■ 2007년도 학생논문 경진대회 수상작

이차원 영상의 라인 드로잉
(Line Drawings from 2D Images)

손 민 정 † 이 승 용 ††
(Minjung Son) (Seungyong Lee)

요 약 라인 드로잉은 적은 표현으로 물체에 대한 많은 정보를 줄 수 있다는 점 때문에 비사실적 렌더링 분야에서 중요시되고 있다. 하지만 라인 드로잉에 대한 연구는 이차원 영상에 비해 물체에 대한 정보가 충분한 삼차원 모델을 대상으로 주로 이루어졌다. 본 논문에서는 이차원 영상을 라인 드로잉 형태로 표현하는 효과적인 방법을 제시한다. 이를 위한 알고리즘은 크게 필터링, 선 연결, 스타일화 세 단계로 나뉜다. 필터링 단계에서는 영상의 어느 부분에 선이 그려질지를 우도 함수를 이용하여 예상한다. 선 연결 단계에서 필터링 결과를 클러스터링 및 그래프 검색을 이용하여 연결, 라인 스트로크들을 찾아낸다. 마지막 스타일화 단계에서는 찾아낸 라인 스트로크들을 곡선 근사, 텍스쳐 매핑 등을 이용하여 여러 비사실적 렌더링 형태로 표현한다. 이러한 방법을 이용하여 실제 이차원 영상에서 라인 스트로크를 얻고, 디테일 제어를 적용하여 여러 가지 원하는 스타일의 라인 드로잉을 만들 수 있다.

키워드 : 컴퓨터 그래픽스, 비사실적 렌더링, 라인 드로잉, 영상 필터링, 특징선 검출, 스타일화

Abstract Line drawing is a widely used style in non-photorealistic rendering because it generates expressive descriptions of object shapes with a set of strokes. Although various techniques for line drawing of 3D objects have been developed, line drawing of 2D images has attracted little attention despite interesting applications, such as image stylization. This paper presents a robust and effective technique for generating line drawings from 2D images. The algorithm consists of three parts; filtering, linking, and stylization. In the filtering process, it constructs a likelihood function that estimates possible positions of lines in an image. In the linking process, line strokes are extracted from the likelihood function using clustering and graph search algorithms. In the stylization process, it generates various kinds of line drawings by applying curve fitting and texture mapping to the extracted line strokes. Experimental results demonstrate that the proposed technique can be applied to the various kinds of line drawings from 2D images with detail control.

Key words : computer graphics, non-photorealistic rendering, line drawing, image filtering, contour extraction, stylization

1. 서 론

컴퓨터 그래픽스 분야가 생겨난 이후 사람들은 보다 사실적인 영상을 만드는 것에 큰 관심을 보였고, 이를 위하여 실제 환경에서 시각에 영향을 주는 색상, 빛 등을 표현하고 처리하기 위한 연구가 많이 이루어졌다. 그 결과 현재 컴퓨터 그래픽으로 만들어진 영상은 영화나 사진 등 어디에서나 쉽게 찾아볼 수 있을 정도로 많이 사용되게 되었다. 또한 그 결과물의 정밀함도 점차 실제와 구분하기 힘들 정도로 뛰어나게 발전하였다. 이러한 사실적인 렌더링에 대한 연구가 어느 정도의 수준에 이르게 되자, 이제는 사람이 손으로 직접 그린 듯한 영상이 연구 대상으로 떠오르기 시작하였다. 다양한 재료와 기법으로 예술가들이 그린 듯한 회화적인 영상을 만드는 것이 목표인 이러한 연구 분야를 비사실적 렌더링

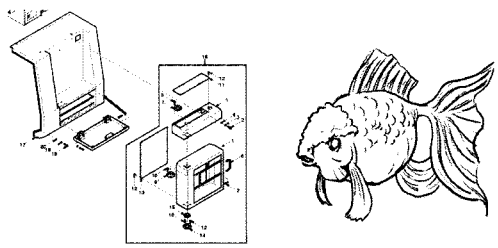
† 학생회원 : 포항공과대학교 컴퓨터공학과
sionson@postech.ac.kr
†† 종신회원 : 포항공과대학교 컴퓨터공학과 교수
leesy@postech.ac.kr
논문접수 : 2007년 5월 29일
심사완료 : 2007년 11월 1일

정보과학회논문지 : 시스템 및 이론 제34권 제12호(2007.12)

(non-photorealistic rendering)이라고 한다.

