http://dx.doi.org/10.5392/JKCA.2014.14.12.545

깊이 영상 기반 적응적 체인 코드를 이용한 한자 학습 시스템

Depth Image based Chinese Learning Machine System Using Adjusted Chain Code

김기상, 최형일
숭실대학교 미디어학과

Kisang Kim(illusion1004@gmail.com), Hyung-Il Choi(hic@ssu.ac.kr)

요약

본 논문에서는 깊이 카메라를 이용한 실시간 사용자 한자 학습 시스템을 제안한다. 사용자 학습 방법으로는 사용자가 화면에서 손을 움직여 한자를 입력하고, 입력 제스처와 미리 저장된 템플릿을 매칭하여 사용자가 한자를 올바르게 썼는지 판단한다. 이를 위해 본 논문에서는 손가락 검출 및 검증을 통한 손 영역 검출 및 추적 방법과 스트로크의 연속성을 분석하기 위해 적응적 체인 코드를 제안한다. 손가락 검출로는 깊이 값을 이용하여 손 영역을 검출 후, 손가락의 축을 생성, 손가락의 두께를 이용하여 검증한다. 손 영역 추적으로 생성된 스트로크는 추적된 점들과 순서 그리고 길이 정보가 포함되어 있다. 이들을 이용하여 사용자가 올바른 입력을 했는지 확인하기 위해 적응적 체인 코드 방법을 제안한다. 이 방법은 매칭 속도와 스트로크 안에서 잘못 입력된 부분을 찾는데 매우 효율적이다. 실험 결과에서는 본 논문에서 제안한 시스템이 실시간으로 동작하며 학습 과정과 오류 검출에 매우 효과적임을 보여준다.

■ 중심어 : | 제스처 인식 | 체인 코드 | 학습 시스템 |

Abstract

In this paper, we propose online Chinese character learning machine with a depth camera, where a system presents a Chinese character on a screen and a user is supposed to draw the presented Chinese character by his or her hand gesture. We develop the hand tracking method and suggest the adjusted chain code to represent constituent strokes of a Chinese character. For hand tracking, a fingertip is detected and verified. The adjusted chain code is designed to contain the information on order and relative length of each constituent stroke as well as the information on the directional variation of sample points. Such information is very efficient for a real-time match process and checking incorrectly drawn parts of a stroke.

■ keyword : | Gesture Recognition | Chain Code | Learning Machine System |

I. 서 론

과학 기술이 발달과 역사가 진행되는 등, 우리 주변에는 항상 새로 배워야 할 것들이 많이 생긴다. 이러한 것들을 배우는데 있어 가장 많이 사용되는 방법은 반복 학습이다. 하지만, 이러한 방법은 지루함을 제공하고, 학습의 효율을 떨어뜨리게 된다. 최근에는 이런 지루함을 줄어주며, 효율적인 학습을 하기 위해 많은 연구들

* 본 연구는 서울시 산학연 협력사업(SS110013)과 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2013R1A1A2012012)
접수일자 : 2014년 09월 26일
수정일자 : 2014년 11월 25일
심사완료일 : 2014년 12월 08일
교신저자 : 최형일, e-mail : hic@ssu.ac.kr

이 진행되고 있다. 그 중 가장 효율적이라고 알려져 있는 방법은 게임을 통한 학습 시스템이다. 일반적으로 사람들은 게임을 함으로서 흥미와 즐거움을 동시에 느낀다. 또한, 게임은 사람의 삶에 매우 밀접하게 연결되어 있다. 이러한 장점을 이용하기 위해 예로부터 이러한 연구가 많이 진행되고 있다. 가장 단순하게는 퍼즐이나 퀴즈게임이 일반적으로 널리 알려진 게임을 통한 학습 방법이다. 본 논문에서는 한자 게임 시스템을 위한 적응적 체인 코드를 이용한 사용자의 손동작 인식을 통한 한문 학습 시스템을 제안한다.

