잡영블랍 검출에 의한 잡영가지 제거 방법의 개선

김성옥[†] · 임은경[††] · 김민환[†††]

요 약

어떤 물체 영역의 골격(skeleton)을 얻기 위해 세선화(thinning)하는 과정에서 잔가지(parasitic branch)가 발생하므로, 이러한 것을 효과적으로 제거하기 위한 여러 가지 연구가 이루어져 왔다. 이중에서 잔가지가 한 픽셀 두께의 가지로 나타나는 속성에 착안하여, 윤곽선 추적에 의한 대칭 경로(symmetric path)를 검출함으로써 잔가지를 제거하는 방법이 매우 효과적인 것을 알 수 있었다. 본 연구에서는, 이 방법을 영상 분할이나 연결 요소 추출 등에 의해 구해진 물체 영역의 윤곽선 부분에 나타나는 잡영가지(noise branch)를 제거하는데 활용할 수 있도록 개선한 방법을 제안한다. 즉, 한 픽셀 두께의 잡영가지 뿐만 아니라, 부분적으로 두 픽셀 이상이 뭉쳐져 둥그스름한 덩어리(잡영블랍, noise blob)를 형성하고 있는 잡영가지도 제거할 수 있는 개선된 방법을 제안한다. 대칭 경로를 찾기 위해 4-8방향 윤곽선 추적 알고리즘을 이용하며, 잡영블랍이 포함된 경우에는 준대칭(quasi-symmetric) 경로를 정의하여 추출한다. 제안한 방법의 시간 복잡도는 윤곽선 픽셀수의 선형 함수로 표현되며, 사용자가 잡영블랍과 잡영가지의 크기를 임의로 설정하도록 하여 융통성있고 다양하게 잡영가지를 제거할 수 있도록 하였다. 실제 형상과 인위적 형상에 대한 실험을 통해 제안된 방법의 유용성을 확인할 수 있었다.

An Enhancement of Removing Noise Branches by Detecting Noise Blobs

Seongok Kim[†], Eunkyung Lim[††] and Minhwan Kim[†††]

ABSTRACT

Several methods have been studied to prune the parasitic branches that cause unfortunately from thinning a shape to get its skeleton. We found that the symmetric path finding method was most efficient because it followed the boundary pixels of the shape just once. In this paper, its extended method is proposed to apply to removing the noise branches that protrude out of the boundary of a segmented or extracted shape in a given image. The proposed method can remove a noise branch with one-pixel width and also remove the noise branch that includes a round shape called a noise blob. The method uses a 4-8-directional boundary-following technique to determine symmetric paths and finds noise branches with noise blobs by detecting quasi-symmetric paths. Its time complexity is a linear function of the number of boundary pixels. Interactively selectable parameters are used to define various types of noise branches flexibly, which are the branch-size parameter and the blob-size parameter. Experimental results for a practical shape and various artificial shapes showed that the proposed method was very useful for simplifying the shapes.

Key words: 윤곽선 추적(boundary following), 잡영가지(noise branch), 잔가지(parasitic branch), 세선화(thinning)

접수일 : 2002년 9월 2일, 완료일 : 2002년 12월 16일
[†] 종신회원, 한국신발피혁연구소 자동화연구부 연구부장
[††] 준회원, 부산대학교 대학원 컴퓨터공학과 박사과정
[†††] 종신회원, 부산대학교 컴퓨터공학과 교수

1. 서 론

형상 표현법(representation)은 영상 처리와 패턴 인식 분야에서 매우 중요한 역할을 하기 때문에, 주어진 형상의 전체 화소 또는 형상의 윤곽선(boundary)을 기반으로 한 여러 가지 표현법들이 연구되어 왔다[1,2]. 그러나, 형상으로부터 돌출된 잡영가지(noise branch)로 인해, 이러한 표현법들을 형상 분석 및 인식 과정에 활용하는데 공통적으로 많은 어려움을 겪고 있다.

잡영가지는 주어진 형상에 대해 잘못된 골격(skeleton)을 생성하거나 또는 세선화(thinning) 과정에서 불필요한 잔가지(parasitic branch)를 생성하는 원인이 되기 때문에, 특히 형상의 골격화 및 세선화 연구 분야에서 이러한 잡영가지에 의해 세선화 결과에 영향이 미치는 것을 최소화 하기 위한 방법들이 많이 연구되었다[3]. 먼저 세선화 과정에서 잡영가지의 영향을 줄이기 위해 근사화(approximation)를 통해 주어진 형상을 단순화하는 방법이 있다. 예를 들어, 다각형으로 근사화 한 후에 세선화를 하든지 또는 다해상도(multi-resolution) 스케일 스페이스 기법으로 잡영가지에 의한 영향을 덜 받으면서 세선화하는 방법이 있다[4]. 그러나, 이 방법들은 보다 유용한 세선화 결과를 얻는 것을 목적으로 하는 것으로서, 본래의 주어진 형상에서의 잡영가지를 제거하는 방법으로 볼 수 없다. 다른 접근 방법으로서, 세선화 과정에서 얻어지는 정보를 이용하여 세선화 결과에서의 잔가지를 제거하는 방법[5]이 있으며, 또한 세선화 과정에서의 정보를 사용하지 않고 즉, 세선화 과정과는 별도로 세선화 결과에서의 잔가지를 모폴로지(morphology) 연산 기법을 이용하여 잔가지를 제거하는 방법[6]도 제안되어 있다. 이 방법들은 세선화 결과 즉, 한 픽셀 두께로 주어지는 형상에 대해서만 불필요한 가지를 제거하는 것으로서, 임의 형상의 경계로부터 돌출된 형태의 잡영가지를 제거하는 데에는 직접 활용할 수 없다.

