 고정되어 있다고 말할 수 있는데, 푸쉬어가 항상 일정하게 나오기 때문이다. L_1은 L_4와 L_5의 교점을, L_3는 L_2와 L_6의 교점을 이용하여 구한다. 그림 5에서 앞면의 추출 예를 보이고 있다.

슬라브 옆면의 추출 방법은 앞면의 추출 방법보다 단순한데, 옆면 영역은 대부분 입력 영상을 수평으로 가로질러 위치하고 있기 때문이다.

입력 영상에서 구한 에지를 Hough 변환하여 파라메타 공간(parameter space)을 생성한 다음 가장 긴 선(점수가 가장 큰 직선)을 찾는다. 그 직선을 그림 6에서와 같이 편의상 L_1이라고 하자. 그러면, L_1과 일정 거리(슬라브의 두께와 관련) 이상 떨어져 있는 선들 중 가장 긴 선을 찾고, 그 선이 $x=0$와 만나는 점을 구한다. 그 점을 지나고 L_1과 평행이 되는 선이 L_2가 된다. 일반적으로 슬라브는 약간 기울어져 있기 때문에 L_1과 L_2사이의 영역을 회전 변환하여 완전한 수평으로 바로 잡는다.

그림 6. 슬라브 옆면 영역을 찾은 모습.

3.2 슬라브 앞면 번호의 영역 추출

슬라브 앞면 번호 영상의 특징은 그림 1-1에서 보듯이 배경의 노이즈와 더불어 석필에 의하여 쓰여진 세선(thin line) 문자들이 존재한다는 점이다. 따라서 앞면 번호 영역의 추출 시 이러한 것들을 효과적으로 제거해야 한다. 석필에 의하여 쓰여진 문자들과 비교해볼 때 슬라브 앞면 번호의 특징은 상대적으로 선이 굵고 배경과의 밝기 차이가 크다는 점이다. 이러한 특징을 바탕으로 한 번호 영역 추출 알고리즘은 다음과 같다.

우선 Sobel 연산자를 이용하여 입력 영상의 gradient를 구하고, 그것에 adaptive smoothing 필터를 적용한다 [1]. 그 결과를 I_{gas}라고 하자.

그림 7에서 그림 1-1을 입력 영상으로 했을 때의 처리 결

그림 7. adaptive smoothing을 한 모습.

그림 8. 이진화한 결과.

그림 9. run-length 필터 등을 적용한 모습.

그림 10. 슬라브 앞면 번호를 찾은 모습.

과를 보이고 있다. 번호 영역에 비해 배경 영역이 평활화가 더 많이 이루어졌음을 알 수 있다.

I_{gas}는 Otsu의 전역적인 이진화 방법에 의해 이진화된다 [2](그림 8 참조). 그런 후, run-length 필터를 적용하여 수평으로 일정 거리 이내에 있는 픽셀들을 서로 연결한다 [3]. 결과적으로 번호 영역을 강조하는 효과를 보게 된다. 그런 다음, morphological opening을 이용하여 가늘게 이어진 영역들을 분리하고, 면적이 작은 고립된 영역들은 제거한다. 그림 9에서 처리 결과를 보이고 있다. 마지막으로, CRE(connected region extraction)를 하여, 고립된 영역들마다 기하학적인 정보를 얻는다. 그리하여 번호 영역으로서는 부적당하다고 판단되는 것들은 제거한다. 남은 번호 후보 영역에서 그 영역의 평균적인 수평, 수직 위치로부터 많이 벗어나는 불필요한 부분을 제거한 후, 남은 영역을 둘러싸는 상자를 구한다. 그림 10에서 그 결과를 보이고 있다. 만약 번호 후보 영역이 존재하지 않으면 번호가 존재하지 않는 것으로 판단한다.

3.3 슬라브 번호가 포함된 옆면 영상의 선택

현재 슬라브의 옆면 영상을 얻기 위해 3대의 카메라가 설치되어 있는데 슬라브의 길이가 다양하고 옆면 번호의 위치도 일정하지 않기 때문에 3장의 영상 중 어디에 슬라브 번호가 있는지를 판단해야 한다. 이 때, 이웃하는 카메라들의 시야가 어느 정도씩 중첩되어 있는 관계로 슬라브 번호의 동일한 부분이 두 영상에 동시에 나타날 수도 있으므로, 이

런 경우 가장 온전한 번호 영상이 있는 곳을 선택할 수 있어야 한다. 나아가 슬라브 번호가 존재하지 않을 수도 있다는 점도 같이 고려해야 한다.

이러한 것들을 일관되게 판단하기 위해 각 영상마다 번호의 존재에 대한 정보를 얻어, 그 정도를 점수화한다. 그리하여 최고 점수를 얻는 영상이 번호를 포함하고 있다고 본다. 만약 최고 점수가 특정량 이하이면 번호는 3장의 영상 모두에 대하여 존재하지 않는다고 본다.

번호의 존재에 대한 정보를 얻기 위해 3.4.1절에서 제시한 이진화 알고리즘을 이용하여 3장의 영상을 모두 이진화한다. 그런 후, CRE를 하여 미리 정해진 가정(숫자의 종횡비, 수직 위치 등)에 위배되는 CR(connected region)들을 제거한다.

번호의 존재 여부를 판단하기 위한 기초적인 정보를 추출하기 위하여 이진화된 영상을 수평, 수직 투영하여 각각의 프로파일(profile)을 얻고, 가우시안 필터링(Gaussian filtering)을 한다. 수직 투영의 프로파일에서는 극대점들을 구한다. 이 때, 미미한 극대점들은 제거된다.

구한 수평, 수직 프로파일과 극대점들을 이용하여 현재 영상에 숫자가 존재할 가능성을 점수화한다. 그러기 위하여 다음의 각 특징값에 대하여 숫자 존재 가능성을 나타내는 함수를 만든다.

AvgHProj : 수평 투영의 평균값.
nOverAHProj : AvgHProj보다 큰 행의 수.
AvgVProj : 수직 투영의 평균값.
nOverAVProj : AvgVProj보다 큰 열의 수.
nPeaks : 극대점들의 수.
PeakInterval : 이웃하는 극대점들 사이의 평균 폭.

위의 값들을 입력으로 하여 그림 11의 함수들로부터 S_0, …, S_5를 구한다. 그러면 점수는 다음과 주어진다.

$$\text{score}=(S_0+S_0S_1+S_2+S_3+S_4+0.9S_5)/5.9. \qquad (2)$$

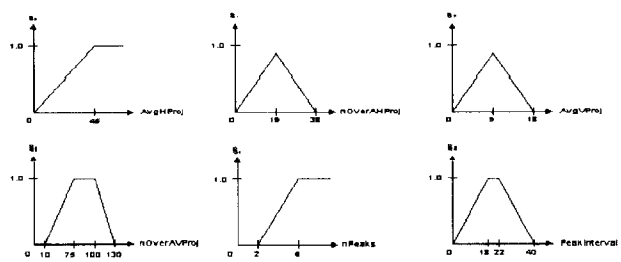


그림 11. 점수를 구하는데 필요한 함수들.

