

# 한글 음절 내의 자음, 모음 위치 상관관계 기반 출력물 상하 반전 여부 탐지 기법 연구

## A Study on Detection Algorithm for Upside Down Printed Materials based on Consonant, Vowel's Position Correlation

김인제  
컴퓨터공학과

Injae Kim, Dept. of CSE, Dongguk Univ.

### 요 약

문서 스캔 기술과 장비의 발전에도 불구하고 수많은 스캔 과정은 자동화되지 못하고 있다. 예를 들어 많은 스캔본 들은 위아래가 뒤집힌 채로 스캔 되는데, 스캔본의 위아래가 뒤집혔는지 사람이 일일이 검사하고 있다.

본 연구에서는 음절 내의 자음, 모음의 위치 상관관계를 기반으로 하는 알고리즘을 제안하며 실험 결과 스캔본의 위아래가 뒤집혔는지에 대하여 95% 이상의 정확도를 얻었다.

**주제어** : 스캔본 상하 반전, 자음, 모음, 음절, 위치 상관관계

### Abstract

Even though modern scan technology and equipment is highly developed and widely spread that now we can scan book only using smartphone, many printed matters aren't automated. For instance, many printed materials are often upside down. To check whether scanned matters is upside down or not, matters should be checked by humans.

By using detection algorithm that checks position correlation between consonant and vowel in syllable, checking whether scanned matters is upside down by humans can be fully automated. Result of using this algorithm, We got over 95% accuracy for whether scanned matters is upside down

This study will suggest an algorithm that checks scanned matter's direction automatically and experiment result.

**Key Words** : Upside down detection, consonant, vowel, position correlation

## I. 서 론

책이나 문서에 대한 디지털화 는 현재 스캔 기술 발전에 힘입어 활발히 진행 중 이다.

하지만 문서를 스캔하여 디지털 화 하는 과정에서 문서가 상하 반전 된 채로 스캔되는 경우가 많은데 이는 스캔 과정이 완전히 자동화 된 것이 아닌 사람이 진행하기 때문이다. 따라서 현재 스캔 결과물의 상하 반전 여부를 일일이 사람이 확인하고 있다.

본 연구에서는 한글의 언어학적 특징인 음절 내의 자음, 모음의 위치 관계에 기반하여 스캔 결과물에 대한 상하 반전 여부를 탐지 할 수 있는 방법을 제시 한다.

## II. 한글 음절 내의 자음, 모음 위치 상관관계 기반 스캔본 상하 반전 탐지 기법

### 1. 방법론

현대 한글은 음절 로 이루어져 있다. 한글의 음절 이란 한글 글자마다의 첫소리와 가운데 소리 글자, 또는 첫소리, 가운데소리, 끝소리 글자로 이루어진 한글의 단위를 말한다.





Figure 4. 자음 + 모음 만으로 이루어진 음절과 상하 반전 시의 예시

Figure 4 에서 자음 + 모음 만으로 이루어진 음절들과 그 음절을 상하 반전 하였을 때의 예시를 살펴볼 수 있다. '가, 가, '거', 겨' 와 같은 경우 자음이 모음 왼쪽에, 모음이 자음의 오른쪽에 위치하였지만 상하 반전 시 모음이 자음의 왼쪽에, 자음이 모음의 오른쪽에 위치하는 것을 알 수 있다.

또한 '고', '교', '구', '규', '그', '기' 와 같은 경우 자음이 모음 위쪽에, 모음이 자음 아래쪽에 위치하였지만 상하 반전 시 모음이 자음의 위쪽에, 자음이 모음의 아래쪽에 위치하는 것을 알 수 있다.

따라서 자음 + 모음 으로만 구성된 음절들의 이와 같은 상하 반전 시 자음, 모음의 위치 상관관계가 역전되는 특징을 이용해 문서의 상하 반전 여부를 탐지할 수 있다.

## 2. 구현

첫 번째로, 스캔본을 OpenCV를 사용하여 불러온 후 스캔본의 문자 후보 영역을 검출하기 위해 모폴로지 처리 중 닫힘 연산과 경계선 인식을 통해 문자 후보 영역을 검출한다.

Fig. 5, 6 의 과정에서 그 결과를 확인할 수 있다.