한문 학습 시스템을 위해서는 가장 중요한 것은 실시간 손동작 인식이다. 최근에는 손동작 인식 연구에 대해 많은 보고가 올라오고 있다. 예전부터 가장 널리 사용되는 방법으로는 은닉 마르코브 모델(Hidden Markov Model)이 있다[1]. 이 방법은 시공간적인 변이를 가진 패턴을 표현하고 인식할 수 있는 확률 모델이라는 장점이 있다. 반면에, 스트로크를 구성하는데 있어 모든 제스처가 스트로크가 될 수는 없다. 이러한 제스처가 아닌 부분을 제거하는 것에 대한 문제가 존재한다. 다른 방법으로는 DTW(Dynamic Time Warping) 방법이 널리 사용된다[2]. 이 방법의 장점은 입력 데이터가 주어지면 다시 샘플링을 할 필요가 없다는 것이다. 또한, 샘플링을 할 필요가 없기 때문에, 입력간의 거리(속도)가 인식하는데 있어 하나의 구성요소가 될 수 있다. 하지만, 이러한 장점이 있음에도 불구하고, 인식 시간에 있어 입력과 템플릿들 간의 매칭 모델을 매번 만들기 때문에 시간 복잡도가 증가하는 문제가 있다. 최근 연구로 나온 방법은 $1 인식기 방법이 있다[6]. 본 인식기 방법은 미국 워싱턴 대학에서 많은 연구가 진행되었다. 워싱턴 대학에서는 이러한 방법을 확장하여 $N 인식기, $P 인식기 방법에 대해 제안하였다[7][8]. 이들 방법은 DTW보다 빠르고 좋은 결과를 보이는 강점이 있지만, 인식 결과에 있어 아직 정확도가 미흡하며, 인식 속도 또한 여전히 저조함을 보인다. 이러한 문제들을 해결하기 위해 적응적 체인 코드 방법을 본 논문에서 제안한다. 이 방법은 계산이 간단하여 다른 방법들에 비해 빠른 처리속도를 보이며, 잘못된 부분을 세세하게 보여주는 장점이 있다.

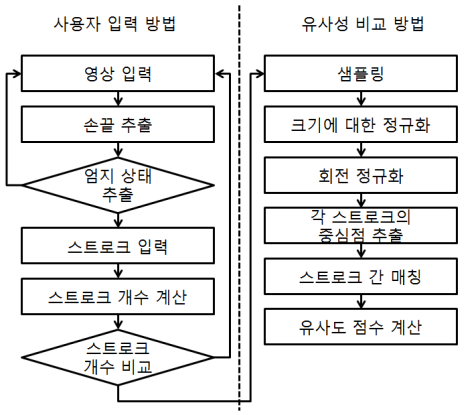

그림 1. 시스템 전체 흐름도

[그림 1]에서는 논문에서 제안하는 시스템에 대한 전체 흐름도를 보여준다. 본론의 1장에서는 스트로크 생성에 대해 설명한다. 학습기와 사용자간의 상호작용을 위한 인터페이스로는 손동작을 사용한다. 사용자가 손동작을 통해 한자를 입력하면, 이를 스트로크들로 표현하기 때문이다. 본론의 2장에서는 적응적 체인 코드와 제스처와 템플릿간의 매칭 방법을 설명한다. 결론에서는 실험결과를 기술하고, 기존의 방법에 비해 제안한 방법에 대한 효율성을 보여준다.

II. 본 론

1. 스트로크 생성

본 논문에서는 사용자가 한자를 입력할 때 검지만 편 상태에서 움직인다는 가정 하에 설계되었다. 이 장에서는 손동작을 이용한 스트로크의 생성 방법에 대해 설명한다.

1.1 검지 검출

검지 검출에 앞서 우선 손 영역 검출을 시도한다. 손 영역 검출하는데 있어 사용자의 손은 항상 몸보다 앞에 있게 된다. 이러한 특징을 이용, 사용자의 손은 깊이 카메라로부터 가장 가까운 거리에 놓여있다고 가정한다. 이 가정을 적용하면, 손 영역 검출은 단순히 임계값을 이용하여 검출할 수 있다. 임계값을 검출하기 위해 우

선 누적 영상(Integral Image, G)을 생성한다. 누적 영상을 통해 일정 영역 크기(ω)에서 평균 깊이 값이 최소가 되는 점을 임계값(T)로 설정한다. 누적 영상을 사용하는 이유는 영역의 평균값을 구하기 위해 빠른 계산을 위해 사용되었으며, 평균값을 사용 시 영상의 잡음으로 인해 나타날 수 있는 오류를 줄여주기 때문이다. 수식 (1)은 ω크기의 마스크에 해당하는 깊이 값의 평균(S)를 구하는 수식이다. ω값은 영상의 깊이 값에 따라 사이즈가 바뀌지만, 최대 크기는 30으로 사용하였다. 그 이상으로 커질 경우, 손 이외의 영역이 많이 포함되어 제대로 된 검출 결과를 얻기 어렵기 때문이다.

$$S(x,y)=\frac{\begin{matrix}G(x+w,y+w)\\-G(x+w,y-w)\\-G(x-w,y+w)\\+G(x-w,y+w)\end{matrix}}{(2w+1)^2} \quad (1)$$

수식 (1)에서 S값들 중 최소가 되는 값에 α를 더한 값을 최종 임계값(T)으로 설정한다. S값을 그대로 사용할 경우 손 영역의 일부만 검출 될 수 있기 때문이다. 본 시스템에서 α는 50으로 사용되었다.