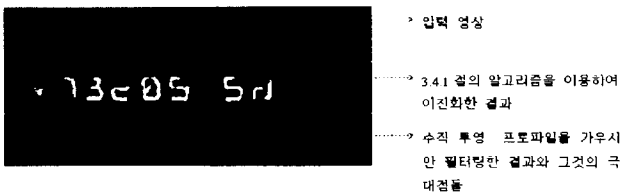


그림 12. 3.3절의 알고리즘을 수행한 예.

3.4. 슬라브 옆면의 문자 영상과 배경 영상의 분리

슬라브 번호가 굵은 폰트로 인쇄되어 있는 경우에는 대부분 배경과 밝기 차이가 크기 때문에 기존의 이진화(binarization) 알고리즘을 이용해도 무난하다. 본 인식 시스템에서는 Niblack의 알고리즘을 이용하여 이진화하였고, Yanowitz와 Bruckstein의 후처리 방법을 이용하여 노이즈를 제거하였다[4]. 그러나, 가는 폰트로 인쇄되어 있는 경우에는 배경과의 밝기 차이가 적거나 배경의 노이즈가 많기 때문에 기존의 이진화 알고리즘들로는 원하는 결과를 얻기가 어려웠다. 따라서 본 인식 장치에서는 기존의 이진화 알고리즘과는 다른 새로운 알고리즘을 개발하여 적용하였다. 새로이 개발한 알고리즘으로는 두가지가 있는데, 개념적으로 첫 번째 알고리즘은 두 번째 것의 바탕이 된다고 말할 수 있다.

3.4.1 기본적인 알고리즘

숫자 영역과 배경 영역을 구분할 수 있는 특징은 크게 두가지로 나뉘어질 수 있을 것이다. 첫번째는 숫자는 배경에 비해 밝은 픽셀들로 구성되어 있다는 점이고 두번째는 그러한 픽셀들이 모여 배경과는 다른 기하학적인 모양을 이루고 있다는 점이다. 후자의 기하학적인 특징은 분석하는 정도에 따라 결국 서로 다른 숫자들을 구분하는 정도까지 영향을 끼치게 될 것이다. 그러나 그것은 인식부의 역할이고 현 단계에서 고려할 수 있는 것은 숫자들간에는 구분을 못하지만 배경과는 구분될 수 있는 기본적인 기하학적인 특징이다. 그러한 특징으로써 문자 부분은 세선(thin line)으로 이루어져 있다는 것을 들 수 있다. 숫자를 이루는 세선들의 특징은 주위의 배경보다는 밝은 픽셀들로 구성되어 있고, 배경과 경계를 이루는 선들이 세선의 양쪽에 대부분 평행하게 놓여져 있다는 점이다. 그래서 일정 간격으로 놓여 있는 평행선들 사이의 밝은 영역을 찾아내는 것이 본 알고리즘의 목표가 된다.

본 알고리즘의 바탕이 되는 생각을 대략적으로 언급하자면, 에지에 해당되는 픽셀 주위에서 명도 영상을 이진화하여 경계치보다 큰 영역, 즉 정해진 영역 내에서 상대적으로 밝은 곳에는 점수를 올려주자는 것이다. 그러면 이진화 시 밝은 곳으로 판명되는 빈도수에 비례하여 점수를 얻게 된다. 따라서 에지가 없는 부분이 점수가 가장 낮게 되고 단일한 에지선으로 되어 있는 경우에는 점수가 좀 더 높고, 평행 에지선들 사이의 부분은 더 높은 점수를 얻을 수 있다. 따라서 후에 점수가 들어 있는 배열을 특정한 값으로 이진화하면 숫자 영역과 배경 영역을 분리해낼 수 있다. 이러한 생각을 바탕으로 한 구체적인 알고리즘은 다음과 같다.

(1) 3.3절에서 찾은 슬라브 옆면 영상이 입력 영상(I)이 된다. 이것을 가우시안 필터링한다. 그 결과를 I_{gf}라고 하자.

(2) I_{gf}에서 에지를 검출하고 그 결과를 I_{edge}라고 하자. 검출된 에지들은 1을, 나머지는 0의 값을 갖는다. 섬세한 에지

의 추출하기 위하여 facet 모델을 이용한 zero-crossing 검출기를 이용하였다 [5].

(3) 입력 영상과 같은 크기의 2차원 배열 S를 잡고, 각 요소를 모두 0으로 초기화한다.

(4) $I_{edge}[i, j]=1$을 만족하는 $I_{gt}[i, j]$ 위에 $N_{in}\times N_{in}$의 창 $W_{in}(i, j)$과 $N_{out}\times N_{out}$의 창 $W_{out}(i, j)$을 연다. 이 때 $N_{in}\leq N_{out}$이고, N_{in}과 N_{out}의 값은 숫자를 이루는 선의 굵기와 관련이 있다 (그림 13 참조).

(5) $W_{in}(i, j)[k, l] = I_{gt}[i+k, j+l]$ 라고 정의하자 $(-N_{in}/2\leq k, l\leq N_{in}/2)$. $W_{in}(i, j)[k, l]$의 값들 중 최대값을 $P_{max}(i, j)$ 라고 하고, 최소값을 $P_{min}(i, j)$라고 하자. $W_{out}(i, j)[m, n] = I_{gt}[i+m, j+n]$ 라고 정의하면 $(-N_{out}/2\leq m, n\leq N_{out}/2)$, 경계치

$$t(i, j)=\frac{P_{min}(i, j)+P_{max}(i, j)}{2} \quad (3)$$

에 대하여 $W_{out}(i, j)[m, n]$ 이 $t(i, j)$보다 크거나 같으면 $S[i+m, j+n]\leftarrow S[i+m, j+n]+1$을 한다 (그림 14 참조).

(6) (4)~(5)를 $I_{edge}[i, j]=1$을 만족하는 모든 픽셀들에 대하여 행한 후 얻은 배열 S를 특정값으로 이진화한다. 그 결과를 I_{bin}이라고 하자. 그림 15에서 그림 1-2의 영상을 처리하여 얻은 배열 S의 모습을 보이고 있다.

(7) I_{bin}에서 Yanowitz와 Bruckstein의 후처리 방법을 이용하여 불필요한 CR들을 없앤다 [4].

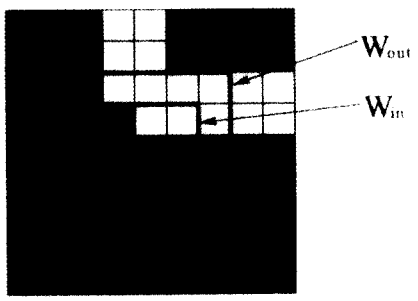


그림 13. 세선 부근의 모습.