한편, 임의 형상에서의 잡영가지를 제거하기 위한 방법으로서, 3×3 크기의 윈도우 내에서의 물체 픽셀의 개수가 주어지는 임계값 이하이면 제거하는 방법[7]이 제안되었다. 그러나, 이 방법은 지역적(local) 정보만을 이용하여 제거 여부를 결정하기 때문에 물체의 연결 정보를 훼손할 수 있으며, 잡영가지 일부분이 제거되지 않는 문제점이 있다. 이와 유사한 개념의 방법으로서, 모폴로지 오프닝 연산을 이용하여 돌출된 잡영가지를 제거할 수도 있다. 그러나, 이 방법도 지역적 정보에 의존해 제거함에 따라 연결 정보를 잃을 수 있으며, 또한 구조화 요소(structuring element)의 형태에 따라 부적합한 제거 결과를 보이기도 한다. 반면에, 물체의 윤곽선을 이루는 픽셀들의 시퀀스(sequence)에서 중복 사용되는 픽셀들을 제거하는 방법[8]에서는 주어진 물체 형상의 전체적인 구조적 정보를 사용하여 돌출된 잡영가지를 제거할 수 있다는 개념을 제안하였다. 그러나, 이 방법에서는 단순히 픽셀의 중복 사용 여부에 대한 정보만을 이용함에 따라, 돌출 잡영가지의 지역적 구성 특성을 활용하지 못하고 있을 뿐만 아니라 형상의 연결 정보도 제대로 보존하지 못하는 문제점을 보이고 있다. 이러한 문제점을 보완한 방법으로서 윤곽선 시퀀스에서의 부분적인 대칭 경로(symmetric path)를 찾아 제거하는 방법[9]이 제안되었다. 이 방법은 물체 형상의 연결 정보를 그대로 유지하면서 돌출된 잡영가지를 효과적으로 제거할 수 있다. 그러나, 한 픽셀 두께의 돌출 가지만을 제거할 수 있어, 실제 응용에서 돌출된 잡영가지의 일부에 덩어리진 부분이 있을 경우에는 적용하지 못하는 문제점이 있다.

일반적으로, 임의 형상에서의 잡영가지를 명확히 정의하는 것은 쉬운 일이 아니다. 우리 사용자는 주어진 형상의 형태에 따라 같은 모양의 돌출 가지를 잡영가지로 취급하기도 하고 반대로 몸체의 일부로 취급하기도 하기 때문이다. 예를 들어, 그림 1(a)에서의 다섯 픽셀로 구성된 돌출 가지는 전체를 잡영가지로 볼 수 있으나, 그림 1(b)의 같은 형태에 대해서는 한 픽셀 두께로 돌출된 세 픽셀만을 잡영가지로 정의하는 것이 보다 적합해 보인다. 이에 따라, 잡영가지를 일률적으로 정의하는 것은 매우 어려운 일이다. 윤곽선 추적시의 대칭 경로(symmetric path)를 갖는 돌출 가지만을 제거하는 방법[9]에서는 한 픽셀 두께로 뻗어나간 돌출 가지만을 잡영가지로 정의한다. 이 방법은 세선화 결과에서의 잔가지 제거에 활용할 수 있으면서 제한적이나마 그림 1(c)에서와 같이 한 픽셀 두께의 돌출 가지도 제거하는데 활용할 수 있다. 그러나, 이 방법으로는 잡영에 의해 그림 1(c)에서의 위 부분의 둥그스름한 형태의 작은 덩어리(blob)가 포함된 돌출 가지는 제거하지 못하여 그림 1(d)에서와 같은 결과를 보인다. 이 부분에서는 윤곽선 추적시에 대칭 경로를 형성하지 못하기 때문이다.

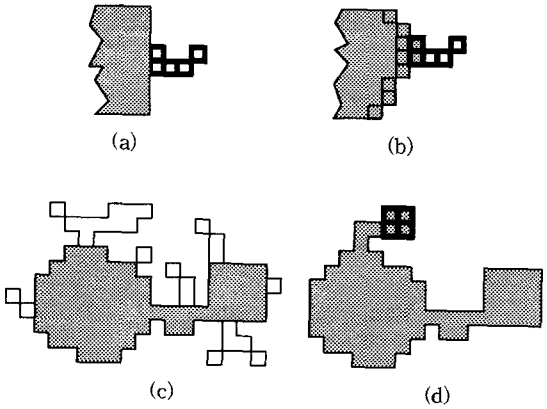


그림 1. 잡영가지 정의의 어려움 및 잡영블랍이 포함된 돌출 가지의 예

본 연구에서는 그림 2(c)-(e)에서와 같이 소수의 픽셀로 구성된 작은 덩어리(잡영블랍, noise blob)을 포함하는 돌출 가지도 윤곽선 추적에 의해 효과적으로 제거할 수 있는 방법을 제안한다. 즉, 몸체로부터 한 픽셀 두께로 뻗어나온 돌출 가지로서 일정 크기 이하의 잡영블랍을 포함하는 것을 준대칭(quasi-symmetric) 경로를 정의하여 윤곽선 추적에 의해 효과적으로 제거할 수 있는 방법을 제안한다. 제안한 방법은 사용자가 원하는 모든 돌출 가지를 정확히 한꺼번에 제거해 주지는 못할지라도 어느 정도 불필요하다고 생각되는 돌출 가지를 자동적으로 제거해 줄 수 있는 방법으로서, 일종의 필터링(filtering) 연산으로 활용이 가능하다.

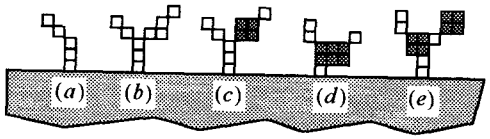


그림 2. 돌출된 잡영가지의 예

본 논문에서의 2장에서는 4-8-방향 윤곽선 추적 알고리즘을 이용한 잡영가지 제거의 기본 원리를 다루고, 3장에서는 잡영블랍에 대한 정의와 이것을 포함하고 있는 돌출 가지를 잡영가지로 정의하기 위한 준대칭 경로에 대한 정의를 소개한다. 4장에서는 이러한 잡영가지를 제거하기 위한 알고리즘을 소개하고, 5장에서는 실제 형상과 인위적으로 만들어진 다양한 형상들에 대한 실험을 통해 제안한 방법의 효용성을 보인다.

2. 윤곽선 추적 알고리즘을 이용한 잡영가지 제거의 기본 원리

주어진 형상의 윤곽선을 4-8-방향 윤곽선 추적(4-8-Directional Boundary Following, 4-8-DBF) 알고리즘에 의해 추적할 때, 대칭 경로를 검출함으로써 잡영가지를 결정한다.