손 영역 검출 후, 손 끝 검출 및 검증을 진행한다. 손 끝 검출하기 위해 본 시스템에서는 사용자가 제스처를 입력할 때 검지의 방향은 항상 위로 향한다는 가정을 이용, 손 영역에서 가장 위에 있는 픽셀 좌표를 손 끝 후보로 정한다. 후보로 정하는 이유는 모든 상황에서 가장 위에 있는 픽셀이 손 끝 이라고 볼 수 없기 때문이다. [그림 2]에서 (a)는 손끝이 맞을 경우의 예제이며, (b)는 손끝이 아닌 경우에 예제이다. 이러한 문제점을 해결하기 위해 손 끝 검증 방법을 제안한다.

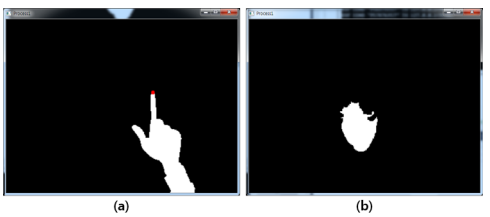

그림 2. 검지 검증이 필요한 사례. (a) 손 영역에서 검지가 있는 경우 (b) 손 영역이 아닌 영상

손 끝 검증으로는 검지의 두께와 깊이 값을 이용한다. 일반적으로 카메라와 손의 거리가 가까울수록 손의 크기는 커지고, 반대의 경우는 작아지는 특성과 깊이 값은 거리에 비례하는 점을 이용한다. [그림 3]은 손 끝 검증 예제이다. [그림 3]에서 검지에 두 개의 선이 있는데, 위에 있는 선을 먼저 계산한다.

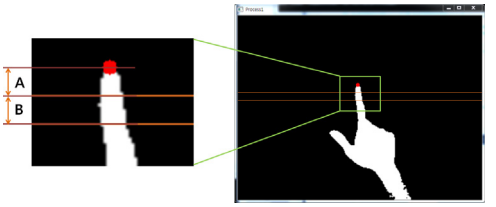

그림 3. 검지 검증 방법

우선 위에 있는 선은 손 끝 후보 점으로부터 y축으로 15픽셀만큼 밑에 있는 선이다. 이 선에서 손 영역에 해당하는 만큼 길이(A)를 측정한다. 밑에 있는 선은 윗선으로부터 y축으로 A픽셀만큼 밑에 있는 선이다. 밑 선에서 손 영역에 해당하는 만큼 길이(B)를 측정한다. 두 선의 길이의 두께가 측정되면, 두 두께의 관계를 비교한다. 영상 안에 검출된 검지의 두께는 위쪽 방향으로 향할 경우, 손바닥의 두께에 비해 월등히 얇게 된다. 이러한 전제 조건을 이용하여 수식(2~5)과 같은 조건을 만족할 경우 검지가 검출되었다고 판단한다.

$$C_1=\begin{cases}1 \text{ if}, A<1.5\times B\\0\ otherwise\end{cases} \quad (2)$$

$$C_2=\begin{cases}1 \text{ if}, B<1.5\times A\\0\ otherwise\end{cases} \quad (3)$$

$$C_3=\begin{cases}1 \text{ if}, A<Th \text{ and } B<Th\\0\ otherwise\end{cases} \quad (4)$$

$$C=\begin{cases}1 \text{ if}, C_1, C_2 \text{and } C_3 are\ true\\0\ otherwise\end{cases} \quad (5)$$

수식 (4)에서 Th는 검지 두께의 최대값이다. 이러한 이유는 이 이상 두께가 커질 경우, 이미 손 영역이 영상에서 너무 많은 부분을 차지하여 제대로 된 입력을 할 수 없기 때문이다. 수식 (5)에서 C값이 1일 경우 손 끝 검증은 참이며, 0일 경우 손끝이 아니라고 판단한다.

[그림 4]는 검지 검증 적용 예제이다.

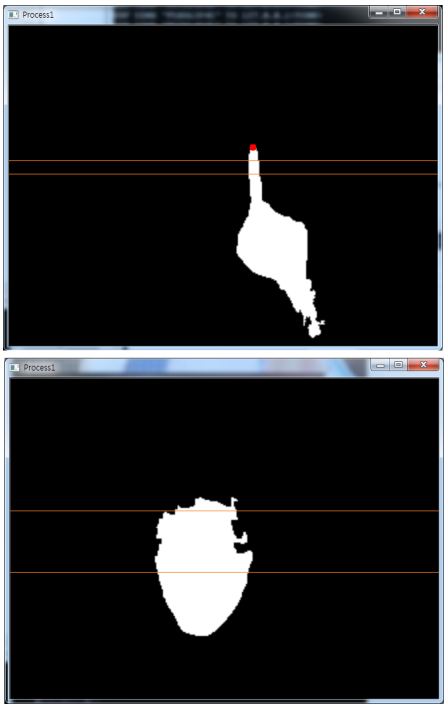

그림 4. 검지 검증을 통한 검지 검출 예제

1.2 엄지 상태 검출

스트로크를 구분하기 위해서는 검지의 움직임도 필요하지만, 시작과 끝을 알 수 있는 제스처도 필요하다. 이러한 제스처로 본 논문에서는 엄지의 상태를 보고 판단하는 방법을 소개한다. 손 모양을 변형시키는데 있어 엄지만큼 움직이기 수월한 손가락이 없기 때문에 엄지를 통해 상태를 인식한다. 영상에서 엄지가 없는 상태의 경우 입력을 종료한다. [그림 5]의 (a)영상에서 엄지가 존재하면 스트로크 입력을 시작한다. [그림 5]의 (b)는 엄지가 존재하지 않는 상태를 나타내며, 엄지의 존재를 검사하기 위해 본 논문에서는 검지와 엄지 사이의 골짜기 유무를 보고 판단한다.