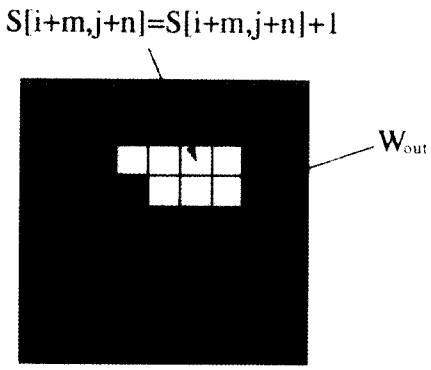


그림 14. 배열 S의 값이 증가되는 곳들.

그림 15. 배열 S의 모습.

그림 16. 배열 S를 이진화한 후 후처리까지 마친 모습.

S후처리까지 완료된 후의 결과를 I_{bin}이라고 하자. 그림 16에서 I_{bin}을 보이고 있다.

위의 알고리즘을 통하여 얻은 배열 S를 살펴보면, 앞에서 언급한 기본적인 생각과 거의 일치하는 결과를 얻었음을 알 수 있다.

위의 알고리즘은 사용하는 에지 검출기에 따라 성능이 변한다. 숫자 영역과 배경 영역의 경계가 되는 부분에서 에지가 제대로 추출되지 않으면 숫자의 획이 끊기는 현상이 발생한다. 에지 검출기의 직접적인 사용을 없애고, 세선의 검출과 배경의 평활화가 순환적인 구조로 되어 있는 것이 다음에 소개 할 발전된 알고리즘이다.

3.4.2 발전된 알고리즘

3.4.1절에서 제시한 알고리즘에서는 에지 검출기를 이용하여 찾은 에지 픽셀 주위에 대해서만 각 명도 픽셀이 숫자 영역에 속할 정도를 나타내는 점수를 구하였다. 그러나 본 알고리즘에서는 모든 명도 픽셀에 대하여 그러한 작업을 하기 때문에 특별한 에지 검출기가 필요없다.

또 한가지 크게 달라진 것은 배경 영역을 평활화한다는 점이다. 슬라브 영상을 보면 슬라브 표면이 불균일하여 노이즈에 해당되는 여러가지 무늬가 발생한다. 그것을 없애기 위하여 주위 픽셀들의 평균값보다 작은 값을 갖는 픽셀은 그 평균값으로 바꾼다. 평균값을 구할 때 각 픽셀이 기여하는 정도(가중치)가 다른데, 픽셀이 숫자 영역에 속할 정도를 나타내는 점수가 클수록 가중치는 작아진다. 이렇게 해서 얻은 새로운 영상을 다시 입력으로 받아 같은 작업을 반복한다. 몇 번 반복한 후에 얻은 영상을 보면 숫자 부분이 유지되면서 배경 부분은 평활화가 되어 있고, 그만큼 노이즈가 줄어 있다는 것을 알 수 있다. 그러면 점수가 담겨 있는 배열을 특정값으로 이진화하여 숫자 영역과 배경 영역을 분리해 낼 수 있다. 더우기 부수적으로 얻은 배경이 평활화되어 있는 영상은 후처리나 인식등에서 사용될 수도 있다. 본 시스템에서는 후처리에만 이용하였다. 알고리즘을 단계별로 설명하면 다음과 같다.

(1) 입력 영상을 I라고 하자. 그러면, I와 크기가 같은 2차원 배열 S를 잡고 각 요소를 0으로 초기화한다.

(2) I의 각 픽셀 $I[i, j]$ 에 대하여 $N_{in}\times N_{in}$의 창 $W_{in}(i, j)$과 $N_{out}\times N_{out}$의 창 $W_{out}(i, j)$을 연다. 이 때 $N_{in}\leq N_{out}$이다.

(3) $W_{in}(i, j)[k, l] = I[i+k, j+l]$라고 정의하자 $(-N_{in}/2 \le k, l \le N_{in}/2)$. $W_{in}(i, j)[k, l]$의 값들 중 최대값을 $P_{max}(i, j)$라고 하고, 최소값을 $P_{min}(i, j)$라고 하자. $W_{out}(i, j)[m, n] = I[i+m, j+n]$라고 정의하면, $(-N_{out}/2 \le m, n \le N_{out}/2)$, 경계치

$$t(i, j) = \frac{P_{min}(i, j) + P_{max}(i, j)}{2} \tag{4}$$

에 대하여

$$\begin{cases} S[i+m, j+n] \leftarrow S[i+m, j+n] + s^+(i, j) \\ \quad \text{for } I[i+m, j+n] \ge t(i, j), \\ S[i+m, j+n] \leftarrow S[i+m, j+n] - s^-(i, j) \\ \quad \text{otherwise} \end{cases} \tag{5}$$

한다. 이 때

$$s^+(i, j) = \frac{1}{1+\exp(-\lambda(x-x_0))}, \tag{6}$$

$$x = P_{max}(i, j) - P_{min}(i, j), \tag{7}$$

$$s^-(i, j) = 1 - s^+(i, j) \tag{8}$$

로 주어진다. 식 6의 함수 s^+는 λ가 커질수록 x_0에서 기울기가 커진다.

(4) I의 모든 픽셀에 대하여 (2)~(3)을 행한 후 얻은 배열 S에서 $S[i, j]$이 0보다 작거나 같으면 $S[i, j] \leftarrow 0$을 행한다. 이렇게 해서 얻은 배열 S의 각 요소의 값은 대응되는 I의 각 픽샐이 숫자 부분을 이룰 가능성에 대한 점수라고 할 수 있다.

(5) S와 동일한 크기의 2차원 배열 W를 잡는다. W의 각 요소에는

$$W[i, j] = \exp\left(-\frac{S^2[i, j]}{2\chi^2}\right) \tag{9}$$

이 대입된다. 식 9에서 주어진 함수는 $S[i, j]$가 0보다 크거나 같으면 단조 감소하는 함수임을 유념할 필요가 있다.

(6) I와 동일한 크기의 2차원 배열 $\bar{I}$를 잡는다. $\bar{I}$의 각 요소는 다음과 같이 주어진다 (배경 영역의 평활화).

$$\bar{I}[i, j] = \begin{cases} b(j, j) & \text{if } I[i, j] < b(i, j) \\ I[i, j] & \text{otherwise} \end{cases} \tag{10}$$

$$b(i, j) = m(i, j) + \mu\sigma(i, j) \tag{11}$$

$$m(i, j) = \frac{1}{P} \sum_{k, l=-M/2}^{M/2} W[i+k, j+l]\, I[i+k, j+l] \tag{12}$$

$$\sigma^2(i, j) = \left(\frac{1}{P} \sum_{k, l=-M/2}^{M/2} W[i+k, j+l]\, I^2[i+k, j+l]\right) - m^2(i, j), \tag{13}$$

$$P = \sum_{k, l=-M/2}^{M/2} W[i+k, j+l]. \tag{14}$$

그림 17. 배열 S의 모습.

그림 18. 배경이 평활화된 모습.

그림 19. 배열 S를 이진화한 후 후처리까지 마친 모습.

여기에서 $\sigma(i, j)$는 평활화를 가속시키는 역할을 한다.

(7) $I \leftarrow \bar{I}$ 한다.

(8) (1)~(7)을 일정 횟수 반복한다.

