개선된 Line Following 방식의 세선화 알고리즘

조영원°, 한상훈, 조형제
동국대학교 컴퓨터공학과

A Modified Thinning Algorithm Using Line Following

Cho Youngone, Han Sanghoon, Cho Hyungje
Department of Computer Engineering, Dongguk University

요 약

기존의 Line Following 알고리즘은 곡선으로 이루어진 영상 패턴을 세선화할 때 두꺼운 분기점을 효과적으로 처리하지 못할 뿐 아니라 폐곡선을 형성하는 부분이 끊어지는 단점이 있어 선분 형태 이외의 일반적인 문자나 이미지 등에 적용하기 어려우므로 Line Following 방식에 근거를 둔 개선된 새로운 세선화 알고리즘을 제안한다.

본 연구에서는 두꺼운 분기점의 문제를 해결하기 위해 선의 모양에 따라 동적으로 변하는 윈도우의 크기를 일정 비율로 조절하고, 폐곡선을 형성한 부분에서는 분기점마다 특정한 tag를 두어 선의 끝을 결정하는 단계에서 tag와 만나는 점에 대해 별도의 처리를 하였다. 이 알고리즘은 기존 알고리즘과 비슷한 처리 속도를 유지하면서도 기존 알고리즘의 단점을 효과적으로 개선하여 곡선이나 복잡한 영상 외에 문자 영상 등에 대해서도 좋은 결과를 보여 주었다

I. 서 론

영상 처리, 형태 인식, 컴퓨터 비전 분야의 전처리 및 특징 추출하는 단계에서 곡선으로 이루어진 영상 패턴의 세선화 방법은 자주 이용되고 있다. 이러한 세선화 알고리즘은 과거 30 년 전부터 꾸준히 연구되어 왔으며 근래에는 영상 압축의 한 방법으로도 연구되고 있다[3].

패턴을 세선화하여 표현하는 이유는 형태 분석을 쉽게 할 수 있다는 이점뿐만 아니라 데이터 량을 최소화시켜야 할 필요성이 있기 때문이다. 문자의 경우 세선화된 패턴이 그 패턴에 대한 인간의 인지 개념에 더 가깝기 때문에 보다 간단한 구조 분석과 직관적인 인식 알고리즘을 설계할 수 있게 해준다[1].

세선화 방법에는 반복적 화소제거 방법과 비 반복적 방법이 있는데 본 연구에서는 비 반복적 방법의 하나인 Line Following 알고리즘의 문제점을 효과적으로 개선한 방법을 제안한다. Line Following은 처리 속도면에서 다른 알고리즘에 비해 훨씬 우수하고, 지역적 정보만을 이용하는 다른 방법에 비해 비교적 전역적 정보를 이용하여 잡음에 덜 민감하다는 장점이 있다[4].

그러나 두꺼운 선으로 된 분기점을 처리하기 어렵고, 폐곡선 형태를 가진 부분에서는 끊어지는 현상이 발생할 수 있다. 윈도우의 크기가 동적으로 변할 때 어느 한 방향으로만 윈도우가 커짐으로서 두꺼운 분기점인 경우에 정확히 분기점임을 인식하지 못하게 되는데, 이때 윈도우의 크기를 일정한 비율로 양방향으로 조절하여 곡선의 형태를 정확하게 인식할 수 있도록 배려해 줄 필요가 있다. 그리고 폐곡선을 형성하는 경우에는 선의 끊김이나 교차 현상이 발생할 수 있다. 이는 분기점마다 특정한 tag를 두어 라인의 끝을 결정하는 과정에서 tag와 만나는 점에 대해 별도의 처리를 함으로서 가능하다. 또한 곡선으로 된 영상에서는 그리 중요하지는 않지만 다른 영상에 적용할 경우를 고려하여 시작점의 정의를 단순화하여 할 필요가 있다.

이와 같이 개선된 세선화 방법은 곡선으로 된 지문과 같은 복잡한 영상에 대해서만 적용하던 Line Following 알고리즘을 문자와 같은 영상에도 적용할 수 있음을 보인다.