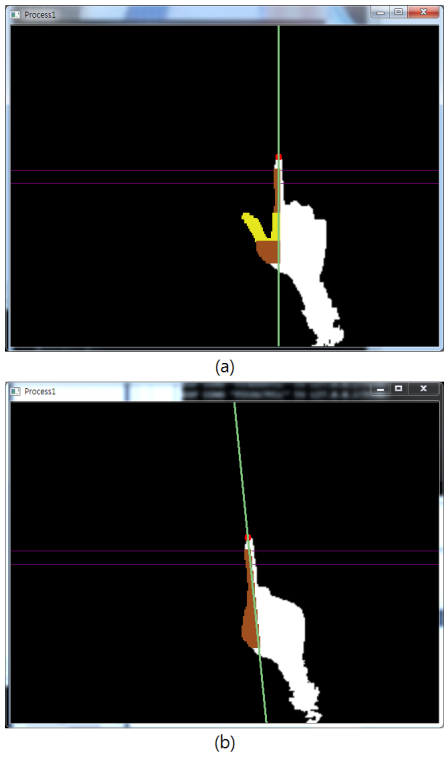

그림 5. 엄지 상태 검출 예제. (a) 엄지가 존재하는 상태 (b) 엄지가 존재하지 않는 상태

골짜기를 검출하기 위해 검지의 중심축을 계산한다. 축을 구하는 방법은 앞에서 구한 두께 (A)에서 중심 위치와 두께 (B)에서의 중심 위치를 연결하는 직선을 검지의 축으로 결정한다. 또한 골짜기를 찾기 위해 손 영역에 대한 에지영상을 추출한다. 축과 에지영상을 이용, 손끝을 시작점으로 수식 (6)에서의 E값의 길이만큼 축과 직교하는 방향으로 왼쪽에 대해서만 테두리 교차점의 개수를 구한다. 수식 (6)은 DepthSense 카메라 기반에서 깊이 값에 대해 필요한 영역을 계산하는데 있어 경험적으로 구해진 수식이다.

$$E = 1.8^{(600-depth)/100} \tag{6}$$

수식 (6)에서 E는 손끝의 깊이 값(depth)을 기준으로 손의 크기를 결정하는 함수이다. 왼쪽에 대해서만 계산하는 이유는 사용자는 항상 오른손을 사용한다는 전제

하에 계산하기 때문이다. [그림 5]에서 보여 지듯이 오른손의 엄지는 항상 축으로부터 왼쪽에 존재한다. 검증된 모든 직교한 선들 중 교차점 개수가 2개 이상인 선들의 비율이 15%이상이라면 엄지가 화면상에 존재한다고 판단한다. 반대로 그 이하라면 엄지가 없다고 판단한다. 15%라는 수치는 경험적 수치이며 일반적으로 해당 영역이 최소한 그 이상만큼 존재하기 때문이다.

2. 적응적 체인 코드를 이용한 매칭

2.1 적응적 체인 코드

사용자가 입력한 제스처와 화면에 보여 지고 있는 저장된 템플릿간의 매칭을 위해 적응적 체인 코드를 제안한다. 이 체인 코드는 한자의 특징을 표현하고 매칭하는데 있어 적합하다. 한자는 일반적으로 여러 개의 스트로크들로 구성되어 있다. 각각의 스트로크는 순서가 있으며, 길이, 입력 방향 등의 형태가 정해져있다. 이러한 것들을 표현할 수 있는 체인 코드 형태를 제안한다. [그림 6]는 적응적 체인 코드의 한 예제를 보여준다.

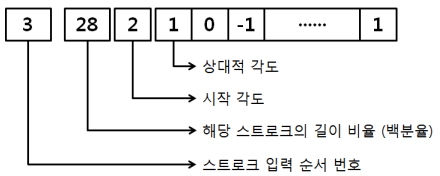

그림 6. 적응적 체인 코드 구성요소

체인 코드를 만들기 위해 각각의 입력된 스트로크를 12개의 동일간격으로 이루어진 점들로 다시 샘플링 한다. 12개로 정한 이유는 12개보다 많은 경우 鬱과 같이 복잡한 한문의 경우 많은 처리 시간을 요구하기 때문이다. 반대로 적은 경우에는 한자의 모든 특징을 표현하는데 있어 부적합한 문제가 있다.