2.1 4-8-방향 윤곽선 추적 알고리즘

4-8-DBF 알고리즘은 시계 방향으로 8-방향 윤곽선 추적 방법[1]을 기본적으로 사용한다. 그러나, 8-연결 픽셀과 바로 인접한 4-연결 픽셀이 있으면, 4-연결 픽셀로 우선적으로 추적해 간다. S와 B를 각각 주어진 형상을 이루는 픽셀들의 집합과 그 여집합(배경 영역)이라고 한다면, 4-8-DBF에 의해 찾은 윤곽선은 그림 3에서 보듯이 B와 4-인접 또는 8-인접하는 S의 픽셀들의 집합이다. 윤곽선 픽셀들의 집합을 C라고 하면 S와 C의 차집합, $S-C$는 형상의 내부를 나타내며, 이 논문에서는 내부 픽셀 집합을 I로 표현한다. 본 논문에서는 형상의 내부에 홀이 없는 것으로 간주한다.

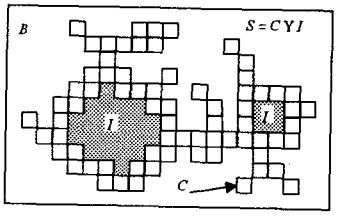


그림 3. 테스트 물체의 물체 집합(S), 배경 집합(B), 내부 집합(I), 윤곽선 집합(C)

4-8-DBF 알고리즘은 다음과 같다.

1. 주어진 형상에 대해서, 영상의 위에서 아래로, 좌에서 우로 픽셀들을 스캔하여 시작점의 픽셀 s(∈S)를 찾는다.
2. 윤곽선 추적에서 현재 픽셀을 c라고 할 때, c = s라고 설정하고, s의 왼쪽으로 4-이웃 방향에 위치하는 픽셀을 b(∈B)로 설정한다.
3. c를 중심으로 b부터 시계 방향 순서로 여덟 개의 4-이웃 또는 8-이웃 픽셀들 $n_1, n_2, \cdots, n_8$을 살펴보아, S에 포함되는 첫번째 픽셀 n_i를 찾는다.
4. 만약 n_i가 c의 8-이웃 픽셀이고 $n_{i+1} \in S$이라면, $c = n_{i+1}$와 $b = n_i$로 설정한다. 그렇지 않은 경우에는 $c = n_i$와 $b = n_{i-1}$로 설정한다.
5. 단계 3과 4를 c = s가 될 때까지 반복한다.

2.2 잡영가지 결정하기

그림 4는 세 개의 잡영가지 예를 보여주고, 표 1에서는 이것의 시계 방향의 8-DBF와 4-8-DBF 추적 결과를 보여준다. 그림 4(a),(b)에서의 잡영가지에 대한 4-8-DBF 결과는 대칭 경로를 가진다는 것을 알 수 있다. [9]의 잡영가지 제거방법은 그림 4(a)에서 잡영가지에 대한 대칭 경로 (2,3,4,5,4,3,2)를 검출하여 픽셀 3,4,5를 제거한다. 4-8-DBF을 이용하면 그림 4(b)에서 또 다른 대칭 경로 (2,3,4,5,6,5,4,3,2)를 검출할 수 있으나, 그림 4(c)에서는 어떠한 대칭 경로도 찾지 못함을 알 수 있다. 이에, 본 논문에서는 네 픽셀 4,5,6,7을 하나의 잡영블랍으로 간주함으로서 준대칭 경로 (2,3,[4,5,6,7,4],3,2)를 검출하여 잡영가지로 검출해내는 방법을 제안한다.

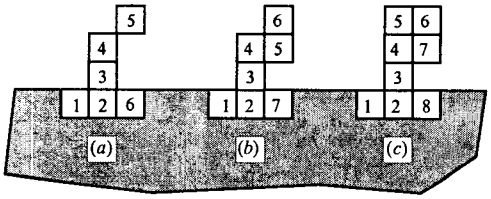


그림 4. 잡영가지의 예제

표 1. 그림 4의 잡영가지의 8-DBF와 4-8-DBF 추적 결과

	8-DBF	4-8-DBF
(a)	1,3,4,5,4,3,6	1,2,3,4,5,4,3,2,6
(b)	1,3,4,6,5,3,7	1,2,3,4,5,6,5,4,3,2,7
(c)	1,3,4,5,6,7,3,8	1,2,3,4,5,6,7,4,3,2,8

3. 잡영가지의 정의

3.1 잡영블랍의 정의

그림 2,4에서 보듯이 윤곽선 집합 C에 잡영블랍을 포함하는 돌출 가지들이 자주 나타나는데, 본 논문에서는 잡영블랍을 다음 조건들을 만족하는 하나의 픽셀들의 집합, NB로 정의한다.

(조건1_{NB}) $NB \subset C$
(조건2_{NB}) NB의 원소 개수 ≤ 6
(조건3_{NB}) 각 픽셀의 8-연결성 정도 ≥ 3

각 픽셀의 8-연결성 정도는 그 집합 내에 있는 픽셀들 중에서 그 픽셀의 8-이웃이 되는 픽셀 개수이다. 픽셀 집합들의 예를 그림 5에서 볼 수 있는데, 박스 안의 숫자는 8-연결성 정도를 나타내고 빗금친 박스는 윤곽선 픽셀이 아닌 것을 의미한다. 여기에서, 잡영블랍은 단지 두 가지 형태로 나타남을 알 수 있는데, 그림 5에서는 회색 박스로 나타내었다.

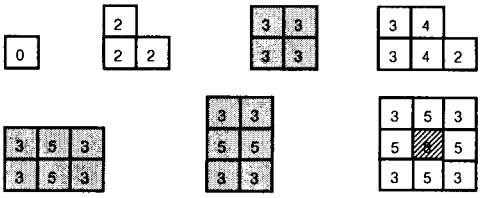


그림 5. 잡영블랍과 비잡영블랍의 예

3.2 잡영가지의 정의

영상처리 환경에 따라 잡영가지를 다르게 정의할 필요가 있기 때문에 잡영가지를 정형적으로 서술하기는 어렵다. 그러므로, 몇 가지 매개변수가 항상 이용되는데, 이 매개변수의 값은 사용자들이 편리하게 결정할 수 있어야 한다. 1장에서 언급한 방법[6,9]에서는 오직 한 픽셀 두께로 된 돌출 가지만을 잡영가지로 간주하였기 때문에, 돌출 가지의 최대 허용 길이를 나타내는 하나의 매개변수만을 사용하였다. 그러나, 본 논문에서는 그림 1(c)와 그림 2에서와 같은 형태의 돌출 가지를 제거해야 하므로, 여러 가지 제약 사항들을 정의하여 사용하여야 한다.