비사실적 렌더링은 화법에 따라 다양한 방향으로 연구되었으나 그 중 공통되는 부분이 있다. 재료의 특성을 살려 물체의 내부를 칠하고, 마찬가지로 특징이 되는 선을 재료의 특징이 잘 나타나도록 그려준다는 점이다. 화법에 따라서는 내부 칠하기(interior shading)나 라인 드로잉(line drawing) 중 하나만을 하기도 한다. 예를 들어 유화의 경우 내부 칠하기 만으로 이루어지고, 수묵화의 백묘법은 라인 드로잉 만으로 이루어진다.

이 중 라인 드로잉은 적은 표현으로 많은 양의 정보를 줄 수 있다는 점 때문에 물건의 설명서 그림에서부터 여러 가지 미술 화법에 이르기까지 다양하게 이용되어 왔다(그림 1). 특히 선을 그릴 때 사용하는 재료의 질감 및 선의 굵기와 진하기, 그리는 사람의 습관 등에 따라 개성있는 결과 영상을 얻을 수 있고, 내부 채우기만으로는 부족한 정보를 보는 사람에게 쉽게 전달해주기 때문에 보다 사람이 그린 것에 가까운 비사실적 렌더링을 위하여 매우 중요하다. 이러한 라인 드로잉은 실제 영상에는 드러나지 않는 것을 표현하는 추상화된 과정으로 볼 수 있다.



(a) 설명서 (b) 수묵화

그림 1 라인 드로잉의 예

1.1 연구 동기

라인 드로잉은 특정한 재료를 사용하여 선을 그림으로써 그 대상을 표현하는 방법이다. 재료의 특성을 살린 단순한 선 몇 개로 실제 물체에 대한 많은 정보를 함축적으로 줄 수 있기 때문에 비사실적 렌더링 분야에서 굉장히 중요한 의미를 갖는다.

라인 드로잉을 위해서는 대상의 특징을 잘 나타낼 수 있는 부분을 따라 길고 부드럽게 연결된 라인 스트로크(line stroke)가 필요하다. 일단 라인 스트로크만 있다면 해당 스트로크를 따라 굵기를 가진 선을 그리거나 원하는 재료의 선 텍스쳐(texture)를 매핑하여 그 질감이 나타나는 선을 만드는 등의 방법으로 다양한 종류의 라인 드로잉을 쉽게 할 수 있다.

기존의 라인 드로잉은 물체에 대한 정보가 충분한 삼차원 모델을 대상으로 이루어졌다. 삼차원 모델의 경우 물체를 이루는 각 점들의 연결 관계 및 성질이 잘 정의되어 있기 때문에 이를 이용하여 특징선(feature edge)을 정의할 수 있고, 이 특징선들을 연결하여 비교적 쉽게 라인 스트로크를 얻을 수 있다. 원하는 삼차원 모델을 만드는 것이 어렵다는 단점이 있지만, 삼차원 모델만 있으면 원하는 방향에서 원하는 자세의 대상에 대한 라인 드로잉을 만들 수 있다. 따라서 이러한 삼차원 모델에 대하여 특징선을 찾아 연결하고 재료의 특성을 살려 실제 사람이 그린 것 처럼 라인 드로잉 하는 방법에 대한 연구가 다양하게 진행되었다.

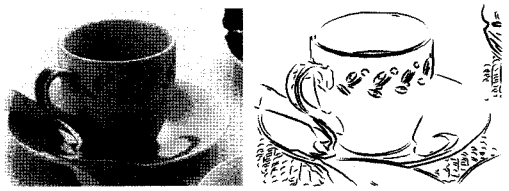
반면 이차원 영상은 규칙적인 좌표 상의 픽셀(pixel)들이 각 점에서의 색상을 저장하고 있을 뿐 영상 내의 물체에 대한 정보가 부족하다. 따라서 이차원 영상에서의 선 관련 연구는 대부분 픽셀 단위로 특징선에 해당되는 부분을 찾아내는 정도까지만 이루어졌다. 이는 찾아낸 픽셀 단위 정보만으로는 라인 드로잉을 위해 필요한, 사람이 선을 그릴 때 수행하는 것과 같은 추상적인 과정을 통해 얻을 라인 스트로크를 쉽게 만들 수가 없기 때문이다. 하지만 이차원 영상은 사진기 등을 이용하여 누구나 쉽게 만들 수 있다는 장점이 있고, 이러한 이차원 영상을 입력으로 하여 다양한 스타일로 라인 드로잉 하는 것은 영상의 스타일화 측면에서도 의미있는 일이다.

따라서 지금까지 많이 연구되지 않았던 이차원 영상에 대하여, 자동으로 라인 스트로크를 구성하고 원하는 스타일로 라인 드로잉 하는 연구를 수행하게 되었다.

1.2 문제 정의

본 연구에서는 한 장의 이차원 영상을 입력으로 받아 사용자가 원하는 스타일의 라인 드로잉 영상를 자동으로 생성하는 것을 목적으로 한다. 예를 들어 그림 2(a)가 입력 영상으로 들어왔을 경우, 그림 2(b)와 같이 사용자가 원하는 스타일의 라인 드로잉 영상이 결과로 출력된다.