(9) 최종적으로 구한 배열 S의 각 요소를 특정한 경계치에 의해 이진화한 영상 I_{bin}을 얻는다.

위에서 제시한 알고리즘에서 파라메타 λ, χ, μ와 반복 횟수는 여러 테스트 영상에 대하여 실험을 한 후 적당한 값을 취한다. 특히 반복 횟수는 처리 시간과 밀접한 관계가 있으므로 신중하게 정해야 한다. 그림 17에서 그림 1-2의 영상을 처리하여 얻은 배열 S의 모습을 보이고 있고, 그림 18에서 배경이 평활화된 결과를 보이고 있다. 반복 횟수는 10회로 하였다.

후처리로는 다음의 3가지를 행한다.

(1) I_{bin}을 3×3 크기의 채워진 상자 모양을 갖는 SE (structuring element)로 mophological closing한다.

(2) CRE를 한 후, 이웃하는 CR들 중 유사한 성질을 갖는 CR끼리 여러 가지 SE를 이용하여 연결한다. $C(i)$에 속하는 픽셀들의 평균값과 표준편차를 각각 $m(i)$와 $\sigma(i)$라고 하면 두 CR $C(i)$와 $C(j)$가 주어졌을 때, 유사도는

$$s = |m(i) - m(j)| \tag{15}$$

로 주어지고, $s < k\sigma(i)$, $k\sigma(j)$이면 지정된 SE로 연결한다 (k는 파라메타). 이 때 사용된 SE로는 −, |, \, / 의 4가지이고, 크기는 모두 5×5이다.

(3) Yanowitz와 Bruckstein의 후처리 방법을 적용한다 [4]. 그런데, gradient 값을 구하는데 있어서 입력 영상 I을 사용하는 것이 아니고 평활화된 영상 $\bar{I}$를 사용한다. 이것을 이용하는 것이 입력 영상을 이용하는 것보다 효과적임은 실험을 통해서 증명되었다.

후처리까지 마친 후의 결과 영상을 I_{bin}이라고 하자. 그림 19에서 I_{bin}을 보이고 있다.

다양한 슬라브 옆면 영상에 대해 실험해 본 결과, 전체적으로 3.4.1절의 알고리즘보다 3.4.2절의 알고리즘의 성능이 우수함이 판명되었다. 이것에 대한 객관적인 기준으로, 다음에 언급할 개별 숫자로의 분할 단계에서 분할 알고리즘에 사용되는 파라메타들을 동일하게 놓고 위의 두 알고리즘을

적용해보았을 때, 어느쪽이 더 분할이 잘 되는 가를 들 수 있다. 그러나 수행 시간은 3.4.1절의 알고리즘이 더 짧으므로, 성능과 수행 시간 중 어느 쪽이 더 중요한 가에 따라 알고리즘의 선택이 달라질 것이다.

3.5 슬라브 옆면 번호의 영역 추출

3.4절에서 얻은 이진화된 슬라브 옆면 영상을 이용하여 슬라브 번호의 BB(bounding box) 를 찾아낸다. BB를 구성하는 요소로는 BB의 왼쪽, 윗 모퉁이의 위치(left, top)와 BB의 폭, 높이 (width, height)가 있다. BB를 찾는 알고리즘은 크게 두가지로 나누어지는데, 첫 번째가 번호의 수직 영역(top, height)을 찾는 알고리즘이고, 두 번째가 번호의 수평 영역(left, width)을 찾는 알고리즘이다.

3.5.1 슬라브 번호의 수직 영역 찾기

본 알고리즘의 바탕이 되는 기본적인 가정은 슬라브 번호의 높이는 크게 변하지 않는다는 것이다.

3.4절에서 구한 I_{bin}을 CRE하여 얻은 CR들을 $C(i)$ $(i=1, \cdots, N)$라고 하자. 또한 $C(i)$를 둘러싸는 BB의 top과 height를 각각 $y(i)$와 $h(i)$라고 하자. 슬라브 번호의 평균적인 높이를 h_{bb}라고 하면 $|h_{bb}-h(i)| \le h_t$를 만족하는 CR들을 찾아낸다(h_t는 파라메타). 그것들을 $C_c(j)$ $(j=1, \cdots, M)$이라고 하자. 그러면 다음의 m_h와 m_y를 구한다.

$$m_h=\frac{1}{M}\sum_{j=1}^{M}h_c(j), \quad m_y=\frac{1}{M}\sum_{j=1}^{M}y_c(j). \tag{16}$$

이 때, $y_c(j)$와 $h_c(j)$는 $C_c(j)$의 top과 height이다. 이제 함수 f를 다음과 같이 정의한다.

$$f(C_c(j)) \stackrel{=}{+} |h_{bb}-h_c(j)|+|m_y-y_c(j)| \\ +|m_h-h_c(j)|. \tag{17}$$

$f(C_c(j))$의 최소값을 갖는 $C_c(j)$의 (top, height)를 (y*, h*)라고 하면 이것이 슬라브 번호의 BB의 (top, height)가 된다.

3.5.2 슬라브 번호의 수평 영역 찾기

본 루틴의 입력으로는 3.5.1절에서 찾은 영역이 된다. 사용되는 기본적인 가정은 번호의 너비가 거의 일정하다는 것이다.

3.5.1절에서 찾은 영역 (I_v)을 CRE하여 얻은 CR들을 $C(i)$ $(i=1, \cdots, N)$이라고 하자. 그리고 $C(i)$의 BB의 (left, top, width, height)를 $(x(i), y(i), w(i), h(i))$라고 하자 I_v와 같은 크기의 2차원 배열 $\bar{I}_v$를 잡고 각 요소를 0으로 초기화한다. $\bar{I}_v$에(left, top, width, height)가 $(x(i), y(i), w(i), h(i))$인, 채워진 상자(filled box)들을 그린다. 상자를 채우는 값 c는

$$c=\begin{cases} 255 & \text{if } w(i)-w_c<0 \\ 255\times\exp\left(-\dfrac{(w(i)-w_c)^2}{2\gamma^2}\right) & \text{otherwise} \end{cases} \tag{18}$$

로 주어진다. 이 때 $w_c=kh^*$이다(γ와 k는 파라메타). 이제 $\bar{I}_v$를 수직 투영하여 프로파일을 얻은 후 가우시안 필터링한다. 그 프로파일의 최대값이 위치하는 곳을 i_{max}라고 하면, 슬라브 번호의 BB의 (left,width)는

$$(x^*, w^*)=(i_{max}-w_{bb}/2, w_{bb}) \tag{19}$$

가 된다. 이 때 w_{bb}는 BB의 평균적인 width에 일정량의 마진이 더해진 값이다.

그림 20에서 슬라브 번호의 BB를 찾은 모습을 보이고 있다.

그림 20. 슬라브 번호 영역을 찾은 모습.