먼저 다음 조건들을 만족하는 픽셀을 분기점 (branch point) p라고 정의한다.

(조건1_{BP}) $p \in C$
(조건2_{BP}) p의 0-1-변환 정도 ≥ 3

어떤 픽셀의 0-1-변환 정도는 그 픽셀의 여덟 개의 이웃 픽셀들을 시계방향 순서로 추적할 때, 0에서 1로 변환되는 회수를 나타낸다. 여기에서, 0은 배경 픽셀값을, 1은 물체 픽셀값을 의미한다. 그러므로 분기점의 0-1-변환 정도는 그 점으로부터 분기되는 가지들의 개수를 나타낸다. 그림 6은 그림 1(c) 형상에서의 두 개의 분기점을 보여준다.

준대칭 경로는 중간에 비대칭적 추적경로를 허용하는 대칭 경로를 의미하며, 대칭 경로도 준대칭 경로로 해석할 수 있다. 그림 7은 괄호로 묶여진 준대칭 경로의 예를 보여주는데, 잡영블랍을 포함하는 돌출 가지들이 준대칭 경로로 표현됨을 알 수 있다.

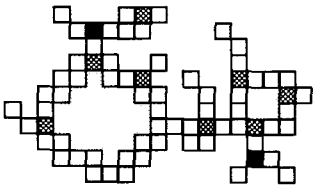
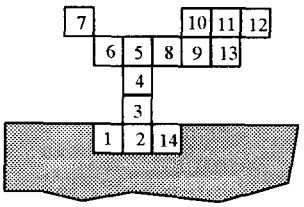


그림 6. 분기점(■)과 뿌리점의 예



1,**[2,3,4**,5,6,7,6,5,8,9,10,11,12,11,13,9,8,5,**4,3,2]**,14
1,2,3,4,**[5,6,7,6,5]**,8,9,10,**[11,12,11]**,13,9,8,5,4,3,2,14
1,2,3,4,5,6,7,6,**[5,8,9**,10,11,12,11,13,**9,8,5]**,4,3,2,14

그림 7. 준대칭 경로의 예

4-8-DBF에 의한 준대칭 경로의 시작점을 뿌리점(root point)이라고 정의한다. 뿌리점은 그림 6에서 보듯이 돌출 가지를 인식하는데 유용하다. 뿌리점에 연결된 돌출 가지를 본 논문에서는 뿌리 가지(root branch)라고 부른다. 뿌리점은 뿌리 가지에 포함시키지 않는다. 분기점은 뿌리점의 한 종류로서 하나 이상의 뿌리 가지를 가진다. 하나의 분기점에 대한 뿌리 가지의 개수는 그 분기점의 0-1-변환 정도 보다 하나 적다.

뿌리 가지는 다음과 같이 세 가지 형태로 분류할 수 있다.

(형태1) **평 가지**(normal branch) : 대칭 경로를 가지는 한 픽셀 두께의 뿌리 가지

(형태2) **모 가지**(mother branch) : 가지의 끝점이 분기점인 한 픽셀 두께의 뿌리 가지

(형태3) **블랍 가지**(blob branch) : 가지의 끝에 잡영블랍을 가지는 뿌리 가지

평 가지는 분기점이나 잡영블랍을 포함하지 않는 뿌리 가지를 나타내는 것으로서, 그림 7에서는 두 개의 평 가지 (6,7)과 (12)를 볼 수 있다. 모 가지는 그림 7에서 (3,4,5) 밖에 없다. 블랍 가지는 준대칭 경로로 표현되며, 3.1절에서 정의된 잡영블랍을 가지 끝에 가지고 있는 것으로서, 그림 7에서는 (8,9,10,11,13)만이 블랍 가지가 된다. 그림 8은 여러 형태의 뿌리 가지에 대한 다른 예를 보여준다.

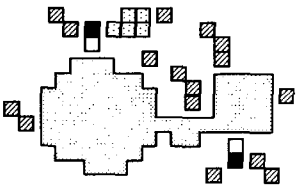


그림 8. 평 가지(빗금친 것), 모 가지(검은 분기점을 포함한 것), 블랍 가지(점박이 가지)의 예

잡영가지를 더욱 유연하게 정의하기 위해서, 본 논문에서는 두 개의 매개변수 즉, 잡영가지 크기에 대한 매개변수 N_S와 잡영블랍 크기에 대한 매개변수 N_B를 사용한다. 여기에서, 크기는 각 대상을 구성하는 픽셀 수를 의미한다. 그리고, 편의상 평 가지나 모 가지의 끝점을 $N_{B=1}$인 블랍으로 간주하여, 블랍 크기는 1, 4, 또는 6으로 표현한다.

이와 같이 정의된 여러 가지 용어를 이용하여, 본 논문에서의 잡영가지는 다음과 같이 정의된다.

(잡영가지의 정의)
잡영가지는 그 크기가 주어진 잡영가지 크기 N_S보다 같거나 작고, 주어진 블랍 크기 N_B 보다 큰 크기의 블랍을 포함하지 않는 뿌리 가지이다

4. 잡영가지 제거 알고리즘

잡영가지들은 4-8-DBF 기법을 이용하여 효율적으로 제거할 수 있다. 4-8-DBF에 의한 추적경로는 하나의 링크드-리스트로 구현되며, 잡영가지가 검출될 때마다 즉시 제거한다. 추적 및 제거 과정에서 분기점에 연결된 다른 뿌리 가지들로부터 모 가지를 분리해내기 위해, 그 점의 0-1-변환 정도 만큼 분기점을 리스트에 추가하는 방법을 사용한다. 또한, 뿌리점이 제거되면 대칭 경로의 추적을 제대로 할 수 없으므로, 추적경로를 유지하기 위해 뿌리점을 한번 더 리스트에 추가하는 방법도 사용한다. 잡영가지 크기 변수 값과 블랍 크기 매개변수 값을 각각 5와 4로 설정했을 때, 그림 7의 형상에 대한 잡영가지 제거 예에 대한 주요 추적 단계는 다음과 같이 나열할 수 있다.