이를 위한 방법은 사람이 선을 그리는 과정과 최대한 유사한 알고리즘으로 만들어진다. 하나의 픽셀이 아닌 주위의 여러 픽셀들의 정보를 함께 이용하여 어디에 선이 그려져야 할 지를 판단하고, 선이 그려져야할 부분에 가능한한 길고 부드럽게 연결된 라인 스트로크를 생성한다. 그리고 라인 스트로크로 연결된 점들을 따라 원하는 재료의 선 텍스쳐를 매핑하여 사람이 그린 듯한 라인 드로잉을 완성한다.



(a) 입력 영상 (b) 출력 영상

그림 2 문제 정의

1.3 연구 목표

본 연구는 하나의 이차원 영상으로부터 다양한 스타일의, 사람이 그린 듯한 라인 드로잉 결과 영상을 만드는 것이 목표이다. 이는 기존의 삼차원 모델을 대상으로 한 라인 드로잉에 비해 입력이 되는 영상을 누구나 쉽게 만들 수 있다는 장점이 있다. 더하여 기존의 이차원 영상에서의 선 관련 연구와 달리 다양한 선 텍스쳐를 이용한 스타일화가 가능하다.

이러한 이차원 영상을 대상으로 한 라인 드로잉을 위하여, 본 연구에서는 크게 다음의 두 내용에 초점을 두었다.

- 라인 드로잉을 위해 특화된 필터링 방법을 정의
  이차원 영상에서 특징선을 찾는 것에 대해서는 많은 방법이 제시되어 왔다. 하지만 이들은 그 결과를 선으로 연결하는 것에 대한 고려 없이 특징선에 해당하는 부분을 픽셀 단위로 찾은 것이기 때문에, 픽셀 각각은 정확한 특징선 위치이더라도 사람이 그린 것처럼 부드러운 선으로 연결되기는 매우 어렵다. 따라서 본 연구에서는 이후 부드러운 선으로 연결되어야 한다는 점을 고려한, 라인 드로잉을 위해 특화된 필터링 방법을 정의하였다.
- 이차원 영상에서 라인 스트로크를 구함
  다양한 스타일화를 위해서는 다양한 재료의 재질을 텍스쳐로 이용하는 방법이 가장 쉽고 효과적이다. 이러한 텍스쳐 매핑이 가능하도록 하기 위하여 이차원 영상 상의 점들로부터 라인 스트로크를 구하는 알고리즘을 정의하였다.

이를 위하여 본 연구는 크게 다음의 세 단계로 나뉘어 구성되어 있다.

- 필터링(Filtering)
  입력된 영상의 그래디언트 영상을 구하고 이를 이용하여 영상의 어느 부분에 선이 그려져야 할지를 판단하는 단계이다.
- 선 연결(Linking)
  선을 그려야 한다고 판단한 부분들을 가능한한 길고 부드럽게 연결하여 라인 스트로크로 만드는 단계이다.
- 스타일화(Stylization)
  선 연결 결과 얻어진 라인 스트로크를 여러 가지 원하는 스타일의 라인 드로잉으로 만들어주는 단계이다.

1.4 논문의 구조

2장에서는 삼차원 모델을 이용한 라인 드로잉이나 이차원 영상을 대상으로 한 선 연구 등 본 연구와 관련된 이전 연구들에 대하여 살펴본다. 3장에서는 이차원 영상의 라인 드로잉을 위한 알고리즘을 개괄적으로 살펴보고, 전체를 크게 필터링, 선 연결, 스타일화 세 부분으로 나눈다. 그리고 각 부분에 대하여 4장, 5장, 6장에서 자세히 설명한다. 7장에서는 앞에서 설명한 알고리즘에 따라 라인 드로잉 한 결과를 제시한다. 마지막으로 8장에서는 본 연구의 결과를 정리하고 향후 연구 과제를 살펴본다.

2. 관련 연구

이 장에서는 이차원 영상의 라인 드로잉과 관련된 이전 연구들을 살펴본다. 일단 삼차원 모델에 대한 라인 드로잉 연구를 살펴보고, 이차원 영상에서 특징선을 검출하는 연구를 살펴본다. 그리고 이차원 영상이나 비디오에 대한 스타일화 연구가 어떠한 방향으로 진행되었는지 알아보도록 한다.

2.1 삼차원 모델의 라인 드로잉

삼차원 모델은 물체에 대한 정보가 충분하다는 점 때문에 특징선이 되는 부분에서 라인 스트로크를 뽑아내어 다양한 스타일의 라인 드로잉을 하는 연구가 비교적 많이 이루어졌다.