3.6 슬라브 옆면 번호의 개별 숫자로의 분할 및 크기의 정규화

3.6.1 슬라브 번호의 개별 숫자로의 분할

3.5절에서 찾은 슬라브 번호는 인식되기에 앞서서 개별 숫자로 분할되어야 한다. 분할 위치는 I_{bin}에서 (left, top, width, height) $=(x^*, y^*, w^*, h^*)$에 의해 둘러싸인 영역(I_{bb})을 이용하여 찾는다. 그런 후, 원래의 명도 영상 (I)으로부터 분할 정보에 따라 개별 숫자로 분리해낸 후, 크기를 정규화한 다음 인식기에 입력한다.

개별 숫자로의 분할 알고리즘에서 가장 기본이 되는 가정은 개별 숫자는 일정한 높이 : 폭의 비율을 갖는다는 것이다. 높이는 BB를 찾을 때 구한다 (h^*에 해당된다). 그러면 개별 숫자의 폭의 추정치는 $w_e=kh^*$로 주어진다. 일반적으로 높이 > 폭이므로 $0<k<1$이 된다.

알고리즘을 단계별로 밝히면 다음과 같다.

(1) I_{bb}를 3×1의 SE로 dilation하여 수평 성분을 강조한다. 그 결과를 $\bar{I}_{bb}$라고 하자. 이 단계는 가는 폰트에만 적용되고 굵은 폰트에는 적용되지 않는다. 왜냐하면 가는 폰트의 경우 수직 투영을 했을 때, 주위의 수직 성분에 의해, 나타나야할 수평 성분의 영향이 나타나지 않는 경우가 있기 때문이다.

(2) $\bar{I}_{bb}$를 수직 투영하여 얻은 프로파일을 가우시안 필터링한 후, 이진화하여 일정량 이상 되는 공백부분에서 숫자들을 분할한다. 분할된 숫자 영상들을 $I_s(i)$ $(i=1, \cdots, N)$라고 하자.

(3) $I_s(i)$를 수직 투영하여 프로파일을 얻은 후, 일정한 값 t_s 잡아 그것보다 큰 값을 갖으면 t_s로 놓는다. 그런 후에 가우시안 필터링을 하여 얻은 프로파일을 $V_s(i)$라고 하자.

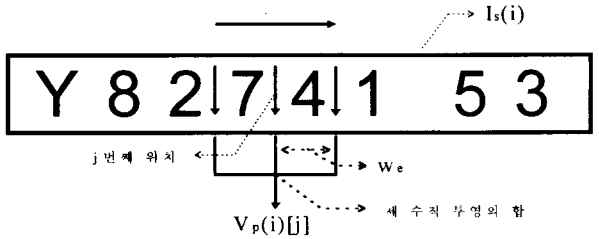


그림 21. $V_p(i)$를 구하는 모습.

(4) $I_s(i)$를 그림 21과 같이 투영하여 얻은 프로파일을 $V_p(i)$라고 하자. 일정한 값 t_m을 잡아 $V_p(i)$의 요소들 중 t_m보다 큰 값을 갖는 것은 t_m으로 놓는다. 그런 후에 가우시안 필터링을 하여 얻은 프로파일을 $V_m(i)$라고 하자.

(5) $V_s(i)$와 $V_m(i)$의 극소점들을 찾는다. 이 때 미미한 극소점들은 제거된다. 그러면 $V_m(i)$의 극소점들의 위치가 숫자 사이의 경계 후보가 되며, $V_s(i)$의 극소점들의 위치는 보조적인 도구로써, 근처에 $V_m(i)$의 극소점이 안 나타나는 경우 숫자 사이의 경계 후보가 된다.

(6) 최종적인 경계를 정하기 위해 몇가지 후처리가 따른다.

위에서 제시한 알고리즘을 이용하여 처리한 결과를 그림 22에서 보이고 있다. 그림 22에 나타나 있는 $V_s(i)$와 $V_m(i)$를 비교해 보면 $V_m(i)$의 유용성을 알 수 있다. 숫자열 Y8274153에서 숫자 4의 경우 수평획이 모두 훼손되어 있다. 따라서 $V_s(i)$의 경우에는 극소점이 4의 중간에 위치하고 있다. 그러나 $V_m(i)$의 경우에는 제대로 위치하고 있다.

$V_m(i)$을 생성하는데 사용되는 기본적인 가정은 현재의 수직 투영 위치가 j이고, 그곳이 이웃하는 숫자들 사이의 경계 중의 한 곳이라면 $j-w_e$와 $j+w_e$도 숫자들 사이의 경계일 가능성이 크다는 것이다. 그리고 이러한 가정은 숫자들의 너비가 거의 일정하고 w_e가 제대로 추정되었다는 가정에서 시작한다.

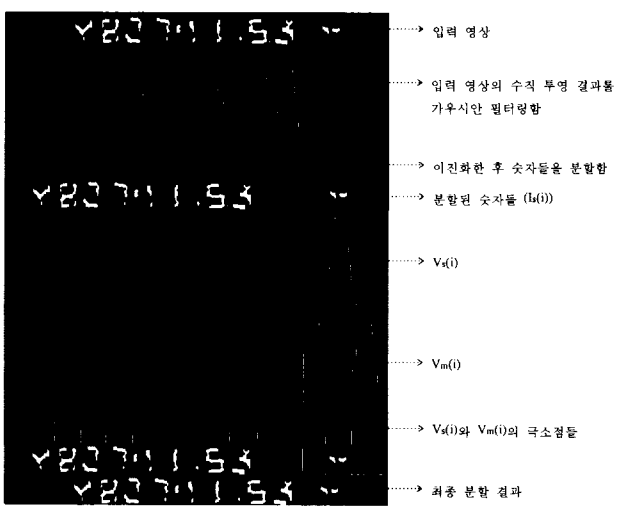


그림 22. 슬라브 변화를 개별 숫자로 분할하는 모습.

위의 예에서 7의 왼쪽 경계와 4의 오른쪽 경계에는 다행이 노이즈가 없다. 따라서 $V_m(i)$에서 4의 중간 부분보다는 7과 4의 경계에서 극소값을 갖게 되는 것이다.

만약 배경의 노이즈와 숫자의 훼손이 복합적을 존재하는 경우에는 본 알고리즘만으로 제대로 분할하기에는 역부족이다. 최근에 발표된 접촉되어 있는 필기체 문자들의 분리 연구 결과들을 보면 인식을 분리의 검증 수단으로 이용하는 경향이 늘고 있다 [6]. 본 인식 시스템에서는 결국 재분할 단계에서 이 방법을 채용한 셈이 된다.

3.6.2 개별 숫자들의 크기 정규화

3.6.1절에서 제시한 분할 알고리즘에 의해 얻어진 i번째 숫자의 이진 영상을 $I_n(i)$라고 하자. 최종적인 목적은 가는 폰트의 경우에는 각 숫자별로 16×16의 명도 영상을 얻는 것이다.