(S1) ·, 1, 2, 3, 4, **5, 5, 5**

(S2) ·, 1, 2, 3, 4, 5, 5, 5, 6, **[7]**, 6

(S3) ·, 1, 2, 3, 4, 5, 5, **(5, 6, [7], 6, 5)**, 8

(S4) ·, 1, 2, 3, 4, 5, 5, 8, 9, 10, **(11, [12], 11)**, 13

(S5) ·, 1, 2, 3, 4, 5, 5, 8, 9, 10, **11**, 13

(S6) ·, 1, 2, 3, 4, 5, 5, 8, **[9, 10, 11, 13, 9]**

(S7) ·, 1, 2, 3, 4, **[5]**, **(5, 8, [9, 10, 11, 13, 9], 8, 5)**, 4

(S8) ·, 1, **(2, 3, 4, [5], 4, 3, 2)**, 14

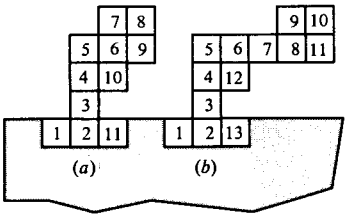
(S9) ·, 1, **2**, 14

첫번째 단계 (S1)은 0-1-변환 정도가 세 개인 분기점을 만나는 경우를 나타낸다. (S2)는 크기가 1인 블랍을 탐지한 경우이다. (S3)과 (S4)는 평 가지를 검출하여 제거하는 경우이다. (S5)에서는 뿌리점 11이 리스트에서 제거되므로, 다시 한번 추가하는 것을 보여 준다. (S6)은 크기가 4인 블랍을 검출하는 것을 보여준다. (S7)에서는 하나의 블랍 가지가 제거되고 크기가 1인 다른 블랍이 발견된 경우이다. (S8)에서는 한 개의 모 가지가 발견되어 제거되는 것을 보여주고, (S9)에서는 뿌리점이 다시 추가되는 것을 보여준다.

제거된 준대칭 경로의 시작점이 뿌리점인지 분기점인지를 살펴 보아야 한다. 시작점의 바로 앞 추적 픽셀이 시작점과 같으면 시작점은 분기점이고, 다르면 뿌리점이 된다. 분기점이 하나씩 제거되어 한 개가 남아 있는 경우에는, (S7)과 (S8)에서 보듯이 하나의 모 가지를 생성하게 된다.

잡영블랍이 제거되기 위해서는 다른 잡영블랍과 적어도 한 픽셀 이상으로 떨어져 있어야 한다. 그림 9(a)에서의 돌출 잡영은 잡영블랍 조건들이 만족되는 블랍을 가진 준대칭 경로를 형성하지 않기 때문에 제거될 수 없다. 반면에 그림 9(b)의 돌출 잡영은 두 개의 블랍 가지들로 구성된 것이지만 제거할 수 있다.

4-8-DBF에 대한 시작점이 돌출 가지의 픽셀일 때, 그 가지를 잡영가지로 분석하여 제거하는 것은 매우 복잡하고 어렵다. 이에, 본 논문에서는 주어진



(a) 1,2,3,4,5,[6,7,8,9,6],10,4,3,2,11

(b) 1,(2,3,[4,5,(6,7,[8,9,10,11,8],7,6),12,4],3,2),13

그림 9. 잡영블랍을 가지는 (a) 제거 불가능한 돌출 가지와 (b) 제거 가능한 돌출 가지

형상의 중심 픽셀이거나 제거할 수 없는 돌출 가지의 픽셀을 새로운 시작점으로 결정하는 방법을 사용한다. 이 새로운 시작점을 안전 시작점이라고 부른다. 형상의 중심은 내부 집합(interior set)과 내부 집합을 이루는 픽셀과 4-연결 또는 8-연결된 윤곽선 픽셀들로 이루어진 집합의 합집합으로 정의한다. 그림 10(a)는 그림 3에 있는 형상의 중심을 보여주는 것으로서, 빈 원과 빈 사각형으로 표시되어 있는 시작점이 사용자에 의해 주어지거나 라스터(raster) 스캔에서 자동 선택된 것이라면, 검은 원과 검은 사각형으로 표시된 픽셀이 안전 시작점으로 결정된다.

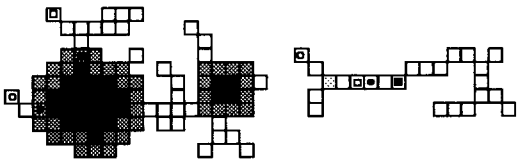


그림 10. 안전한 시작점 찾기 예

제거할 수 없는 돌출 가지는 다음 조건들 중 하나를 만족하는 경우로 정의할 수 있다.

(조건1_{IB}) 주어진 잡영가지 크기인 N_S의 횟수 만큼 윤곽선을 추적하는 동안 분기점이나 뿌리 가지가 없을 경우 (그림 10(b)에서 속이 빈 사각형과 검은 작은 사각형 참조)

(조건2_{IB}) 주어진 잡영가지 크기인 N_S의 횟수 만큼 윤곽선을 추적하는 동안 분기점이 발견되지만, 그 분기점으로부터 윤곽선을 추적하는 동안 (조건1_{IB})을 만족시키는 경우 (그림 10(b)에서 속이 빈 원과 검은 작은 빈 원 참조)

제안하는 잡영가지 제거 알고리즘은 다음과 같다.

단계 1. 잡영가지 크기 매개변수 N_S와 블랍 크기 매개변수 N_B의 값을 설정한다. 주어지는 4-8-DBF의 시작점으로부터 안전 시작점 s를 결정한다. 윤곽선 추적의 현재 픽셀을 나타내는 변수 c를 s로 설정한다(c = s).

단계 2. c로부터 새로운 분기점, 뿌리 가지, 또는 안전 시작점을 만날 때까지 윤곽선을 추적한다. c를 가장 최근에 추적한 픽셀로 설정한다.

단계 3. 다음의 경우 중에 하나를 실행한다.

(경우 1) c가 새로운 분기점이면, c를 [(0-1-변환

정도)-1] 횟수만큼 추적 경로 리스트에 추가한다. 그리고 단계 2로 간다.