다음에서 기존 Line Following 알고리즘에 대해서 소개하고, 본 논문이 제안하는 방법을 설명한다. 이어서 이에 대한 실험 결과를 검토하고 결론을 맺는다.

II. 본 론

1. Line Following 알고리즘

Line Following 알고리즘은 인간이 눈으로 곡선을 따라가면서 주요 골격을 파악하는 원리를 이용한 것으로 곡선을 따라가면서 잡음과 같은 미세한 변화는 무시하고 선의 대략적인 중앙점들을 연결해 가며 끝을 찾는다.

Line Following 알고리즘에서 활용하는 4가지 기본 요소에 대한 의미를 먼저 정의한다.

- LP : 윈도우 경계와 만나는 곡선의 왼쪽 Edge 점
- RP : 윈도우 경계와 만나는 곡선의 오른쪽 Edge 점
- W : 곡선과 평행한 윈도우
- d : 윈도우 경계와 LP, RP와의 거리로 거의 2로 설정

알고리즘의 처리과정은 다음과 같다.

먼저 LP, RP를 둘러싼 윈도우를 만든다. 이 때 윈도우의 크기는 LP, RP를 포함하고, 윈도우 경계가 LP, RP중에서 가까운 점과의 거리를 2로 유지하도록 한다. 그림 1에서 LP, RP점이 잘 나타나 있다.

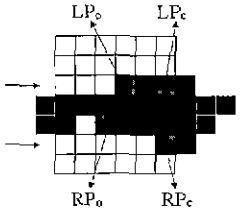


그림 1 LP, RP 점의 예

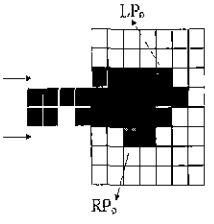


그림 2 곡선의 끝

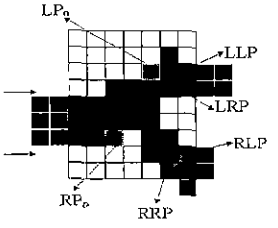


그림 3 분기점인 경우

아진 LP, RP(LPo, RPo)에서 곡선의 에지(edge)를 따라가서 윈도우 경계와 만나는 왼쪽 점을 LP(LPc), 오른쪽 점을 RP(RPc)로 하고 이 점을 이용하여 다음 윈도우를 만든다.

검색된 곡선에 대한 화소(pixel)는 삭제되고 LP, RP를 이용하여 순환적으로 곡선의 에지(edge)를 계속 따라간다.

여기서 한 윈도우 안에서 생기는 곡선의 형태는 곡선이 윈도우를 통과하는 경우, LP와 RP가 만나서 곡선이 끝나는 경우, 분기점(Branch)이 발생하는 경우로 나눌 수 있다.(그림1,2,3) 분기점에서는 왼쪽 가지의 LP를 LLP, RP를 LRP로 두고 오른쪽 가지의 LP를 RLP, RP를 RRP로 두어 왼쪽 가지부터 차례로 호출한다. 각 윈도우 내에서 LP, RP의 중점을 연결한 곡선이 세선화한 결과값이 되고 그림 2와 같이 곡선이 끝날 때 한 곡선의 세선화가 완료된다.

2. 개선된 Line Following 알고리즘

본 연구에서는 기존 알고리즘에서 처리할 수 없었던 두꺼운 곡선으로 된 분기점에서 발생하는 문제와 폐곡선 형태에서의 문제점을 해결하고, 시작점 지정 방법을 단순화하였다.

(1) 윈도우의 크기 비율 조정

기존 방법에서는 윈도우의 크기와 LP, RP와의 거리를 일정한 값으로 지정한 것을 본 연구에서는 윈도우의 크기에 가로/세로의 비를 구하여 정사각형 형태의 윈도우가 유지되도록 하였다. 이것은 윈도우가 가로 혹은 세로 한 방향으로만 확장되는 것을 방지하여 두꺼운 곡선으로 형성된 분기점도 인식하도록 하기 위함이다. 즉, 윈도우의 가로/세로 비율이 2에서 1/2사이에 존재하도록 한다.

먼저 윈도우의 가로/세로 비율을 구한다.

$$2 \le Ratio(W) \le \frac{1}{2}$$

*Ratio(W)*가 범위를 벗어나면 아래와 같이 조정한다.