삼차원 모델의 라인 드로잉을 위해서는 기본적으로 라인 스트로크 상태의 윤곽선(silhouette edge)을 찾는 부분이 선행되어야 하고, 이에 대한 연구는 지금까지 다양하게 이루어져 왔다[1]. 특히 Markosian 등은 확률 기반으로 빠르게 윤곽선을 검출하는 방법을 제시하였고, 그로 인해 실시간으로 윤곽선을 찾는 것이 가능해 졌다[2]. 윤곽선을 실시간으로 찾을 수 있을 때의 장점은 사용자가 시점이나 모델의 위치 등을 바꾸면서 그 결과를 바로 확인할 수 있고, 또한 애니메이션으로 확장이 가능하다는 점이다.

Northrup과 Markosian은 이러한 방법을 기반으로 하여 3차원 메쉬의 윤곽선을 스타일화된 스트로크들로 렌더링하는 방법을 제시하였다[3]. 이는 찾은 윤곽선에 가시성 검사를 거치고 지그재그한 부분을 푼 상태로 윤곽선을 연결하여 긴 경로를 얻어내는 방법으로 이루어졌다.

또한 Kalins 등은 삼차원 모델 상에 사용자가 직접 스트로크를 적용하도록 하는 상호 작용적인(interactive) 방법으로 보다 좋은 결과를 얻어내었다[4].

한편 Rössl과 Kobbelt는 삼차원 모델의 정보를 이차원 상으로 투영, 모델에 대한 정보를 충분히 이용하면서도 투영 과정에서의 부자연스러움 없이 이차원 상에서 그려진듯한 결과 영상을 만드는 라인 드로잉 시스템을 제시하였다[5]. 이 방법에서는 윤곽선 뿐 아니라 내부를 채우는 부분 역시 라인 드로잉으로 표현하였고, 또한 시스템의 각 단계 중 필요한 부분에 상호 작용적인 방법을 이용하여 보다 좋은 라인 드로잉 영상을 얻을 수 있도록 하였다.

삼차원 모델로부터 라인 드로잉을 할 때의 장점은 특징이 되는 선을 비교적 쉽게 라인 스트로크 형태로 얻

어낼 수 있을 뿐 아니라, 모델이나 카메라의 방향 등을 원하는대로 두고 그에 대한 결과를 얻을 수 있다는 점이다.

2.2 이차원 영상의 특징선 검출

이차원 영상은 기본적으로 규칙적 격자(regular grid) 상에서 색상 정보를 갖고 있을 뿐 그 영상에 있는 물체의 정보를 구체적으로 갖고 있는 것은 아니기 때문에 라인 스트로크를 구하기가 매우 어렵다. 이러한 점 때문에 이차원 영상에서 선과 관련된 연구는 영상의 어느 부분에서 색이 급격히 변하는지를 고려하여 특징선에 해당되는 부분을 픽셀 단위로 찾아내는 방법에 대한 연구, 즉 에지 검출(edge detection)에 대한 것이 대부분이었다.

에지 검출의 대표적인 알고리즘은 캐니 에지 검출(Canny edge detection) 방법이다[6]. 이 방법은 가우시언 필터링(Gaussian filtering)을 사용하여 스무딩(smoothing)된 영상의 그래디언트(gradient) 영상을 구하여 그래디언트 방향으로 에지의 법선 방향을 정의하고, 이 방향으로 그래디언트 값이 최대이고 특정 임계값(threshold) T_h보다 큰 그래디언트 값을 갖는 픽셀을 에지로 예측한다. 그리고 그래디언트 방향에 수직 방향으로 있는 픽셀들의 그래디언트 값이 임계값 T_l 이상인지 확인하며 픽셀 단위로 에지를 검출해간다. 이 때 T_h보다 작은 T_l을 사용하여 보다 효과적으로 에지를 검출해낼 수 있다. 하지만 이러한 에지에 해당하는 픽셀을 검출해내는 것만으로는 라인 스트로크를 얻을 수 없으므로 원하는 스타일로 다양하게 라인 드로잉을 하는 것이 불가능하다.

2.3 이차원 영상 및 비디오 스타일화

이차원 영상과 그를 확장한 비디오의 스타일화는 일반적으로 선보다는 내부의 색상 정보를 스타일화하여 사람이 그린 것처럼 표현하는데 치중된 연구가 진행되었다. Hertzmann이 제시한 회화적 렌더링(painterly rendering)이 그 대표적인 예이다[7]. 또한 선을 그려주는 경우라 하더라도, 선 부분은 재료의 특징을 살려주는 스타일화가 아닌 단순히 찾은 에지의 굵기를 적절히 변화있게 조절하여 잉크로 그린 것처럼 보이게 해주는 것으로 스타일화된 내부를 보조해주는 역할을 하는 정도였다.