(경우 2) 뿌리 가지를 발견했다면, 가지의 크기를 N_S와 비교하여 제거 여부를 결정한다. 만약 뿌리 가지가 제거할 수 없는 돌출 가지라면, 단계 2로 간다. 그렇지 않으면, 뿌리 가지를 제거한다. 만약 c가 뿌리점이면, c를 추적 경로 리스트에 한번 추가한다. 단계 2로 간다.

(경우 3) c가 안전한 시작점이면, 윤곽선을 한번 순회한 것이므로 알고리즘을 종료한다.

5. 실험 결과 및 토의

먼저 그림 11(a)에서와 같은 인위적인 물체 형상에 대하여 기존의 방법 및 제안한 방법의 문제점 및 효용성에 대하여 알아 본다. 그림 11(b),(c)는 [7]에서의 윈도우 내의 물체 픽셀 개수에 대한 임계값을 이용하여 제거한 결과로서, 물체 픽셀 개수가 각각 4 또는 5 미만인 경우에 해당 픽셀을 제거한 결과를 보여 준다. 그림 11(d)는 윤곽선 시퀀스를 표현하는데 중복 사용된 픽셀을 제거하는 방법[8]에 의한 결과로서, 본래 4-연결 경로를 사용하고 있으므로 빗금친 픽셀들은 없는 것으로 간주하였으며 돌출 가지의 끝점은 중복 사용된 것으로 처리하였다. 그림 11(e)는 [8]에서 제안한 개념을 보다 현실적으로 반영하기 위하여 8-연결 경로를 사용하여 처리한 결과이다. 그림 11(f)는 세선화 결과에서의 잔가지 제거 방법[6]에서의 첫번 째 단계인 8가지 구조화 요소에 의한 가지 끝점 제거과정을 수행한 결과이다. 그림 11(g)는 이러한 가지 끝점 제거과정을 네 번 반복적으로 적용한 결과로서, 앞의 방법들과 다르게 연결정보를 잘 보존하고 있으며 돌출 잡영가지도 잘 제거하고 있음을 알 수 있다. 그러나, 반복적으로 전체 픽셀을 스캔하면서 끝점을 제거하여야 하므로 연산시간이 많아지며, 또한 잡영블랍도 제거하지 못함을 알 수 있다. 반면에, 대칭 경로 검출에 의한 돌출 잡영가지 제거 방법[9]에서는 윤곽선 추적에 의해 신속하게 그림 11(g)와 동일한 결과를 제시한다. 그림 11(h)는 본 논문에서 제안한 방법에 의한 결과로서, 잡영블랍까지도 신속하게 제거할 수 있음을 보이고 있다.

그림 12는 신발 제작용 피혁 부품을 나타낸 것으로서, 형상 경계 부분에 많은 돌출 잡영가지가 있음을 보이고 있다. 그림 13(a)는 방법[9] 또는 제안한 방법에서 잡영가지의 크기를 14로 하고 잡영블랍의

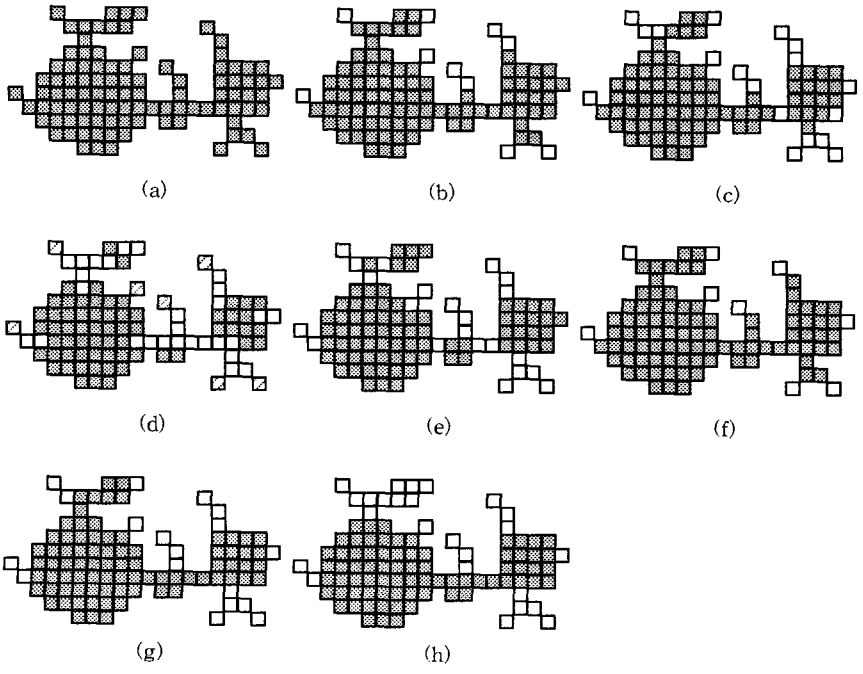


그림 11. 여러 가지 돌출 잡영가지 제거 방법에 의한 처리 결과

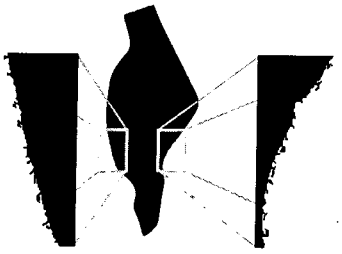
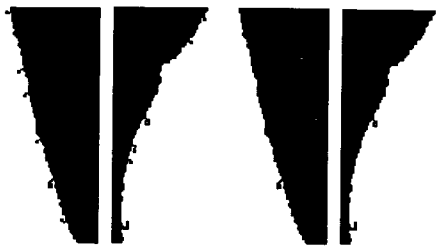


그림 12. 신발 피혁 부품에서의 돌출 잡영가지의 예



(a) (b)

그림 13. 신발 피혁 부품에서의 돌출 잡영가지 제거; (a) N_S = 14, N_B = 1, (b) N_S = 14, N_B = 6

크기는 1로 설정함으로써, 한 픽셀 두께의 돌출 잡영가지만을 제거한 결과이다. 반면에, 그림 13(b)는 잡영블랍의 크기를 6으로 설정함으로써 대부분의 잡영가지가 제거된 것을 보이고 있다. 그러나, 잡영블랍 내에 홀이 있거나 그 크기가 6보다 큰 경우에는 여전히 남아 있음을 알 수 있다. 이러한 잔여 가지를 제거하기 위해서는 후처리 작업이 필요하다. 따라서, 사용자가 원하는 수준의 처리 결과를 얻기 위한 목적에서 보면, 본 논문에서 제안한 방법도 일종의 돌출 잡영가지 제거의 전처리 작업에 활용할 수밖에 없는 정도임을 알 수 있다.