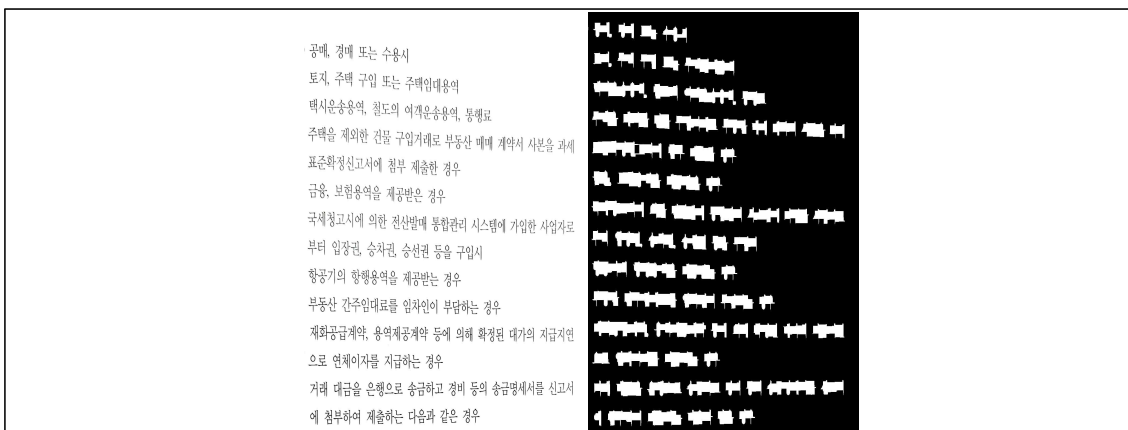


Figure 5. 모폴로지 처리 중 닫힘 연산 결과

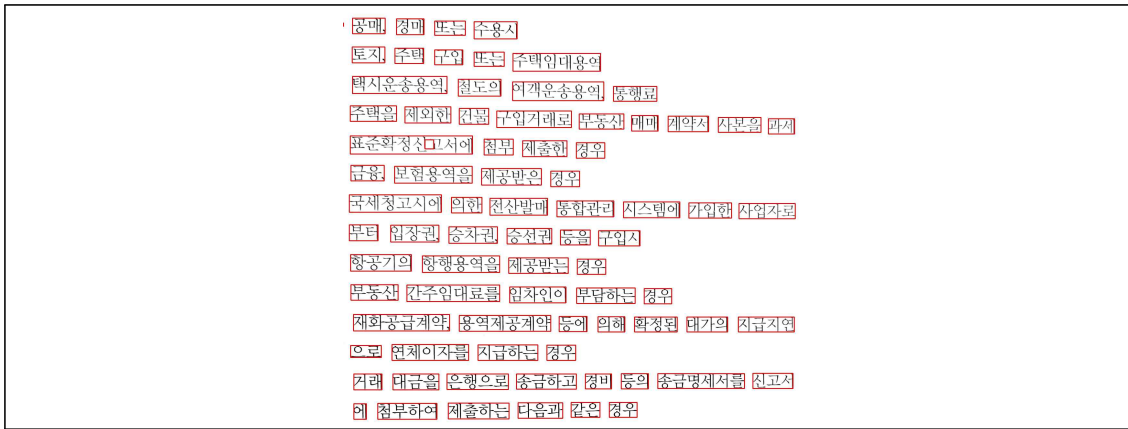


Figure 6. 단합 연산 후 경계선 검출을 통해 문서 내의 문자 후보 영역 검출 결과

그 후 검출된 문자 후보 영역을 음절 단위로 한 글자씩 분리하는 과정을 거치게 된다.

이때 문자 후보 영역의 모든 열 에 대하여 그 열의 픽셀값이 모두 0 이라면 그 열을 기준으로 문자 후보 영역을 분리하게 된다. Fig. 7 의 빨간색 세로선이 문자 후보 영역에서 음절을 분리한 결과를 나타낸다.



Figure 7. 문자 후보 영역에서 음절을 기준으로 글자 분리

문자 후보 영역에서 한 글자씩 음절을 분리한 후에는 그 음절 안에 자음, 모음이 몇 개씩 위치해 있는지를 탐지하게 된다. 이때 음절 안에서 경계선을 따라 bounding box를 추출한 후 bounding box의 가로, 세로 의 비율이 2.5 이상 일 때 (l, ll, ㅌ 의 경우 세로가 가로보다 더 길다) 와 0.4 이하일 때 (ㅡ, ㅍ, ㅠ 의 경우 가로가 세로보다 더 길다) 의 bounding box는 모음 이라고 판단하며 이 밖의 경우에는 자음 이라고 판단한다.

Fig. 8, 9는 자음은 파란색으로, 모음은 빨간색 bounding box로 표시한 결과 이다.

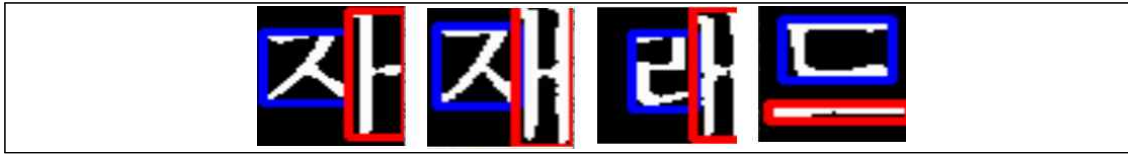


Figure 8. 정 방향 문서에서 검출된 자음 + 모음 으로 구성된 음절



Figure 9. 역 방향 문서에서 검출된 자음 + 모음 으로 구성된 음절

마지막으로 Fig 8, 9 와 같이 한 음절에서 자음 + 모음, 즉 자음 1개와 모음 1개 만으로 한 음절이 구성되었을 경우에만 자음, 모음의 bounding box 에서 x, y 좌표를 비교하여 자음과 모음의 위치 상관관계를 추출한 후, 자음이 모음 왼쪽이거나 위 쪽에 위치할 경우 정방향, 자음이 모음 오른쪽이거나 아래쪽에 위치할 경우 역방향 카운터에 +1 을 해준다.