DeCarlo와 Santella는 내부 스타일화를 위해 수행한 세그멘테이션(segmentation) 결과의 경계 부분을 캐니 에지 검출 과정에서 얻은 에지 체인(edge chain)[6]과 연동하여 스무딩(smoothing)된 특징선을 얻고, 이를 잉크로 그린 것처럼 스타일화 하는 방법을 제시하였다[8]. Wang 등은 DeCarlo와 Santella의 방법을 비디오로 확장하고, 영상 및 비디오 상에서의 여러 정보를 추가적으로 고려하여 특징선을 좀 더 다양한 굵기로 변화있게 스타일화하는 방법을 사용하였다[9]. 하지만 이러한 방식으로 얻어진 특징선은 세그멘테이션 결과의 영향력이 크고 지나치게 한정된 영역에서 단순화된 형태의 결과만이 나오게 된다. 따라서 내부 스타일화 결과를 보조하는 정도에 머물 뿐 독립적인 라인 드로잉에 이용하기에는 무리가 있다.

한편 Winnemöller 등은 비디오를 실시간에서 추상화하는 연구에서 DoG(difference-of-Gaussian) 연산과 몇몇 매개 변수를 이용하여 선택한 픽셀들의 집합이 두께가 자연스럽게 조절된 선처럼 보이게 하는 선 스타일화 방법을 사용하였다[10]. 하지만 이러한 방법은 라인 스트로크를 이용하여 얻을 수 있는 다양한 스타일의 라인 드로잉 결과를 얻는 것은 불가능하다는 단점이 있다.

3. 알고리즘 개괄

이 장에서는 이차원 영상의 라인 드로잉을 위한 전체 알고리즘을 개괄적으로 설명한다. 먼저 실제 사람이 이차원 영상에 대하여 어떠한 방식으로 라인 드로잉을 하는지 설명하고, 이를 이용하여 생성된 전체적인 알고리즘을 간단히 요약한다. 그리고 전체 알고리즘을 크게 세 단계로 나누어 각각에 대하여 간단하게 설명한다.

3.1 전체 개요

실제 사람이 이차원 영상을 보고 선을 그릴 때를 생각해 보자. 일단 영상을 보고 그 중 어느 부분에 선을 그릴지 판단하고, 선을 그려야 한다고 판단한 부분들을 가능한한 길고 부드럽게 연결하여 연필, 펜 등의 재료를 사용하여 선을 그려준다. 이 때 영상의 각 픽셀 정보를 개별적으로 받아들이기 보다는 비슷한 픽셀들을 묶어서 보고 그것으로 선을 만들려는 경향이 강하다. 그 결과 점 하나하나는 원래 영상에서 경계가 되어야 하는 부분과 다소 차이가 있더라도 전체적으로 부드럽고 길게 연결된 선이 완성된다. 이러한 과정과 최대한 유사한 방법으로 라인 드로잉 알고리즘을 만들면 사람이 실제 이차원 영상을 보고 그린 것에 가까운 결과를 얻을 수 있을 것이다.

사람은 영상에서 색이 급격하게 변하는 경계 부분에 색이 바뀌는 방향과 수직으로 하여 가능한한 부드럽게 연결된 선을 그리려는 경향이 강하다. 따라서 그래디언트 영상 상에서 그래디언트 값과 방향을 이용하여 각 픽셀에 대해 어느 부분에 어떤 방향으로 선이 그려질지와 이 선이 영향을 주는 영역을 지역적 우도(local likelihood)로 정의할 수 있다. 이러한 지역적 우도의 영향력을 각 픽셀에 대해 모두 더해 영상 전체에 대한 우도 함수(likelihood function)를 구하면, 기하적으로 봤을 때 이랑(ridge) 부분, 즉 법선(normal) 방향으로 지역적 우

도값이 극대(local maximum)인 부분에서 선이 그려지려는 경향이 크다. 이렇게 얻은 픽셀들 간의 연결 관계 및 연결 비용(connecting cost)을 정의하여 그래프(graph)로 만든 후, 하나의 선으로 그려져야 할 픽셀들이 모여서 하나의 클러스터(cluster)를 이루도록 클러스터링(clustering) 해준다. 그리고 각 클러스터마다 가능한 한 긴 선이 적은 연결 비용으로 연결되도록 라인 스트로크를 만들어 준다. 만들어진 라인 스트로크들은 곡선 근사(curve fitting)를 거친 후 다양한 텍스쳐를 이용한 텍스쳐 매핑(texture mapping)을 통해 스타일화가 가능하다.