$I_n(i)$에는 숫자가 아닌 것들도 포함되어 있을 수 있다. 그림 22에서 5의 왼쪽이나 3의 오른쪽에 있는 노이즈가 그런 예에 속한다. $I_n(i)$에 속한 흰 픽셀들의 수를 세어보았을 때, 그 수가 상당히 적으면 그것은 숫자가 아니라고 볼 수 있다. 따라서 이런 것들은 쉽게 제거될 수 있다. 5의 왼쪽에 있는 노이즈가 그런 예에 속한다. 그러나 노이즈의 경우에도 픽셀 수가 많은 경우가 있다. 그런 경우에는 판단을 인식기에 맡긴다. 3의 오른쪽에 있는 노이즈가 그런 예이다.

숫자 좌우에 남아 있는 마진을 없애기 위해 $I_n(i)$의 수직 투영 결과를 이용한다. 이렇게 해서 개별 숫자의 최종적인 영역이 결정된다.

16×16의 명도 영상을 만들기에 앞서 32×32의 명도 영상을 만든다. 우선 32×32의 이차원 배열을 잡는다. 개별 숫자들은 대부분 32×32보다 작다. 입력 명도 영상으로부터 앞에서 찾은 개별 숫자 영역에 속하는 픽셀들을 32×32 배열의 중심에 오도록 복사한다. 남은 곳들은 개별 숫자의 명도 영상 주위의 픽셀들의 평균값으로 채운다. 마지막으로 16×16으로 축소한다. 그림 23에서 개별 숫자들의 크기를 정규화한 예를 보이고 있다.

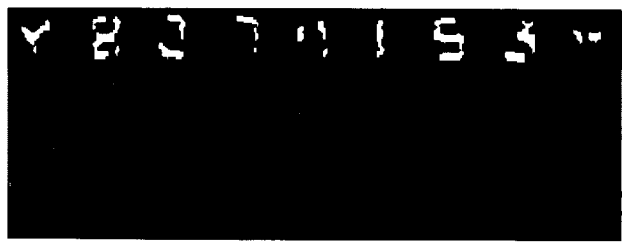


그림 23. 개별 숫자들의 크기를 정규화한 모습.

3.7 개별 숫자들의 인식

본 시스템에서 채용하고 있는 인식기는 2.2절에서 밝혔듯이 KLT를 이용한 부분공간 분류기이다. 이 방법은 Watana-

be에 의해 처음으로 도입되었다[7]. 그러나 명도 영상을 직접적인 입력으로 사용할 때 효력이 있음이 밝혀진 것은 Pentland 등이 얼굴 인식에 적용하면서부터라고 말할 수 있다[8][9]. 얼굴 인식 분야에서는 그 이후에 활발하게 적용되고 있다[10]. 최근에는 3차원 물체의 학습 및 인식에도 이용되고 있다[11]. 또한 슬라브 번호 인식과 유사한 환경인 화차 번호 인식에서도 성공적으로 적용되었다 [12].

일반적으로 인식기를 설계할 때 신중을 기해야 하는 문제 중에 하나가 특징의 선택 (feature selection) 문제이다. 원 입력으로부터 특징을 추출하는 것은 결국 분할하고자 하는 공간의 차원을 줄이는 것과 같은데, 이 때 클래스간의 가분성(separability)이 최대한 유지되는 특징들을 선택해야 한다. 사용되는 특징은 인식하고자 하는 대상에 따라 달라지기 때문에 한마디로 개발자의 노우하우라고 말할 수 있다.

KLT를 이용한 인식기를 사용하는데 있어서 가장 편리한 점은 개발자가 특징을 선택해 줄 필요가 없다는 것이다. 왜냐하면 샘플들이 주어지면 통계적인 방법에 의해 그것이 결정되기 때문이다. 차원을 줄이는데 사용되는 선형변환 중에서 KLT가 최적의 성능을 보임은 수학적으로 증명이 되어 있다 [13].

KLT를 이용한 인식 방법은 대부분의 인식기가 그렇듯이 훈련(training) 단계와 분류(classification) 단계로 나눌 수 있다. 고려하는 클래스의 수는 모두 L개라고 가정하자. i번째 클래스에 속하는 N차원의 랜덤 샘플 벡터(random sample vector)를 $\mathbf{x}(i)$라고 하자. 그러면, 훈련 단계에서는 각 클래스마다 다음의 식들을 계산한다.

$$\mathbf{m}(i)=E[\mathbf{x}(i)], \tag{20}$$

$$\mathbf{C}(i)=E[(\mathbf{x}(i)-\mathbf{m}(i))(\mathbf{x}(i)-\mathbf{m}(i))^T], \tag{21}$$

$$\mathbf{C}(i)\mathbf{e}(i,\ j)=\lambda(i,\ j)\mathbf{e}(i,\ j),\ (\mathbf{e}(i,\ j))^T\ \mathbf{e}(i,\ j)=1,\ \lambda(i,\ j)\geq\lambda(i,\ j+1),\ j=1,\cdots,N \tag{22}$$

$$\mathbf{E}(i)=[\mathbf{e}(i,\ 1)\ \mathbf{e}(i,\ 2)\cdots\mathbf{e}(i,\ M)],\ M<N. \tag{23}$$

여기에서 E는 기대값 연산자(expectation operator)이고, $\lambda(i,\ j)$는 공분산 행렬 $\mathbf{C}(i)$의 j번째 고유값(eigenvalue)이고 $\mathbf{e}(i,\ j)$는 그것에 대응되는 고유벡터(eigenvector)이다. 고유값들은 내림차순으로 정렬이 되어 있다. 최종적으로 구하고자 하는 것은 변환 행렬 $\mathbf{E}(i)$인데, 이 행렬에 의한 변환을 truncated KLT라고 한다. 이 때 M은 변환 후 차원의 크기이다.

분류 단계에서 새로운 패턴 $\mathbf{x}$가 입력되면 다음의 규칙에 의해 i^*번째 클래스로 분류된다.

$$\mathbf{y}(i)=(\mathbf{E}(i))^T(\mathbf{x}-\mathbf{m}(i)), \tag{24}$$

$$\mathbf{x}(i)=\mathbf{E}(i)\mathbf{y}(i)+\mathbf{m}(i), \tag{25}$$

$$\epsilon^2(i)=\|\mathbf{x}-\mathbf{x}(i)\|^2, \tag{26}$$

$$i^*=\arg\ \min_i\ \epsilon^2(i). \tag{27}$$

여기에서 $\mathbf{x}(i)$는 복원 벡터이고 $\epsilon(i)$는 복원 오차(reconstruction error)이다.

본 인식 시스템에서는 가는 폰트 숫자의 경우 16×16의 명도 영상으로부터 row scan order로 256×1의 벡터를 형성한 후, 인식기의 입력으로 사용한다. 클래수의 수는 12개로 0~9, Y, 공백문자로 구성되어 있다.

3.8 혼동되는 숫자들의 검증

2.2절에서 언급하였듯이 숫자의 인식 결과와 사전 정보가 다른 경우, 그것이 미리 지정되어 있는 혼동 클래스들 중의 하나이면, 그 혼동 클래스 내의 숫자들을 구분해낼 수 있는 루틴이 작동하고, 없다면 분할 위치에 오류가 있는 것으로 보고 재분할 루틴이 작동하게 된다.