문서에서 검출된 모든 문자 후보 영역에 대해서 위와 같은 과정을 반복한 후 정방향, 역방향 카운터의 크기를 비교하여 큰 것의 방향을 현재 문서의 방향으로 채택한다.

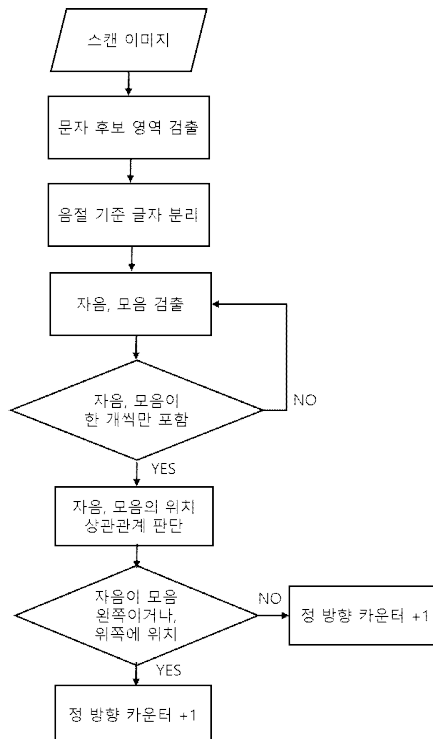


Figure 10. 알고리즘 순서도

제안한 알고리즘의 전체 순서도는 Fig. 10 와 같다.

### III. 실험

#### 1. 실험 환경

총 20개의 문서 스캔본 에 대하여 상하 반전 여부 탐지 실험을 진행하였다. Ubuntu 16.04, OpenCV 4.1.1, Intel Core i7-8565U 1.80GHZ CPU, 16.0GB 메모리를 탑재한 랩탑에 실험을 진행하였다.

실험에 사용된 문서 스캔본은 압축을 진행하지 않은 고해상도의 스캔 원본을 사용하였으며 정방향 스캔본 30장, 역방향 스캔본 30장, 총 60장으로 구성되어 있으며 일반적인 스캔 과정에서 발생하는 노이즈, 문서의 휨, 그림, 수식, 표 등을 포함하고 있다.

#### 2. 실험 결과

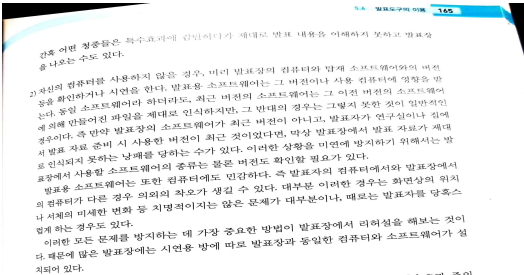
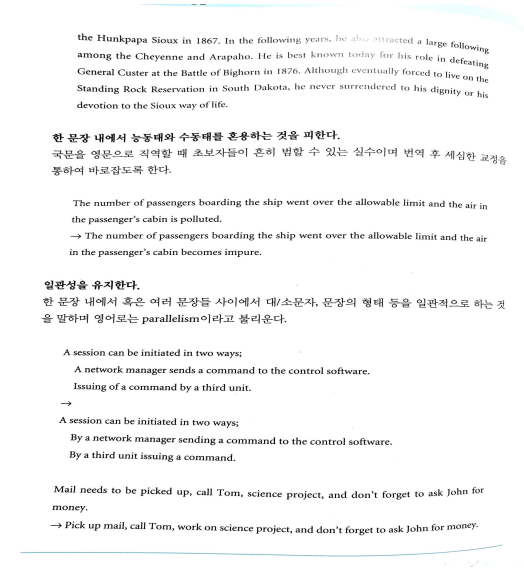
Table 1. 실험 결과

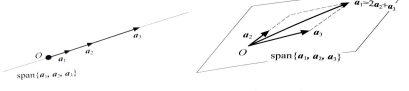
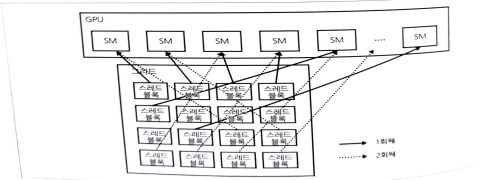

항목	내용
테스트 케이스	60 장
정확도	96.6 %
문서 1장당 처리소요 시간 (평균)	1.80 초

실험 결과 스캔본 상하 반전 여부 판단 정확도는 96.6%로 전체 테스트 케이스 60장 중 58장의 상하 반전 여부를 정확하게 판단할 수 있었다. 이는 하나의 문서 안에 음절이 평균 500개 이상 포함되어 이에 대한 상하 반전 여부 판단을 여러 번 진행할 수 있으므로 좀 더 정확한 판단을 할 수 있기 때문이다.