적응적 체인 코드의 구성요소는 스트로크의 순서, 다른 스트로크와의 상대적 길이, 시작 벡터 그리고 차이 벡터들이다. 스트로크의 순서는 동일 순서의 스트로크간의 비교를 위해 필요하다. 또한, 순서가 잘못되었을 경우 이를 사용자에게 알려줄 수 있다. 상대적 길이는 모든 스트로크 길이의 총 합이 100%라고 했을 경우 각 스트로크의 길이 비율을 나타낸다. 각도는 스트로크간의 유사도를 측정하기 위한 핵심 방법이다.

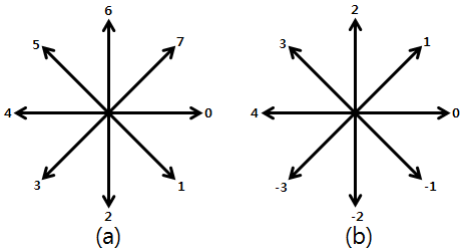

그림 7. 적응적 체인코드 각도 설정. (a) 시작 각도 설정 (b) 상대적 각도 설정

[그림 7]의 (a)에서 보듯 시작 각도는 8방향으로 정했을 경우 0부터 7 사이의 값을 갖게 된다. 이를 기준으로 상대적 각도를 계산하게 된다. 만일 이전 벡터와 현재 벡터가 차이가 없다면 이 값은 0으로 지정하게 되며, 다음 벡터를 비교할 시, 현재 벡터가 아닌 이전 벡터와 비교하게 된다. 하지만 일정 각도 이상 벌어지게 된다면, 다른 숫자가 입력되게 된다. 본 시스템에서는 [그림 7]의 (b)에서 나타내 듯, 8방향으로 정하여, -3부터 4사이의 값으로 설정하였다. [그림 8]은 적응적 체인 코드 생성 예제이다.

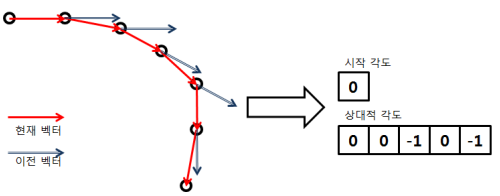

그림 8. 적응적 체인 코드 생성 예제

2.2 사용자 입력과 템플릿간의 매칭

스트로크들을 적응적 체인 코드로 표현하였다면, 다음 단계로는 사용자의 입력과 템플릿 간의 매칭을 통해 유사도를 측정한다. 유사도를 측정하기 위해 스트로크의 순서, 위치, 그리고 잘못된 부분들을 계산한다.

순서와 위치를 확인하기 위해, 각 스트로크마다 중심점$(ax,\ ay)$을 추출한다. 중심점을 추출하기 위해 우선 입력점들의 위치를 정규화 한다. 입력과 템플릿간의 비

교하는데 있어 폭과 높이가 동일해야 위치 유사도를 측정할 수 있기 때문이다. [그림 9]에서 사각형은 입력점들이 정규화 된 것을 나타낸다. 정규화가 되었다면 각 스트로크에서 입력점들의 평균을 계산해 중심점을 추출한다. [그림 9]에서 나타내는 점들이 각 스트로크의 중심점들을 말한다.

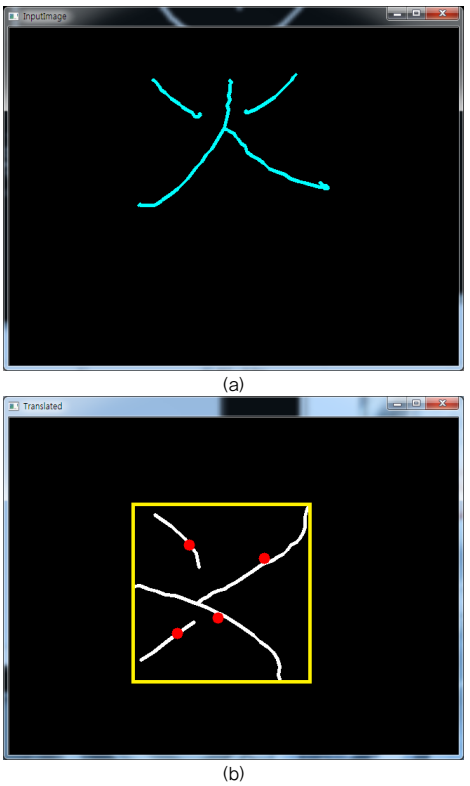

그림 9. (a) 입력 영상 (b) 입력 스트로크들에 대한 정규화, 중심점 그리고 최적의 회전 결과.