If *Width* >= 2*Height* Then *Height* = 0.5*Width*
If *Height* >= 2*Width* Then *Width* = 0.5*Height*

윈도우의 크기를 LP, RP점에만 의존한다면 문자 영상에서 원래 모양을 파악할 수 없게 되는 경우가 자주 발생한다. 예를 들어 그림 4-(a)와 같이 "I" 모양인 경우에 밑 부분에서 (b)와 같이 윈도우의 크기가 가로 방향으로만 확대되어 중심점이 항상 같은 값을 가져 밑에 있는 선이 없어진다. 제안한 방법으로 윈도우의 크기 비율을 조절하면 (c) 와 같은 결과를 얻을 수 있다.

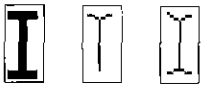


그림 4. "I" 를 세신화 한 결과

(2) 폐곡선 문제

분기점의 가지를 호출하기 전에 모든 LLP, LRP와 RLP, RRP를 연결한 직선을 특정한 tag로 둔다.(그림 5) 곡선을 따라가던 한 점이 분기점에서 표시한 tag값을 가진 한 점과 만날 경우 현재 윈도우에서 곡선이 끝나는 경우로 처리한다. tag 값을 따라 tag의 끝점인 RRP와 RLP를 찾아 RRP와 RLP의 중점과 현재 윈도우 내의 중점을 연결한다.

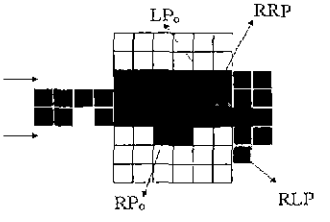


그림 5 분기점 Tag

그림 5와 같이 tag를 이용하면, 폐곡선에서 연결되는 부분이 LP, RP와 RLP, RRP가 서로 일치하지 않더라도 곡선을 정확하게 연결해 줄 수 있다.

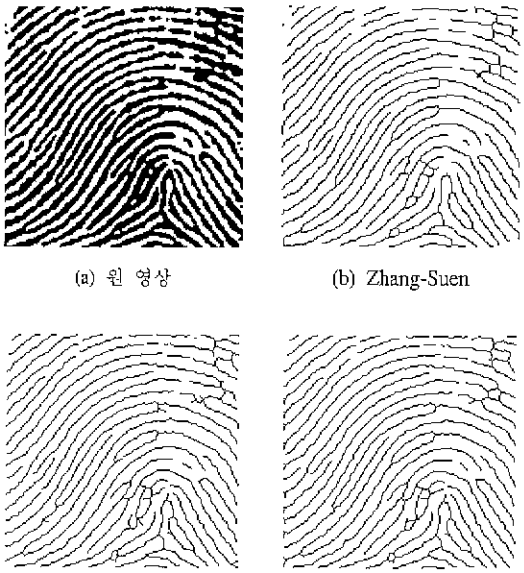
(3) 시작점의 지정

첫 LP, RP점을 지정할 때 무조건 한 점을 LP, RP점으로 둔다. 즉 첫 점을 찾고 연결된 곡선의 방향을 찾을 필요가 없이 처음 만나는 점을 LP, RP점으로 둔다. 이 경우 첫 점을 찾을 때 곡선의 방향을 찾을 필요가 없고 특히 굵은 문자 영상에 대해서도 윈도우가 서서히 커지면서 골격선이 전체 영상과 유사한 형태가 된다.

Ⅲ. 실험 결과 및 분석

본 실험에서 사용한 PC는 CPU가 펜티엄 MMX 200이고, 윈도우 98상에서 Visual C++5.0을 이용하여 실험하였다. 사용된 영상의 크기는 지문 영상 175*175, 문자 영상 300*300 이다.

제안된 방법을 패턴 인식에서 전처리 과정으로 세선화를 많이 적용하고 있는 문자 인식, 지문 인식에 활용가능한지 시험한 결과 지문에 대해서는 효과적이며, 문자에 대해서도 대체로 양호한 것으로 나타났다.