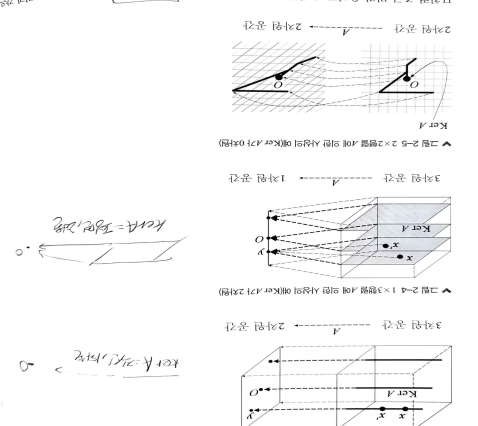
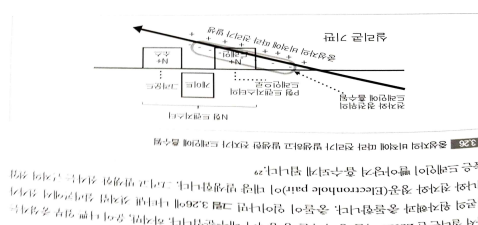
테스트 케이스에 대한 상세 결과는 Table. 2 와 같다.

Table 2. 실험 상세 결과

입력 이미지	정방향 인식 음절 수	역방향 인식 음절 수	최종 판단	수행 시간	이미지 특징
<p>위에서는 박테리아가 지하 깊은 곳에서 식식한다는 논의를 제시하기에 앞서 이와 대립되는 생물학자들의 견해를 간단히 제시한 다음, 반박하며 글을 시작하고 있다. 자신의 견해를 일반적으로 주장하는 것보다 자신의 견해에 대립되는 견해나 주장에 대한 비판적 입장을 제시하며 비교 우위를 드러내는 방식으로 시작하는 것도 하나의 효과적인 방법이 된다.</p> <p><b>(4) 질문과 대답으로 시작하기</b> 과제의 관련된 내용에 대한 질문을 내리고 대답하며 시작하는 방법이다. 내용에 대한 독자의 주의를 끌기시킬 수 있을 뿐만 아니라 앞으로 전개될 내용에 대해 독자가 미리 생각해 볼 여유를 줄 수도 있다.</p> <p>오존층이란 무엇일까 오존층이란 지상 약 15 km에서 약 40 km 사이의 고도에 오존이 밀집되어 있는 층을 말한다. 오존층의 발견은 폴란드 과학자 찰스 데이비슨이 1911년 남극에서 관측된 태양스펙트럼을 조사하는 가운데 나타난 의문의 하나는 파장이 약 300 nm 이하가 되는 자외선 영역의 복사선이 관측되지 않는 것이었다. (김정렬, 『노벨상과 함께 하는 지구여행』, 한국과학기술원출판원 출판부, 2007.)</p> <p>위는 '오존층의 발견'이란 소재물이 붙은 글이다. 글을 시작하면서 오존층이란 무엇인지에 대해 묻고 대답하는 방식으로서 서두를 시작하고 있다. 여기에는 오존층의 개념에 대한 정의와 더불어는 방식도 함께 사용되고 있다.</p> <p><b>(5) 비유로 시작하기</b> 비유는 대상에 대한 이해도를 높이는 데 효과적인 방법이다. 아무리 어렵고 딱딱한 주제나 내용이더라도 비유의 표현으로 서문을 시작하면 독자가 쉽게 접근할 수 있다. 참신한 비유는 독자의 흥미를 끌 수 있지만, 생략되고 전무한 비유는 오히려 흥미를 떨어뜨릴 뿐만 아니라 글에 대한 신뢰성을 감소시킨다.</p> <p>자동차가 고장이 나면 자동차 병원이라고 할 수 있는 카센터에 가서 고치면 된다. 카센터에 가면 구멍 난 타이어를 바꿀 수도 있고, 벗겨진 표면을 도색할 수도 있고, 케칭 헤드라이트를 바꿀 수도 있다. 좀더 좋은 카센터에 가면 렌턴도 바꿀 수 있을 것이다. 그런 면에서 보면 자동차는 정말 많은 주치의들을 쓴 셈이다.</p>	42	0	정방향, 정답	1.59 초	문서가 힘
 <p>간혹 어떤 컴퓨터는 특수 프로그램에 감염되거나 해킹을 당해 내용을 이해하지 못하고 발표할 수 없는 수도 있다.</p> <p>2) 백신의 컴퓨터를 사용하지 않을 경우, 미리 발표장의 컴퓨터와 같은 소프트웨어의 백신을 확인하거나 시연할 한다. 발표용 소프트웨어는 그 버전이나 사용 컴퓨터에 정황을 맞출 것을 확인하거나 확인한다. 최근 버전의 소프트웨어는 그 이전 버전의 소프트웨어와는 다른 기능을 탑재한 과정을 제대로 인식하지만, 그 반대의 경우는 그렇지 못한 것이 일반적이다. 즉, 최신 발표장의 소프트웨어가 최신 버전이 아니고, 발표자가 연구실이나 집에 컴퓨터를 설치한 과정을 제대로 인식하지 못한다. 즉, 발표장에서 발표 자료에 대한 발표 자료 준비 시 사용한 버전이 최신 버전이었다면, 작성 발표장에서 발표 자료가 제대로 인식되지 못하는 상황을 당하는 수가 있다. 이러한 상황을 미연에 방지하기 위해서는 발표용 소프트웨어의 종류는 물론 버전도 확인할 필요가 있다.</p> <p>발표용 소프트웨어는 또한 컴퓨터에도 민감하다. 즉 발표자의 컴퓨터에서와 발표장에서 컴퓨터가 다른 경우 의외의 차이가 생길 수 있다. 대부분 이러한 경우는 화면상의 위치나 형태의 미세한 변화 등 치명적이지는 않은 문제가 대부분이나, 때로는 발표자를 당혹스럽게 하는 경우도 있다.</p> <p>이러한 모든 문제를 방지하는 데 가장 중요한 방법이 발표장에서 리허설을 해보는 것이다. 때문에 많은 발표장에는 시연용 방에 따로 발표장과 동일한 컴퓨터와 소프트웨어가 설치되어 있다.</p> <p>3) 발표장에서 음향 효과가 사용 가능함을 확인하고 사용방법을 구상한다. 특수효과 중의 하나는 음향효과이다. 주의할 점 중의 하나는 많은 발표장이 음향효과를 지원하지 않는다는 것이다. 따라서 발표장 설비가 음향효과를 지원하지 않는다는 확인이 없으면 음향효과에 의존하지 않는 것이 바람직하다.</p> <p>이와 같이 발표 도구에 따라 유의해야 할 사항들이 많으며 이를 고려해야 좋은 발표가 이루어질 수 있다. 일반적으로 발표를 하는 동안, 청중의 생각과 통하는 시각적 경로를 이용하는 것이 가장 설득력이 있다. 이를 잘 사용하면 큰 도움이 될 것이다. 하지만, 나쁜 시각 정보나 선택적 사용될 시각 정보 또는 너무 많은 시각 정보들은 청중에게 어떠한 발표가 시뮬레이션할 수 없기 때문에 자신의 이미지에 심각한 손상을 줄 수도 있다.</p>	41	1	정방향, 정답	2.11 초	문서가 기울어짐
 <p>the Hunkpapa Sioux in 1867. In the following years, he also attracted a large following among the Cheyenne and Arapaho. He is best known today for his role in defeating General Custer at the Battle of Bighorn in 1876. Although eventually forced to live on the Standing Rock Reservation in South Dakota, he never surrendered to his dignity or his devotion to the Sioux way of life.</p> <p><b>한 문장 내에서 동동태와 수동태를 혼용하는 것을 피한다.</b> 국문을 영문으로 직역할 때 초보자들이 흔히 범할 수 있는 실수이며 번역 후 세심한 교정을 통하여 바로잡도록 한다.</p> <p>The number of passengers boarding the ship went over the allowable limit and the air in the passenger's cabin is polluted. → The number of passengers boarding the ship went over the allowable limit and the air in the passenger's cabin becomes impure.</p> <p><b>일관성을 유지한다.</b> 한 문장 내 혹은 여러 문장들 사이에서 대/소문자, 문장의 형태 등을 일관적으로 하는 것을 말하며 영어로는 parallelism이라고 불리운다.</p> <p>A session can be initiated in two ways: A network manager sends a command to the control software. Issuing of a command by a third unit. → A session can be initiated in two ways: By a network manager sending a command to the control software. By a third unit issuing a command.</p> <p>Mail needs to be picked up, call Tom, science project, and don't forget to ask John for money. → Pick up mail, call Tom, work on science project, and don't forget to ask John for money.</p>	22	3	정방향, 정답	1.43 초	한글, 영문 동시 포함