[그림 9]에서 90 정도로 회전된 이유는 매칭 테스트를 위해 템플릿을 인위적으로 90도를 회전시켜 저장해 둔 상태이기 때문이다. 회전을 시켜둔 이유는 회전된 상태에서도 제대로 매칭 될 수 있는지 검증하기 위해 회전된 템플릿을 저장하고 실험하였다. 위치와 순서의 유사도를 측정하기 위해 수식 (7)을 사용한다. (ax_{si}, ay_{si})는 사용자가 입력한 스트로크의 중심점들을 말하며, (ax_{pi}, ay_{pi})는 템플릿 스트로크의 중심점들을 말한다. N은 스트로크의 개수를 나타내며, (cx, cy)는 사각형의 중심점을 나타낸다. 비교하는데 앞서 사각형의 중심점을 기준으로 입력 스트로크들을 각도 θ만큼 회전하여 비교하며, 그중 유사도가 가장 좋은 최소가 되는 값 $S(x,y)$를 최종 유사도로 측정한다.

$$S(x,y)=\frac{1}{N}\sum_{i=1}^{N}\begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix}\cdot\begin{pmatrix}ax_{si}-cx\\ ay_{si}-cy\end{pmatrix}-\begin{pmatrix}ax_{pi}-cx\\ ay_{pi}-cy\end{pmatrix} \quad (7)$$

유사도 $S(x,y)$를 계산하였다면, 수식 (8)을 이용해 유사도에 대해 점수($Score$)를 측정한다. S_x와 S_y는 $S(x,y)$의 각 구성요소이며, n_x와 n_y는 정규화 된 사각형의 폭과 높이를 나타낸다.

$$Score=1-\frac{\sqrt{S_x^2+S_y^2}}{\sqrt{n_x^2+n_y^2}} \quad (8)$$

유사도 점수가 임계값(α, 0~1사이의 값)보다 높은 경우 입력 스트로크들의 위치와 저장된 템플릿의 스트로크들 위치가 유사하다고 판단한다. 본 시스템에서 임계값은 0.6 ~ 0.8 사이의 값으로 설정하였다. 위치가 유사하면 마지막으로 각각의 스트로크 간의 유사성을 판별하기 위해 적응적 체인 코드를 사용한다. [그림 10]은 적응적 체인 코드를 이용한 비교 예제이다.

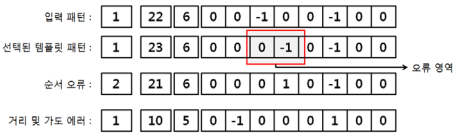

그림 10. 적응적 체인 코드를 사용한 패턴 비교 결과

체인 코드에서 우선 길이 비율을 비교한다. 대부분의 경우 입력 패턴이 템플릿 패턴을 보고 따라 그리기 때문에, 사용자가 제대로 입력하였다면, 두 비율은 매우 유사하다. 이러한 점을 감안하여 길이 비율의 오차는 최대 5%내외로 설정하였다. 마지막으로는 각도에 대해

비교하였다. 각도의 차이를 누적계수를 이용하였기 때문에, 누적 그래프를 그려 그 유사성을 판별하였다. 이 누적 그래프는 유사성을 측정할 수 있을 뿐만 아니라, 어느 부분이 잘못되어 있는지 알 수도 있다. [그림 11]은 비교 누적 그래프의 한 예제이다.

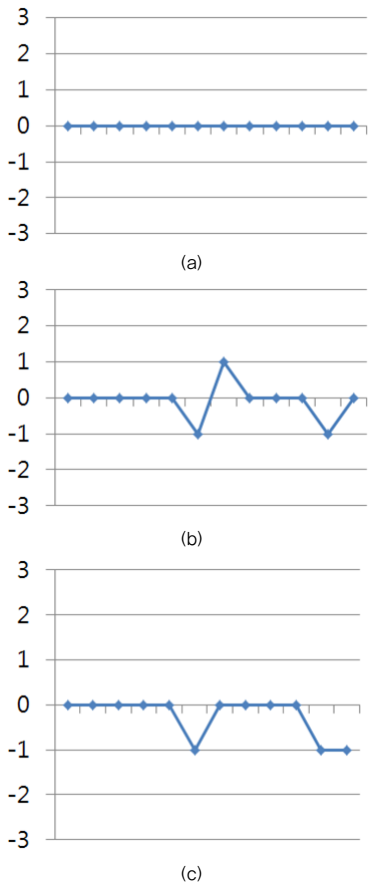

그림 11. 적응적 체인 코드 비교 그래프. (a) 저장된 템플릿의 적응적 체인 코드 (b) 입력 영상의 적응적 템플릿 코드 (c) (a)와 (b)의 비교 결과 및 오류 영역 검출 결과