(a) 원 영상 (b) Zhang-Suen

(c) 기존 방법 (d) 제안된 방법

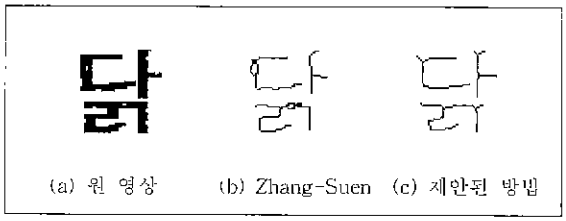
그림 6 지문 영상에 대한 실험 결과

(a) 원 영상 (b) Zhang-Suen

(c) 기존 방법 (d) 제안된 방법

그림 7 문자 영상에 대한 실험 결과

그림 6은 비교 대상으로 많이 사용되는 회소 제거 방식인 Zhang-Suen 알고리즘[6], 기존 방법, 제안된 방법을 사용하여 지문 영상을 실험한 결과이다. 영상의 곡선에 내린 특징을 최대한 반영하면서 선의 굴곡과 연결성에서 제안된 방법이 좋음을 알 수 있다 그림 7은 문자 영상에 대한 실험결과이다. 화소 제거 방법과 비교하여 정확성에서 크게 다르지 않은 결과를 보이고 있다 기존 방법을 이용하면, 폐곡선 부분에서 잘못된 세선화가 나타날 수 있는데 제안된 방법에서는 폐곡선에서도 정확한 결과를 얻을 수 있었다 그림 8은 갑음이 있는 영상에 대한 결과로 회소 제거 방법에 비해 잡음에 강함을 알 수 있다



(a) 원 영상 (b) Zhang-Suen (c) 제안된 방법

그림 8 섭음을 가진 영상에 대한 실험 결과

표 1 수행 시간 비교 (단위 : 초)

시험 방법	Zhang-Suen	제안된 방법
10개의 영상에 대해 10회 반복하여 평균	0.23	0.08

표 1은 지문, 문자 영상의 평균 수행 시간을 나타낸다 제안된 방법이 반복적 화소 제거 알고리즘에 비해 약 3배정도 처리속도가 빠르다는 것을 알 수 있다. 기존 알고리즘에 비해 Overhead를 가지고 있음에도 불구하고 처리속도가 떨어지지 않았다 지문 영상에서 좋은 성능을 보이는 Line Following 알고리즘을 본 연구에서는 다른 일반적인 영상에 대해서도 빠르고 정확하게 사용할 수 있도록 개선하여 좋은 결과를 보이고 있다

Ⅳ. 결 론

세선화에 대해 처리 속도와 정확성을 동시에 만족하기는 어렵지만 본 연구에서 제안한 개선된 Line Following 알고리즘을 이용하여 기존 알고리즘과 비슷한 처리 속도로 더 정확한 결과를 얻을 수 있었다 무엇보다 기존의 Line Following 알고리즘이 복잡하고 사이즈가 큰 영상에 대해서 주로 사용되었지만 개선된 Line Following 알고리즘을 이용해서 문자나 숫자 이미지 등의 일반적인 패턴 인식에도 적용할 수 있게 되었다

차후 영상의 왜곡을 최소화시키는 방법과 그레이 영상에 대한 세선화방법이 연구되어야 할 것이다

참고문헌

[1] 이성환, 문자 인식 -이론과 실제-

[2] Christian NEUSIUS, JAN OLSZEWSKI, "A Noniterative Thinning Algorithm", ACM Trans. on Mathmetical Software, Vol 20, No 1, pp 5-20, 1994

[3] Louisa Lam, Seong-Whan Lee, Ching Y Suen, "Thinning Methodologies - A Comprehensive Survey", IEEE Trans PAMI, Vol 14, No 9 Sept, 1992, pp 869-885

[4] Ont BARUCH, "Line thinning by line following", Pattern Recognition Letters 8, pp 271-276, 1988

[5] CARLO ARCELLI, GABRIELLA SANNITI DI BAJA, "A Width-Independent Fast Thining Algorithm", IEEE PAMI Vol 7, No 4, pp 463-474, 1985

[6] T Y Zhang and C Y Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns", Comm of the ACM, Vol 27, No 3, 1984, pp 236-239