그림 14는 신발 제작용 피혁 부품 20가지를 나타낸 것으로서, 이것들에 대해 여러 가지 방법으로 돌출 잡영가지를 제거하는 실험을 하였다. 이 부품들로부터 제거해야 할 총 픽셀 개수는 그림 13(b)에서와 같이 본 논문에서 제안한 방법으로 제거한 픽셀 수와

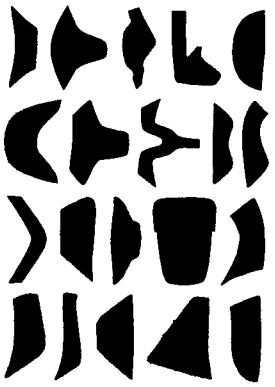


그림 14. 여러 가지 신발 피혁 부품의 예

아직 남아 있는 제거가 필요한 픽셀 수를 더한 것으로 산정하였다. 각 방법에 대해 잡영가지 제거에 제일 적합한 조건 또는 매개변수 값을 설정하여 실험한 결과를 표 2에 나타내었다. 여기에서, 윈도우 내의 물체 픽셀 개수의 이용 방법[7]과 윤곽선 시퀀스에서의 중복 픽셀 제거 방법[8]에서는 잘못 제거한 픽셀들이 있음을 알 수 있다. 이들에 대한 제거 비율은 이러한 픽셀들은 제외하고 계산하였다. 본 논문에서 제안한 방법에 의한 제거 비율이 매우 높게 나타남을 알 수 있다. 한편, 연산 시간 측면에서 보면, [8]에서의 방법을 제외하고는 거의 비슷한 수준을 보이고 있다. 그림 15는 각 방법에 대하여 매개변수 값을 변경하였을 때의 픽셀 제거 비율을 보이고 있다. 윈도우[7] 그래프 내에서의 숫자는 윈도우 내의 물체 픽셀 개수에 대한 임계값을 나타내며, 윈도우[7] 및 시퀀스[8] 그래프들에서 위 부분에 진하게 나타낸 것은 잘못 제거한 비율을 표현하기 위한 것이다. 대칭경로[9] 그래프에서의 숫자는 제거할 잡영가지의 길이 정의에 사용한 매개변수값이며, 제안한 방법에 대한 그래프에서의 슬래쉬(slash) 위아래의 숫자는 각각 잡영가지의 길이와 잡영블랍의 크기 정의에 사용한 매개변수값을 나타낸다. 그림 15의 그래프를 전체적으로 보면, 본 논문에서 제안한 방법이 잘못 제거한 픽

표 2. 여러 가지 방법에 의한 잡영가지 제거 결과 (제거할 총 픽셀 개수 = 3076)

제거 방법	윈도우[7]	시퀀스[8]	모폴로지[6]	대칭경로[9]	제안방법
제거한 픽셀 개수	2947	1592	2432	2432	3021
잘못 제거한 개수	655	312	0	0	0
제거 비율	74.42%	41.61%	79.06%	79.06%	98.21%
연산 시간	0.199	0.274	1.489	0.191	0.206

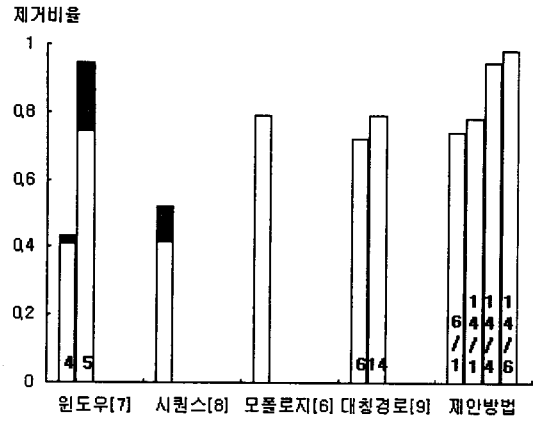


그림 15. 매개변수 변경에 따른 여러 가지 방법의 잡영가지 제거 비율

셀도 없으면서 잡영블랍의 제거 효과에 의해 제거비율도 상대적으로 높음을 알 수 있다.

그림 16은 컬러 인쇄된 지도 영상으로부터 추출된 해안선을 나타낸 것으로서, 잡영가지 제거의 필요성에 대한 다른 예를 보여 주고 있다. 영상에서 해안선의 칼라가 바다 영역에 대한 해프톤 칼라와 유사하기 때문에 많은 돌출 잡영가지를 있음을 알 수 있다[9]. 그림 17은 각기 다른 잡영블랍 크기 매개변수 값에 의한 잡영가지 제거 결과를 보여주는데, 크기를 6으로 했을 때 우리가 원하는 것을 얻을 수 있음을 알 수 있다.

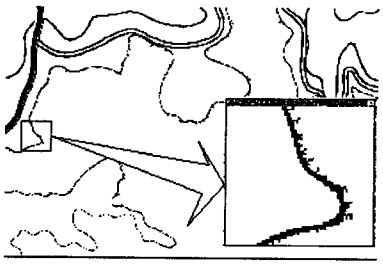


그림 16. 칼라의 지도로부터 추출된 돌출 잡영가지가 많은 해안선

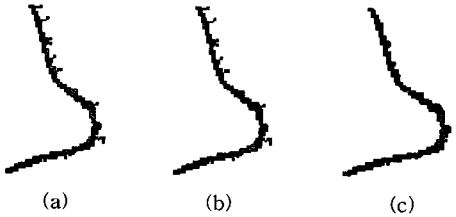


그림 17. 잡영 블랍의 크기를 각각 1, 4, 6로 했을 때의 잡영가지 제거 결과

제안한 잡영가지 제거 방법은 세선화 분야에서 전처리 작업에 유용하게 활용할 수 있다. 단순화된 형상의 세선화는 그만큼 더 잡영가지를 적게 가진다. 그림 18(a)-(c)는 각각 그림 17(a)-(c)에 대한 세선화 [10] 결과를 보여주는데, 그림 18(c)가 훨씬 좋은 골격을 가짐을 알 수 있다. 또한, 그림 18(a),(b)와 같은 세선화 결과에 대하여 제안한 방법을 이용하여 잔가지를 제거할 수도 있다. 그림 18(d)는 그림 18(a)의 골격 형상에 대하여 잔가지를 제거한 것을 보여준다.