이러한 작업의 장점은 영상을 주변의 픽셀들과 함께 선으로 정의하여 구성한 것이므로 선으로 연결하기 좋은 픽셀들이 찾아지고, 찾아진 픽셀들에 대해 길고 부드럽게 연결된 선이 되는 것을 목표로 하여 그래프 검색(graph search)을 적용하였으므로 실제 사람이 그린 것에 가까운 자연스러운 선이 나온다는 점이다.

3.2 단계별 과정

라인 드로잉 과정은 그림 3에서 볼 수 있듯이 크게 필터링, 선 연결, 스타일화의 세 단계로 진행된다.

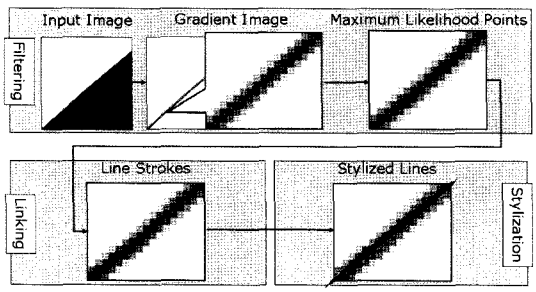


그림 3 단계별 과정

각 과정의 특징과 주요 아이디어는 다음과 같다.

- 필터링

  사람이 이차원 영상을 보고 어디에 선을 그려야할지 판단하는 단계에 해당한다. 이차원 영상 전체에 대해 각 픽셀들이 선을 형성하려는 정도를 우도 함수로 정의하고, 선으로 연결되어야 할 점들을 선택한다.

- 선 연결

  사람이 선을 그려야 한다고 판단한 부분을 가능한한 길고 부드럽게 연결하는 단계에 해당한다. 선으로 연결되어야 한다고 판단된 점들에 대해 연결 관계를 정의하고, 이들을 가능한한 길고 부드럽게 연결하도록 그래프 검색을 수행하여 라인 스트로크들을 얻는다.

- 스타일화

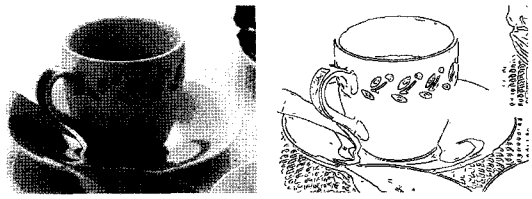
  사람이 다양한 재료를 이용하여 실제 선을 그리는 과정에 해당한다. 다양한 재료의 선 텍스쳐를 라인 스트로크 위에 매핑하여 실제 사람이 해당 재료를 사용하여 그린 것과 같은 효과를 얻는다.

이들에 대해서는 각각 4장, 5장, 6장에서 자세히 설명하겠다.

4. 필터링

이 장에서는 전체 과정 중 첫 번째 단계인 필터링 과정에 대하여 설명한다. 필터링 과정은 입력된 영상의 그래디언트 영상을 필터링하여 라인 스트로크가 쉽게 찾아질 수 있도록 하는 것을 목적으로 한다. 사람이 선을 그리는 과정 중 이차원 영상을 보고 선이 그려져야 할 부분을 판단하는 단계에 해당되며, 영상에서 각 픽셀이 선이 되려는 정도를 우도 함수로 표현하여 이를 판단하였다.

그림 4는 필터링 과정의 입력과 결과 영상이다. 즉, 그림 4(a)와 같은 이차원 영상이 입력으로 들어와 필터링 과정을 거치면, 그림 4(b)와 같이 부드럽게 연결되는 선을 생성할 가능성이 가장 높은 점들이 결과로 나오게 된다.