입력 이미지	정방향 인식 음절 수	역방향 인식 음절 수	최종 판단	수행 시간	이미지 특징
<p>▶ 그림 2-17 (2) <math>\text{span}(a_1, a_2, a_3)</math>가 직선, (3) <math>\text{span}(a_1, a_2, a_3)</math>가 평면</p>  <p>이 기호를 사용하면 <math>\text{Im } f = \text{span}\{a_1, \dots, a_n\}</math>이므로 <math>\text{span}\{a_1, \dots, a_n\}</math>의 차원이므로 <math>\text{rank } f</math>입니다. 2, 3, 4월 '남적하계를 식으로 나타내다'의 논의를 떠올려보면 <math>a_1, \dots, a_n</math>이 선형독립이면 <math>\text{rank } f = \text{span}\{a_1, \dots, a_n\}</math>의 차원은 <math>n</math>, 선형종속이면 <math>\text{rank } f</math>의 차원은 <math>&lt; n</math>이었습니다. 좀 더 자세히 말하면 선형종속의 경우에도 <math>\text{rank } f</math>의 차원을 알려면 무엇을 세면 좋을까? 그것이 이 절의 주제입니다.</p> <p>지금 만약 <math>n</math>개의 벡터 <math>a_1, \dots, a_n</math>이 더 작은 <math>r(&lt; n)</math>개의 벡터 <math>b_1, \dots, b_r</math>을 '재료'로 하여</p> $a_1 = c_{11}b_1 + \dots + c_{1r}b_r$ $\vdots$ $a_n = c_{n1}b_1 + \dots + c_{nr}b_r \quad (2.27)$ <p>(<math>c_{ij}</math>는 수)의 같이 쓰였다고 합시다. 즉, <math>a_1, \dots, a_n</math>이 모두 <math>b_1, \dots, b_r</math>의 선형결합으로 표현되는 상황입니다. 그러면 <math>\text{span}\{a_1, \dots, a_n\}</math> 내의 벡터 <math>y</math>는 모든 <math>b_1, \dots, b_r</math>으로 쓰여집니다. 실제 <math>y</math>는 식 (2.26)과 같이 쓰여지므로</p> $y = x_1 a_1 + \dots + x_n a_n$ $= x_1(c_{11}b_1 + \dots + c_{1r}b_r) + \dots + x_n(c_{n1}b_1 + \dots + c_{nr}b_r)$ $= (c_{11}x_1 + \dots + c_{n1}x_n)b_1 + \dots + (c_{1r}x_1 + \dots + c_{nr}x_n)b_r$ $= (\sum x_i)c_{i1}b_1 + \dots + (\sum x_i)c_{ir}b_r$ <p>의 형태로 변형할 수 있습니다. 그러면 <math>n</math>개의 실정값 <math>x_1, \dots, x_n</math>을 아무리 움직인다 한들 <math>y</math>는 <math>\text{span}\{b_1, \dots, b_r\}</math>에서 나오지 않습니다. 사모강의 있는 영역 <math>\text{span}\{b_1, \dots, b_r\}</math>은 겨우 <math>r</math>차원이므로 <math>\dim \text{span}\{a_1, \dots, a_n\} \leq r</math>이 됩니다. <math>a_1, \dots, a_n</math>이란 훌륭한 <math>n</math>개의 조합처럼 꾸미고 있지만 본색을 드러냈다는 느낌이네요. <sup>53</sup></p> <p><sup>53</sup> 앞의 바치는 "..."는 사원이라고 말하려고 싶어하는데, <math>b_1, \dots, b_r</math> 자체에도 아직 군더더기가 포함되어 있지 않습니까? 그러므로 "..."는 사원이라고 말할 수밖에 없습니다. 또한, '가무'라는 단어의 의미는 무복의 각수 값을 참고하세요.</p> <p><sup>54</sup> 이 이미지는 2.3.5절의 보색에 관한 문제를 다른 표현으로 못하고 있는 것 같습니다.</p> <p>182</p>	26	10	정방향, 정답	2.31 초	수식, 그림 포함
<p><b>GPU 프로그램 최적화</b> 성능 이끌어내기</p> <p>GPU에는 많은 연산기가 있으며, 장치 메모리에도 높은 대역폭의 GDDR5 메모리 등을 사용하므로 CPU와 비교하면 높은 연산 성능과 높은 메모리 대역폭을 가집니다. 그러나 이를 효율적으로 사용하는 프로그램이 아니라면 이들 자원이 놀게 되므로 성능을 끌어내지 못합니다. 이 절에서는 GPU에서 높은 성능을 이끌어내기 위한 주의 사항이나 기법을 살펴보겠습니다.</p> <p><b>NVIDIA GPU의 그리드 실행</b></p> <p>NVIDIA의 GPU는 그림 4.8에 나타낸 것처럼 스트레드 블록(스트레드 묶음)을 묶어 그리드라는 단위로 실행을 수행합니다.</p> <p>▶ 그림 4.8 스트레드 블록을 SM에 할당</p> 	17	2	정방향, 정답	2.17 초	그림 포함
<p>⑥ 혁신적인 환경에너지 시스템 실용화 ⑦ 차세대 산업용 떠받치는 새로운 기능 장치/고성능 재료 만들기 ⑧ 근미래형 제조업용 선도할 혁신적인 설계/제조 공정 개발 ⑨ 우주의 기본 법칙과 진화의 해명</p> <p>예를 들어 ①은 생체 조직 등을 분자, 원자 수준으로 모델화하여 약제와의 상호작용이 어떤 결과를 나타내는지 시뮬레이션하여 구한 다음, 부작용이 적고 효과가 좋은 약을 만들기 위한 기반을 개발하는 과제입니다. 원자 수준에서 움직이는 힘을 계산하는 것이므로, 원자의 수가 많아 엄청난 계산이 필요합니다.</p> <p>그림 3.19는 물속에 DNA가 있는 상황을 시뮬레이션한 결과의 한 화면입니다. 여기서 배경에 흩어져 있는 바늘처럼 보이는 것이 물 분자입니다. 기대 분자인 DNA와 대량의 물 분자가 주위에 존재하는 상황의 해석은 종래는 불가능했습니다. 물질/재료 연구기구나 런던대학(University College London) 등의 팀이 종래의 100배 이상의 원자를 포함하는 대규모 제1원리(제1원리 계산이란 원자 사이에 움직이는 힘을 물리 원리에 기반을 두고 계산하는 방법) 시뮬레이션 방법 개발에 성공하여 이러한 종류의 시뮬레이션이 가능해졌습니다.</p> <p>▶ 그림 3.19 물 용매 안의 DNA 시뮬레이션*</p>  <p>* 이미지 제공: 국립연구개발인문융합연구소 및 이화여자대학교</p> <p>121</p>	27	1	정방향, 정답	1.38 초	그림 포함