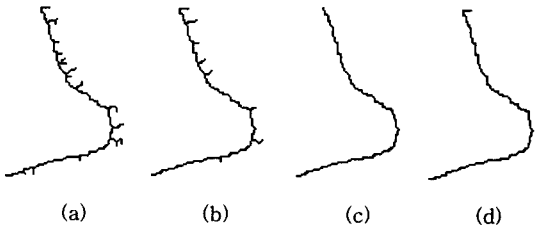


그림 18. 잡영가지 제거 예(그림 17)에 대한 세선화 결과 및 잔가지 제거 결과

제안한 방법은 연쇄적으로 짧은 가지들이 연결되어 있을 경우에, 전체적인 가지 길이가 긴 것임에도 불구하고 각 짧은 가지들을 연쇄적으로 제거함으로써 가지 전체를 제거하게 되는 단점이 있다. 그림 19는 잡영가지 크기 매개변수 값을 5로 주었을 때 연쇄적으로 연결된 긴 가지가 모두 제거될 수 있는 경우를 보여준다. 따라서, 향후에는 연쇄적 잡영가지 형태의 돌출 가지에 대한 전체 픽셀 수를 계산하여 제거 대상에서 배제하는 방법을 개발할 필요가 있다.

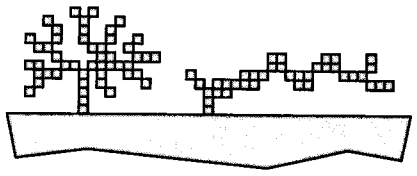


그림 19. 연쇄적 잡영가지의 예

6. 결 론

잡영블랍 및 잡영가지를 명확히 정의한 후에 4-8-방향 윤곽선 추적 알고리즘을 이용하여 제거하는 방법을 제안하였다. 이 방법으로 한 픽셀 두께의 잡영가지 뿐만 아니라, 잡영블랍을 포함하는 돌출 가지도 제거할 수 있었다. 매개변수로는 잡영가지 크기와 잡

영블랍 크기만을 사용하여 다양한 잡영가지를 편리하게 정의할 수 있었다. 제안한 방법은 형상의 윤곽선을 단순화하거나 세선화된 형상의 잔가지를 제거하는 데에도 응용할 수 있었다.

그러나, 제안한 방법은 짧은 잡영가지들이 연쇄적으로 연결되어 있는 전체적으로 큰 형태의 돌출 가지도 제거해 버리는 문제점을 아직 가지고 있다. 근본적으로, 사용자가 제거하기를 원하는 모든 돌출 가지를 수용할 수 있는 정확한 잡영가지의 정의는 매우 어려울 것으로 보인다. 따라서, 본 논문에서 제안한 방법은 사용자의 돌출 가지 제거에 대한 요구를 단계적으로 충족시켜 나가기 위한 일종의 필터링 도구로 활용하는 것이 적합할 것으로 판단한다.

참 고 문 헌

[1] R. jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, McGraw-Hill, Singapore, 1995.

[2] S. O. Kim, S. Y. Kim, J. M. Kim, and M. H. Kim, Usefulness of Boundary Sequences in Computing Shape Features for Arbitrary Shaped Regions, *Proc. Int. Conf. on Pattern Recognition*, Vol.3, pp. 355-358, Quebec, Canada, Aug. 2002.

[3] Y. S. Chen and W. H. Hsu, Parallel Thinning Algorithm for Binary Digital patterns, in *Handbook of Pattern Recognition & Computer Vision*, World Scientific, Singapore, 1993.

[4] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, Brooks/Cole Publishing Co., 1999.

[5] G. S. D. Baja, Well-shaped, stable and reversible skeletons from the (3,4)-distance transform, *J.Visual Commun. Image Repres*, Vol. 5, No. 1, pp. 107-115, 1994.

[6] R. C. Gonzalez and R. E. Wood, *Digital Image Processing*, Addison Wesley, 1993.

[7] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.

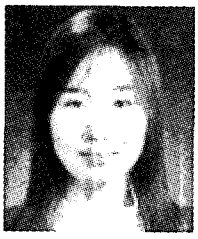
[8] J. Sklansky, Recognition of convex blobs, *Pattern Recognition*, Vol. 2, pp. 3-10, 1970.

[9] 권태균, 김종민, 김성영, 김민환, 2×2 마스크를 이용한 윤곽선 추적 알고리즘의 응용에 관한 연구, 한국정보처리학회 97추계 학술발표논문집, 제4권2호, pp.1163-1168, 1997

[10] T. Y. Zhang and C. Y. Suen, A fast parallel algorithm for thinning digital patterns, *Communications of ACM*, Vol. 27, No. 3, pp. 236-239, 1984.

김 성 옥

1981년 부산대학교 수학과 학사
1998년 부산대학교 대학원 컴퓨터공학과 석사
2003년 부산대학교 대학원 컴퓨터공학과 박사
1982년~1989년 한국기계연구원 연구원
1989년~현재 한국신발피혁연구소 자동화연구부 연구부장
관심분야 : 화상처리, 컴퓨터비전, 패턴인식



임 은 경

1999년 신라대학교 전자계산학과 학사
2001년 신라대학교 대학원 전자계산학과 석사
2002년~현재 부산대학교 대학원 컴퓨터공학과 박사과정

관심분야 : 화상처리, 컴퓨터비전, 신경망알고리즘

김 민 환

1980년 서울대학교 전기공학과 학사
1983년 서울대학교 대학원 컴퓨터공학과 석사
1988년 서울대학교 대학원 컴퓨터공학과 박사
1991년~1992년 미국 워싱톤 대학 객원연구원
1986년~현재 부산대학교 컴퓨터공학과 교수
관심분야 : 화상처리 및 이해, 칼라공학

교 신 저 자

김 민 환 609-735 부산시 금정구 장전동 산30