(a) 입력 영상 (b) 필터링 결과 영상

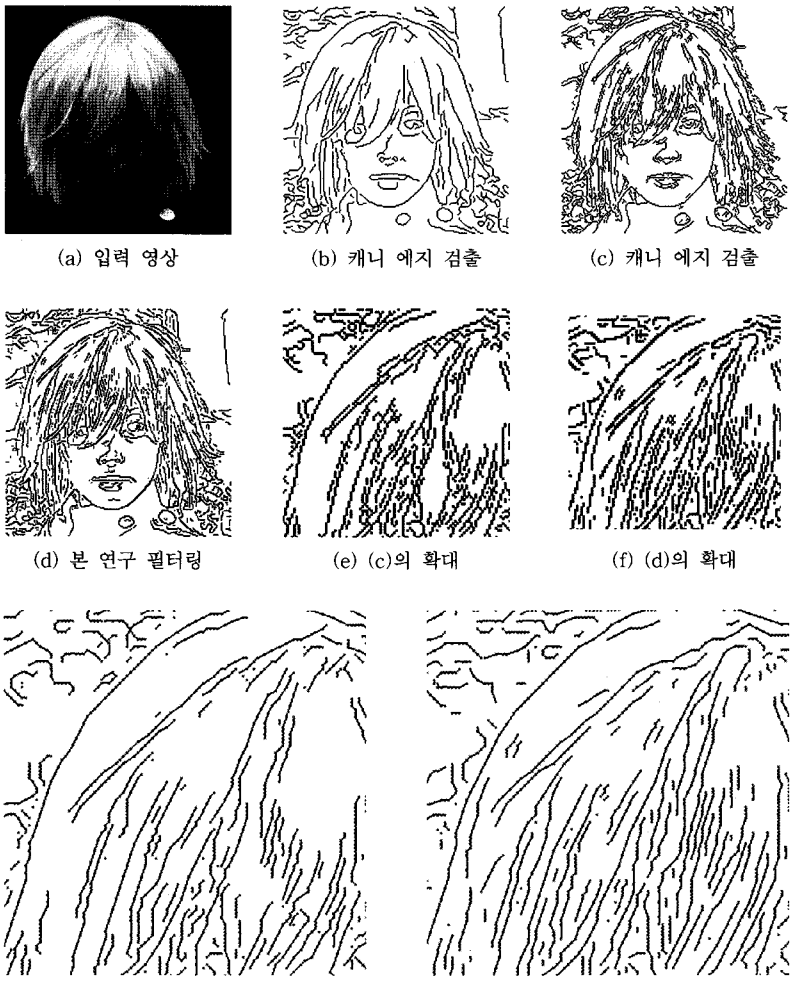
그림 4 필터링 과정 입출력 영상

이 장에서는 먼저 4.1장에서 라인 드로잉에 특화된 필터링의 필요성을 제시한 후, 4.2장에서 전체 필터링 과정을 요약한다. 그리고 각 과정에 대하여 4.3장, 4.4장, 4.5장, 4.6장에서 차례로 설명한다. 마지막으로 4.7장에서는 필터링을 수행한 결과 데이타들을 시각화하여 보여준다.

4.1 필요성

필터링 과정의 결과는 언뜻 보아서는 기존에 연구되었던 방법으로 에지를 찾아낸 결과와 큰 차이가 없어 보인다. 하지만 기존의 에지 검출 방법은 찾아낸 픽셀들을 선으로 연결하는 것에 대한 고려가 없었기 때문에 검출된 결과 픽셀들이 독립적이고, 그 때문에 선으로 연결하기에 적합하지 않은 결과가 많이 나온다. 따라서 선으로 연결될 것이라는 것을 고려하여 선이 그려져야 할 부분을 찾는 필터링 방법이 필요하다.

그림 5는 이러한 필터링의 필요성을 잘 보여준다. 그림 5(b)는 그림 5(a)에 대해 기본적인 캐니 에지 검출을



(a) 입력 영상 (b) 캐니 에지 검출 (c) 캐니 에지 검출

(d) 본 연구 필터링 (e) (c)의 확대 (f) (d)의 확대

(g) (c)의 선 연결 결과 (h) (d)의 선 연결 결과

그림 5 필터링 과정의 필요성

수행한 결과이다. 특징선 부분이 부드럽게 잘 찾아지긴 하였으나 사진을 보고 그릴 경우와 비교하면 머리카락 부분에 선이 지나치게 적은 것을 알 수 있다. 그림 5(c)는 머리카락 부분에서 충분한 특징선이 찾아지도록 변수를 조절하여 캐니 에지 검출을 수행한 결과이다. 이 경우 특징선으로 지정된 픽셀들은 서로 선으로 연결하기 매우 어려운 구조를 갖고 있다. 그에 비하여 본 논문에서 제시한 필터링 방법을 사용한 결과 영상인 그림 5(d)는 그림 5(c)와 비슷한 밀도의 픽셀들이 필터링 결과로 나왔으면서도 서로 선으로 연결되기 쉬운 구조를 갖고 있다. 그림 5(e), 그림 5(f)는 각각 그림 5(c), 그림 5(d)의 머리카락 부분을 알아보기 쉽도록 확대한 영상이다. 그림 5(e)의 캐니 에지 검출 결과에 비해 그림 5(f)의 필터링 결과가 선으로 연결되기 쉬운 구조를 갖고 있고, 이는 실제 선 연결을 한 결과로도 확인할 수 있다. 그림 5(g)와 그림 5(h)는 각각 그림 5(c)의 캐니 에지 검출 결과와 그림 5(d)의 본 연구의 필터링 결과를 선 연결 한 결과이다. 그림에서 볼 수 있듯이 본 연구의 필터링 결과가 보다 길고 실제 머리카락의 방향을 따라 부드럽게 잘 연결되는 경향을 보인다.