III. 실험 결과 및 결론

1. 실험 결과

본 시스템을 개발하기 위한 환경으로 운영체제는 Windows 7 Enterprise K 32Bit OS를 사용하였으며, 개발 툴로는 Visual Studio 2010의 MFC환경과 OpenCV 2.4.9를 사용하였다. 하드웨어로는 SoftKinetic사의 DS325 적외선 조명 카메라를 사용했다. DS325 특징상 근거리에서밖에 인식이 안되므로, 카메라와 손 사이간 거리는 10~20cm수준에서 인식이 가능하였다. Intel사의 i5-3470 CPU와 3.47GB 메모리로 구성된 컴퓨터로 개발 하였다. 손끝을 검출하고 추적하는데 있어 본 시스템은 60 fps(frames per second)를 보였다. 제스처 매칭으로는 8개의 한자 템플릿을 실험을 위해 사용하였으며, 샘플링 점들은 최소 48개로부터 최대 96개로 구성되었다. 매칭 시간은 46~54 msec가 걸렸다. 매칭시간은 스트로크의 수가 많을수록 증가하였다. 48개의 샘플링 점의 경우 최소 시간인 46msec가 걸렸지만 최대인 96개의 점이 있을 경우 54msec의 시간이 걸렸다. [그림 12]는 시스템의 전반적인 프로세스를 나타낸다.

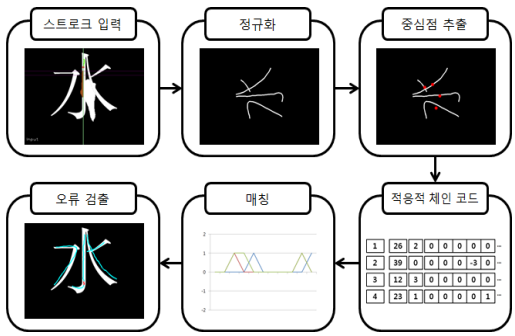

그림 12. 적응적 체인 코드를 이용한 매칭 시스템

[그림 13]에서는 한자 학습을 위한 인터페이스를 나타낸다. 우선 한자를 화면에 보여준다. 그 다음 입력 할 부분에 대해 순서대로 표시를 해준다. 이를 통해 사용자가 어떤 순서로 그려야 할지 쉽게 알 수 있다.

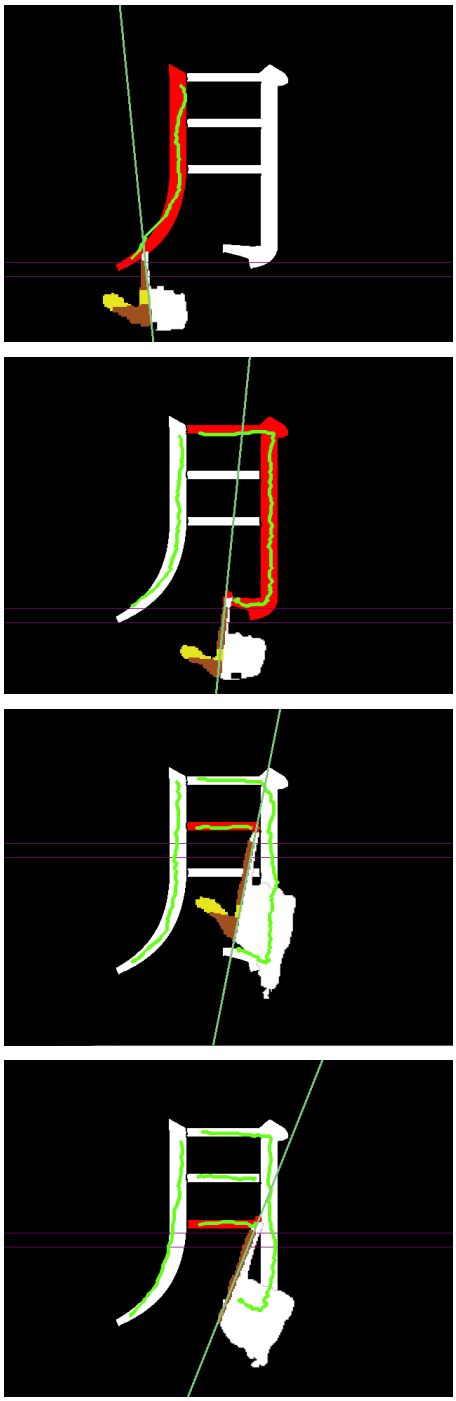

그림 13. 사용자의 한자 학습을 위한 인터페이스

[그림 14]에서는 사용자가 입력한 패턴과 저장된 템플릿 패턴을 매칭 후 오류가 난 부분을 검출한 결과이다. [그림 14]에서 하나의 스트로크 전체가 오류가 난 부분이 있는데, 이 부분은 인위적으로 거꾸로 입력한 결과다. 또한 다른 부분에서 흔들림 때문에 오류가 난 부분도 존재한다. 木에서 2번째 획에서 흔들리는 부분은 다른 색상으로 표기되어 오류 표시를 하였으며, 3번째 획은 거꾸로 입력하여 입력 순서가 잘못되었음을 나타낸다.