입력 이미지	정방향 인식 음절 수	역방향 인식 음절 수	최종 판단	수행 시간	이미지 특징
<p>OpenACC에 대해 설명합니다. OpenACC는 컴파일러가 컴파일 시점에 OpenACC 지시자를 사용하여 병렬 코드를 생성하고, 실행 시에 하드웨어 가속을 제공하는 프로그래밍 언어입니다. OpenACC는 OpenMP와 유사하지만, OpenACC는 컴파일 시점에서 병렬화를 시키고, OpenMP는 런타임에 병렬화를 시킵니다. OpenACC는 OpenMP와 달리, 컴파일 시점에서 병렬화를 시키고, OpenMP는 런타임에 병렬화를 시킵니다. OpenACC는 OpenMP와 달리, 컴파일 시점에서 병렬화를 시키고, OpenMP는 런타임에 병렬화를 시킵니다.</p> <p>OpenACC는 OpenMP와 달리, 컴파일 시점에서 병렬화를 시키고, OpenMP는 런타임에 병렬화를 시킵니다. OpenACC는 OpenMP와 달리, 컴파일 시점에서 병렬화를 시키고, OpenMP는 런타임에 병렬화를 시킵니다.</p>	5	15	역방향, 정답	2.55초	한글, 영문 동시 포함
<p>Kernel의 크기에 따른 성능 차이를 설명합니다. Kernel 크기가 작을수록 데이터 전송 비용이 낮아 성능이 향상됩니다. Kernel 크기가 클수록 데이터 전송 비용이 높아 성능이 저하됩니다. Kernel 크기에 따른 성능 차이를 설명합니다. Kernel 크기가 작을수록 데이터 전송 비용이 낮아 성능이 향상됩니다. Kernel 크기가 클수록 데이터 전송 비용이 높아 성능이 저하됩니다.</p> 	13	10	정방향, 오답	1.28초	그림 포함
<p>OpenACC의 성능을 높이기 위한 방법을 설명합니다. OpenACC의 성능을 높이기 위해서는 컴파일러 옵션 설정, 하드웨어 가속기 사용, 그리고 코드의 최적화가 중요합니다. OpenACC의 성능을 높이기 위한 방법을 설명합니다. OpenACC의 성능을 높이기 위해서는 컴파일러 옵션 설정, 하드웨어 가속기 사용, 그리고 코드의 최적화가 중요합니다.</p> 	28	24	정방향, 오답	2.58초	그림 포함