혼동 클래스는 형태의 유사성에 의해 인식기가 통계적으로 자주 혼동을 일으키는 패턴들의 집합이다. 본 인식 시스템에서는 특히 가는 폰트의 경우가 굵은 폰트의 경우에 비해 혼동되는 숫자들이 많이 발생하였다. 가는 폰트의 경우 혼동되는 숫자들의 부류는 (1 ↔ 7), (9 → 4, 7), (0 → 4, 5, 7)이다. 이런 숫자들을 구분하기 위해 국부적인 정보를 추출하는데, 혼동 클래스 내의 숫자 특성에 따라 수평/수직 에지를 구한 후 수평/수직 투영을 하여 그 결과를 이용한다. 예를 들면,


: 9를 4로 혼동한 경우로 수평 에지를 구한 후 상단을 수평 투영하여 그 결과를 이용한다.

3.9 피드백의 조건

그림 3에서 피드백이 있는 곳에 번호가 부여되어 있다. 그 조건을 명시하면 다음과 같다. 편의상 사전 정보에 의한 슬라브 번호의 문자열을 S, 인식 장치에 의한 슬라브 번호의 문자열을 R, S의 길이를 n, R의 길이를 m이라고 하자. ① 이 동작할 조건은 S≠R이고 $|n-m|\leq 3$인 경우이다. ②가 동작할 조건은 $|n-m|>3$인 경우인데, 다음으로 가능성이 있는 슬라브 영상이 선택된다. ③이 동작할 조건은 ②의 피드백이 실패한 경우로써 슬라브 영상은 원래의 것이 되고 폰트의 종류가 바뀐다. 마지막으로 ④가 동작할 조건은 ①, ②, ③의 피드백이 일정한 횟수 이상 시도되었으나 모두 실패한 경우이다.

3.10 슬라브 번호의 개별 숫자들로의 재분할

3.9절의 피드백 조건에서 ①의 조건이 발생하면 사전 정보를 참조하여 개별 숫자로 잘못 분할된 것들을 시정하는 것이 본 루틴의 역할이다.

우선 숫자의 나열이 특정한 형태를 가지고 있고, 개별 숫

자의 너비와 숫자들간의 간격이 거의 일정하다는 가정을 한다. 여기서의 가정은 주어진 숫자 영상에 한정된다. 즉 모든 숫자 영상에 대해서 똑같은 너비와 간격이 적용되는 것은 아니다.

위의 가정을 통해서 인식기에 의한 번호의 문자열과 사전 정보에 의한 번호의 문자열을 비교(string matching)하여 인식이 잘못된 영역 또는 분실된 개별 숫자 영역을 예측하고, 이렇게 예측한 영역을 이진화하여 숫자 영역을 추출하면 더욱 신뢰성이 있는 개별 숫자의 분할을 이룩할 수 있다. 알고리즘을 단계별로 밝히면 다음과 같다.

(1) 인식기에 의한 번호의 문자열과 사전 정보에 의한 번호의 문자열을 비교해서 서로 같은 부분을 연결한다.

(2) 두 문자열의 연결을 살펴서 최초의 분할에서 찾지 못한 부분이나 분할 위치가 잘못된 부분들을 알아 낸다.

(3) (2)에서 알아낸 부분 주위에서 이진화를 하여 좀 더 세밀하게 개별 숫자 영역을 찾아낸다.

(4) 인식을 하고 사전 정보와 다시 비교해본다.

(5) 비교해본 결과 불일치 시에는 너비를 변화시키면서 (1)~(4)를 재실행하고 일치 시에는 피드백을 멈춘다. 단 여기서 너비의 변화는 일정한 범위에 한정한다.

그림 24에서 재분할 알고리즘을 적용한 예를 보이고 있다.

그림 24-1. 재분할 전의 모습.

그림 24-2. 재분할 후의 모습.

4. 실험 및 결과

현재까지 개발되어 있는 인식 시스템의 성능은 다음과 같다.

슬라브 앞면의 경우에는 번호의 영역 추출 실험까지 하였다. 테스트에 사용된 슬라브 앞면 영상은 총 427장으로 378장이 영역을 올바르게 찾아서 성공률은 88.5%가 나왔다. 영역을 올바르게 찾지 못한 경우를 세분하면 표 1과 같다.

표 1. 슬라브 앞면 번호 영역 찾기 결과.

전체 슬라브 앞면 영상의 수	427
올바로 찾은 경우	378(88.5%)
숫자이외의 영역을 포함한 경우	40(9.4%)
숫자의 일부가 잘린 경우	7(1.6%)
하나 이상의 숫자를 못 찾은 경우	2(0.4%)

표 2. 3장의 옆면 영상 중 번호를 포함한 영상 찾기.

	Class A	Class B	Class C	Total
영상 찾기	797/808 (98.6%)	403/421 (95.7%)	265/278 (95.3%)	1465/1507 (97.5%)

표 3. 가는 폰트 실험 결과.

	Class A	Class B	Class C	Total
개별 숫자 분할	130/135 (96.3%)	220/251 (87.6%)	120/178 (67.4%)	470/564 (83.3%)
인 식	119/135 (88.1%)	176/251 (70.1%)	74/178 (41.6%)	369/564 (65.4%)
재분할/ 검증	135/135 (100%)	223/251 (88.8%)	114/178 (64.0%)	472/564 (83.7%)

표 4. 굵은 폰트 실험 결과.

	Class A	Class B	Class C	Total
개별 숫자 분할	665/673 (98.8%)	158/170 (92.9%)	88/100 (88.0%)	911/943 (96.6%)
인 식	650/673 (96.6%)	157/170 (92.4%)	58/100 (58.0%)	865/943 (91.7%)
재분할/ 검증	668/673 (99.3%)	164/170 (95.9%)	89/100 (89.0%)	920/943 (97.6%)

표 5. 전체 실험 결과.

	Class A	Class B	Class C	Total
개별 숫자 분할	795/808 (98.4%)	378/421 (89.8%)	208/278 (74.8%)	1381/1507 (91.6%)
인 식	769/808 (95.2%)	333/421 (79.1%)	132/278 (47.5%)	1234/1507 (81.9%)
재분할/ 검증	803/808 (99.4%)	386/421 (91.7%)	203/278 (73.0%)	1392/1507 (92.4%)

슬라브 옆면의 경우에는 영상의 상태에 따라 세가지의 클래스로 나눈 후 각 클래스 별로 실험 결과를 내었다.

숫자가 대체로 선명하고 배경의 노이즈가 적은 경우를 클래스 A로 분류하였고, 숫자의 훼손이나 배경의 노이즈가 심하거나 숫자 부분과 배경 부분의 밝기 차가 거의 나지 않는 경우 클래스 C로 분류하였다. 클래스 B는 클래스 A와 C의 중간정도의 상태를 갖는다. 각 클래스별 영상의 예를 그림 25에서 보이고 있다.

옆면 이미지의 각 클래스별 실험 결과는 표 2, 3, 4, 5와 같다. 표 2는 3장의 옆면 영상 중 슬라브 번호가 존재하는 영상을 찾는 실험으로 총 1507장에서 1465장을 올바르게 찾아서 97.5%의 성공률을 보였다. 이 실험 결과는 다음에 이어지는 실험과는 독립적으로 행하여졌다.