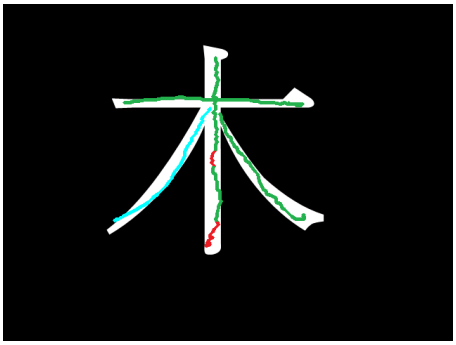

그림 14. 오류 검출 결과

2. 결론

본 논문에서는 적응적 체인 코드를 이용한 실시간 한자 학습 시스템을 제안하였다. 제스처 입력 시스템으로는 사용자가 검지를 통해 입력한다고 가정하였고, 이를 이용한 맞춤형 입력 시스템을 설계하였다. 또한 스트로크의 시작과 끝을 알기 위해 엄지의 상태 인식을 제안하였다. 이러한 입력 방법을 사용자의 학습을 위해 본 논문에서는 적응적 체인 코드와 이를 이용한 매칭 방법에 대해 제안하였다. 실험 결과에서 보듯 이 방법은 사용자가 학습하는데 있어 어느 부분이 잘못되어 있는지 알려줄 수 있기 때문에 매우 적합한 방법임을 보여주었다.

참 고 문 헌

[1] Robert J. Elliott, Lakhdar Aggoun, and John B. Moore, *Hidden Markov Models*, Springer, 1995.

[2] M. Müller, "Dynamic time warping," Information retrieval for music and motion, pp.69-84, 2007.

[3] T. Starner and Alex Pentland, "Real-time american sign language recognition from video using hidden markov models," Motion-Based Recognition, Springer Netherlands, pp.227-243, 1997.

[4] A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001, Proceedings. IEEE ICCV Workshop on. IEEE, 2001.

[5] G. A. Ten Holt, M. J. T. Reinders, and E. A. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," Thirteenth annual conference of the Advanced School for Computing and Imaging, Vol.119, 2007.

[6] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," Proceedings of the 20th annual ACM symposium on User interface software and technology, ACM, 2007.

[7] L. Anthony and J. O. Wobbrock, "$N-Protractor: A fast and accurate multistroke recognizer," Proceedings of Graphics Interface (GI '12), Toronto, Ontario, Toronto, Ontario: Canadian Information Processing Society, pp.117-120, 2012.

[8] Vatavu, Radu-Daniel, Lisa Anthony, and Jacob O. Wobbrock, "Gestures as point clouds: a $ P recognizer for user interface prototypes," Proceedings of the 14th ACM international conference on Multimodal interaction, ACM, 2012.

[9] Y. Li, Protractor: A fast and accurate gesture recognizer, Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '10). Atlanta, Georgia (April 10-15, 2010), New York: ACM Press, pp.2169-2172, 2010.

[10] L. Anthony and J. O. Wobbrock, A lightweight multistroke recognizer for user interface prototypes, Proceedings of Graphics Interface (GI '10), Ottawa, Ontario (May 31-June 2, 2010), Toronto, Ontario: Canadian Information Processing Society, pp.245-252, 2010.

[11] Mitra, Sushmita and Tinku Acharya, "Gesture recognition: A survey," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 37.3, pp.311-324, 2007.

[12] Y. Wu and T. S. Huang, "Vision-based gesture recognition: A review," Gesture-based communication in human-computer interaction, Springer Berlin Heidelberg, pp.103-115, 1999.

[13] H. K. Lee and J. H. Kim, "An HMM-based threshold model approach for gesture recognition," Pattern Analysis and Machine Intelligence, IEEE Transactions on 21.10. pp.961-973, 1999.

저 자 소 개

김 기 상(Kisang Kim) 정회원

- 2007년 2월 : 숭실대학교 컴퓨터학부(공학사)
- 2009년 2월 : 숭실대학교 미디어학과(공학석사)
- 2014년 3월 ~ 현재 : 숭실대학교 미디어학과 박사과정

<관심분야> : 컴퓨터 비전, 패턴인식, 영상처리

최 형 일(Hyung-Il Choi) 정회원

- 1972년 : 연세대학교 전자공학과 (공학사)
- 1982년 : 미시간대학교 전자공학과(공학석사)
- 1987년 : 미시간대학교 전자공학과(공학박사)
- 1995년 : 퍼지 및 지능시스템학회 이사
- 1996년 : 정보과학회 컴퓨터비전 및 패턴인식 연구회 위원장
- 1997년 : IBM Waston Lab 방문연구원
- 2005년 : 한국정보과학회 이사
- 현재 : 숭실대학교 미디어학과 교수

<관심분야> : 컴퓨터공학, 컴퓨터 비전, 패턴인식, 영상처리