### 3. 한계점

정방향 문서의 경우 문서 내에 그림이 나 수식, 영문이 어느 정도 포함되어 있는 문서라도 다수의 한글 음절이 추출 가능하므로 상하 반전 여부 판단을 정확하게 수행할 수 있음을 확인하였다.

하지만 역방향 문서에서는 하나의 음절에 대한 상하 반전 여부를 잘못 판단한 경우가 정방향 문서에 비해서 평균적으로 3회 많았으며 역방향 문서에 그림, 수식, 영문이 포함되어 있는 경우 문서 자체의 상하 반전 여부를 잘 못 판단한 경우도 발생하였다.

또한 스캔 후 압축을 진행하였거나 스캔 품질 자체가 떨어지는 저해상도의 스캔본에 대해서는 음절의 구분이 쉽지 않으며 음절내의 자음, 모음을 구분할 수 있을 정도의 해상도를 스캔본이 보유하고 있지 않기 때문에 본 연구에서 제시한 알고리즘 으로는 문서의 상하 반전 여부 판단을 진행할 수 없었다.

한 장의 문서에 대한 처리 소요 시간은 평균적으로 1.8 초로 대량의 문서 스캔 과정에 있어서 실시간성을 보장하기는 어렵지만, 본 연구에서는 CPU만을 사용한 점, 알고리즘이 단순하여 최적화가 가능한 점, GPU 사용이 가능한 점 등으로 처리소요시간을 단축할 수 있을 것 이라고 예상된다.

## IV. 결 론

본 연구에선 스캔본의 상하 반전 여부를 알고리즘을 통해 탐지하고 자동화하는 것을 목표로 하였다. 한글의 언어학적 특징인 자음, 모음의 위치 상관관계를 사용하여 문서가 기울어져 있거나 왜곡되어 스캔되어도 이에 강인하게 대처할 수 있는 알고리즘을 제안하였다.

이와 같은 방법은 한 음절 내에 자음, 모음이 포함되는 한글이나 중국어와 같은 문자에 동일하게 적용할 수 있다.

하지만 본 연구에서 사용된 알고리즘은 스캔본의 품질이 떨어져 자음, 모음을 구분할 수 없게되면 적용할 수 없다는 단점과 영어와 같이 한 음절은 하나의 모음이나 하나의 자음으로만 이루어 지는 언어에는 적용할 수 없다는 한계점을 가지고 있다.