표 3은 가는 폰트로 인쇄된 번호에 대한 실험 결과로써 개별 숫자로의 분할, 번호의 인식, 재분할 및 검증으로 세분하여 산출하였다. 실험 결과로는 564장 중 472장을 제대로 인식하여 83.7%의 인식률을 보였다.

표 4는 굵은 폰트로 인쇄된 번호에 대한 실험 결과로써 가는 폰트와 마찬가지로 실험 결과를 세분했으며 단 가는 폰트와는 달리 검증 부분이 없다. 실험 결과로는 943장 중 920장을 제대로 인식하여 97.6%의 인식률을 보였다.

표 5는 가는 폰트와 굵은 폰트의 실험 결과를 합친, 옆면 슬라브 번호 인식의 전체 실험 결과를 나타낸다. 최종 실험 결과는 1507장 중 1392장을 제대로 인식하여 92.4%의 인

그림 25-1. 클래스 A에 속하는 영상의 예.

그림 25-2. 클래스 B에 속하는 영상의 예.

그림 25-3. 클래스 C에 속하는 영상의 예.

식률을 보였다.

위의 인식 결과는 개별 숫자 단위가 아니고 슬라브 단위임에 주의할 필요가 있다. 따라서 인식의 경우, 번호 중 한 숫자라도 인식이 잘못되면 오인식으로 분류된다.

본 실험 결과를 분석해보면 KLT를 이용한 인식기가 굵은 폰트 번호를 인식하는데는 충분하지만 가는 폰트 번호를 인식하는데는 미약한 점이 있다는 것을 알 수 있다. 그 이유는 가는 폰트 번호의 경우 변화가 훨씬 다양하기 때문이다. 따라서 인식부의 보강이 필요하다.

5. 결론 및 향후 개발 방향

본 논문에서는 산업 현장에서 활용될 수 있는 문자 인식 시스템의 예로써 슬라브에 기록되어 있는 번호를 인식하는 시스템을 다루었다. 특히 기존의 OCR(optical character recognition) 시스템에서 채용되는 알고리즘들과는 차별되는, 노이즈에 강건한 알고리즘들을 제시하였다.

산업 현장에는 슬라브 번호와 같이 여러가지 노이즈가 존재하는 문자들을 인식하는 시스템을 필요로 하는 곳이 많으리라 여겨진다. 따라서 이러한 상황에 강한 알고리즘들이 많이 개발되어야 할 것이다.

끝으로 향후 개발 방향을 밝히면 다음과 같다. 우선 하드웨어 측면에서는 현재 옆면에 설치되어 있는 에어리어(area) 카메라들을 라인(line) 카메라로 대체할 예정이다. 그럼으로써 카메라들 사이에 생기는 시야의 중첩 문제라든가 광각 렌즈의 사용으로 발생하는 문제점들이 해결되리라 여겨진다.

소프트웨어 측면에서는 모든 부분에 있어서 계속적인 향상이 있어야겠지만 특히 인식부의 취약점들 - 공백 문자와 숫자의 구분, 혼동되는 숫자들의 검증, 보다 지능적인 재분할 - 이 보강되어야 할 것이다.

참고문헌

[1] Philippe S. M. and J. S. Chen, "Adaptive smoothing : a general tool for early vision," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 13, no. 6, pp. 514-529, June, 1991.

[2] Otsu N., "A threshold selection method from gray-level histograms," IEEE Trans. on Sytems, Man, and Cybernetics, vol. 9, pp. 62-66, 1979.

[3] H. S. Baird, H. Bunke, and K. Yamamoto, Structured Document Image Analysis, Springer-Verlag, pp. 99-114, 1992.

[4] ϕ. D. Trier and A. K. Jain, "Goal-Directed evaluation of binarization methods," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 17, no. 12, pp. 1191-1201 December, 1995.

[5] R. M. Haralick and L. G. Shapiro, Computer and Robot Vision, vol. 1, Addison-Wesley Publishing Co., pp. 392-403, 1992.

[6] R. G. Casey and E. Lecolinet, "Strategies in character segmentation : A survey," in Proc. of International Conference on Document Analysis and Recognition, pp. 1028-1033, 1995.

[7] E. Oja, Subspace Methods of Pattern Recognition, Research Studies Press LTD., pp. 73-108, 1983.

[8] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 586-591, 1991.

[9] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 84-91, 1994

[10] R. Chellappa and S. Sirohey, Human and Machine Recognition of Faces : A Survey, Tech. Report, CAR-TR-731, Univ. of Maryland, 1994.

[11] H. Murase, "Visual learning and recognition of 3-D object from appearance," International Journal of Computer Vision, 14, pp. 5-24, 1995.

[12] 구 정회, 장 정훈, 홍 기상, "Karhunen-Loève 변환을 이용한 옥외환경에서의 화차번호 인식 시스템 설계," 제 7회 영상처리 및 이해에 관한 워크샵, pp. 109-115, 1995.

[13] V. R. Algazi, and D. J. Sakrison, "On the optimality of the Karhunen-Lo ve expansion," IEEE Trans. on Information Theory, vol. 15, no. 2, pp. 319-321, March, 1969.

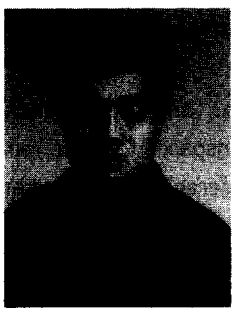
저 자 소 개

홍 기 상

1977년 서울대학교 전자공학과 졸업(공학사)
1979년 한국과학기술원 전기및전자공학과 졸업(공학석사)
1984년 한국과학기술원 전기및전자공학과 졸업(공학박사)
1984년~86년 한국 에너지 연구소 선임연구원
1988~89년 Carnegie Mellon University 교환교수
1991년~현재 포항공과대학교 전자전기공학과 부교수

양 종 렬

1993년 경북대학교 컴퓨터공학과 졸업(공학사)
1995년 경북대학교 대학원 컴퓨터공학과 졸업(공학석사)
1995년~현재 포항공과대학교 첨단공학연구소 연구원

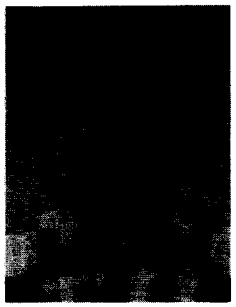


김 태 원

1996년 포항공과대학교 전자전기공학과 졸업(공학사)
1996년~현재 포항공과대학교 대학원 석사과정

장 정 훈

1994년 한양대학교 전기공학과 졸업(공학사)
1996년 포항공과대학교 대학원 전자전기공학과 졸업(공학석사)
1996년~현재 포항공과대학교 대학원 전자전기공학과 박사과정



김 승 진

1996년 포항공과대학교 전자전기공학과 졸업(공학사)
1996년~현재 포항공과대학교 대학원 석사과정