본 연구에서 제한한 방법을 통해 인쇄물 상하 반전 여부를 사람이 아닌 알고리즘을 통해 판단하므로 대량의 인쇄물을 스캔하여 디지털화 하는 관공서, 병원, 학교 등에서 추가적인 인력 낭비를 줄일 수 있을 것이다.

## 참 고 문 헌

- [1] M. Kim, S. Yim, Y. Lee, M. Kim and J. Jung, "A Study on Automated Checking for Upside Down Printed Materials based on Optical Character Recognition," 2018 International Conference on Fuzzy Theory and Its Applications (iFUZZY), Daegu, Korea (South), 2018, pp. 115–118. doi: 10.1109/iFUZZY.2018.8751690
- [2] 딥러닝과 OpenCV를 활용해 사진 속 글자 검출하기,  
<https://d2.naver.com/helloworld/8344782>, (검색일: 2019.11.20.)
- [3] OpenCV 4.1.1 document,  
<https://docs.opencv.org/4.1.1>, (검색일: 2019.12.16.)

## 부 록

### 1. 소스코드

```
import cv2
import numpy as np
import time
import os

path_dir = './img/img2'
file_list = os.listdir(path_dir)

for file in file_list:

    start = time.time()

    img = cv2.imread(path_dir+'/'+file)
    _img = img.copy()

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    _, thresh = cv2.threshold(gray, 170, 255, cv2.THRESH_BINARY_INV)

    kernel = np.ones((25, 25), np.uint8)
    result = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)

    contours, hierarchy = cv2.findContours(result, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
```

```

for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.rectangle(_img, (x, y), (x + w, y + h), (0, 0, 255), 2)

direction_good_count = 0
direction_bad_count = 0
character_count = 0

for cnt in contours:    # for each string

    x, y, w, h = cv2.boundingRect(cnt)

    crop = thresh[y:y+h, x:x+w]

    pass_flag = False

    crop_points = [0]

    _crop = cv2.cvtColor(crop, cv2.COLOR_GRAY2RGB)

    # slicing string by one by one character
    for col in range(w):
        for row in range(h):

            if crop[row][col] > 0:
                pass_flag = False
                break

            if row == h - 1 and pass_flag == False:
                cv2.line(_crop, (col, 0), (col, row), (0, 0, 255), 2)
                crop_points.append(col)
                pass_flag = True

    crop_points.append(w - 1)

    one_character_width = 0
    one_character_height = h

    for i in range(len(crop_points) - 1):    # for each one character

```

```

consonant_list = []# 자음 리스트
vowel_list = [] # 모음 리스트

one_character_width = crop_points[i+1] - crop_points[i]

if crop_points[i+1] - crop_points[i] > 0:

    crop_one_character = crop[:,crop_points[i]:crop_points[i+1]]

    backtorgb = cv2.cvtColor(crop_one_character,cv2.COLOR_GRAY2RGB)
    _contours, _hierarchy = cv2.findContours(crop_one_character,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # 한 글자 안에 너무 많은 자음/모음이 검출되거나 아예 검출되지 않
은 경우 pass
    if len(_contours) >= 4 or len (_contours) == 0:
        continue

    character_count += 1

    for cnt in _contours:
        x, y, w, h = cv2.boundingRect(cnt)

        if h / w >= 2.5 or h / w <= 0.4: # 모음 후보
            cv2.rectangle(backtorgb, (x, y), (x + w, y + h), (0, 0,
255), 2)
            vowel_list.append([x, y, w, h])

        else: # 자음 후보
            cv2.rectangle(backtorgb, (x, y), (x + w, y + h), (255, 0,
0), 2)
            consonant_list.append([x, y, w, h])

    # 자음이 여러개, 모음이 하나만 검출된 경우 (best case)
    if len(consonant_list) == 1 and len(vowel_list) == 1:

        consonant_w, consonant_h = consonant_list[0][2:]
        vowel_w, vowel_h = vowel_list[0][2:]

```

```

# 모음이 너무 작은경우, 직선 등의 노이즈가 모음으로 잘못 검출된 경
우
    if vowel_w / one_character_width < 0.5 and vowel_h /
one_character_height < 0.5:
        continue

    # 자음이 너무 작은경우
    if consonant_w / one_character_width < 0.25 and consonant_h /
one_character_height < 0.25:
        continue

    if consonant_list[0][0] < vowel_list[0][0]:    # 모음이 자음보다
오른쪽에 위치, ex) 가
        direction_good_count += 1

    elif consonant_list[0][1] < vowel_list[0][1]:    # 모음이 자음보다
아래에 위치, ex) 그
        direction_good_count += 1

    else:
        direction_bad_count += 1

print()
print('--- RESULT: ',path_dir+'/' + file, '---')
print('good: ', direction_good_count)
print('bad: ', direction_bad_count)
print('total character: ', character_count)
print('direction: ', direction_good_count - direction_bad_count)
print('time: ', time.time() - start)

cv2.destroyAllWindows()

```