4.2 필터링 과정 요약

필터링 과정은 결국 사람이 이차원 영상에서 선으로 그려야 한다고 판단하는 부분이 어느 부분인지를 분석하는 것이다. 즉, 사람이 이차원 영상을 보고 어떠한 부분을 선으로 그려야한다고 인식하는지를 자동화한 과정이라고 할 수 있다.

사람은 우선 영상에서 색상이 급격하게 변하는 부분을 따라서 선으로 그리려고 한다. 즉, 각 픽셀 단위로

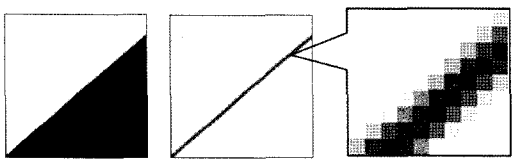
그래디언트 값이 큰 부분에서 그래디언트 방향과 수직으로 선을 그리려고 한다고 생각할 수 있다. 하지만 사람은 영상을 각각의 픽셀 단위가 아니라 주위와 묶어서 함께 보며, 그래디언트 값이 가장 크지는 않더라도 다른 픽셀과 부드럽게 연결되는 픽셀들을 선호한다. 그러므로 각 픽셀을 일정 범위 안에 있는 주위 픽셀과 함께 보면서 그래디언트 값을 가중치로 삼아 위치를 가중 평균내면 그 점에서 역시 가중 평균된 그래디언트 방향으로 선이 그려진다고 생각할 수 있다. 이것을 각 픽셀에 대한 지역적 우도로 정의하고 영상 전체 픽셀에 우도 함수를 구해주면 우도 함수 값이 큰 픽셀들이 선으로 그려져야 할 점들이 되고 이들은 전체적으로 부드럽고 자연스럽게 연결된다.

이상 설명한 필터링 과정을 알고리즘만 간단히 요약하면 다음과 같다. 일단 그래디언트 영상을 구한 후(4.3장), 각 픽셀에 대하여 일정 범위 이내의 주위 픽셀을 고려하여 어디에 선이 놓일지를 예상한다. 그리고 그 선이 어느 정도 영향력을 주는지를 결정하여 지역적 우도로 정의하고(4.4장), 모든 픽셀에 대하여 지역적 우도 값을 더해주면 영상 전체에서 어디에 선이 놓일지에 대하여 우도 함수가 완성된다(4.5장). 이 우도 함수 값이 법선 방향으로 극대인 점들이 선이 되려는 경향이 가장 높은 점들이므로 이를 찾으면 필터링 과정이 끝난다(4.6장). 이 때, 입력으로 비국소적 평균 노이즈 제거 방법(non-local means denoising method)를 통하여 노이즈(noise)가 제거된 영상을 사용하면 더 좋은 결과를 얻을 수 있다[11].

4.3 그래디언트 영상 구하기

사람은 선을 그릴 때 색이 갑자기 변하는 곳, 즉 그래디언트 값이 큰 부분에 그래디언트 방향과 수직으로 선을 그리려고 한다. 따라서 필터링 과정은 일단 입력 영상을 그래디언트 영상으로 바꾼 후 그 위에서 이루어진다. 그래디언트 영상을 구하는 것은 가장 일반적인 소벨 연산(Sobel operator)[12]을 사용하였다. 그림 6은 간단한 영상에 대하여 그래디언트 영상을 구해본 결과이다.

한 픽셀의 그래디언트 값은 그 픽셀 위에 선이 놓일 가능성, 즉 중요도가 얼마나 되는지를 나타낸다. 또한 그래디언트 방향은 선이 놓일 경우 그 법선 방향이 어느 쪽인지를 나타낸다. 따라서 그래디언트 영상에서의 한 픽셀은 위치와 법선 방향, 중요도를 가진 점이라고 생각할 수 있다.



(a) 이차원 영상 (b) 그래디언트 영상

그림 6 그래디언트 영상의 예

4.4 지역적 우도 정의

좌표가 i인 픽셀의 그래디언트 값을 g_i, 그래디언트 방향을 d_i로 둘 때, 이 픽셀은 위치 i에서 중요도 g_i, 법선 방향 d_i를 갖는 점 p_i라 정의할 수 있다. 점 p_i를 중심으로 반지름 h인 등방성 커널(isotropic kernel)에 대하여 그 안에 포함되는 점들의 위치와 법선 방향을 가중 평균(weighted average) 내면, 구해진 위치 c_i를 지나고 구해진 법선 방향 n_i에 수직인 맞춤선(fitting line) l_i를 구할 수 있다(그림 7). 이 맞춤선 l_i가 바로 점 p_i를 기준으로 거리가 h 이내인 점들을 봤을 때 그려지려는 선에 해당한다[13]. 이 때, p_i를 중심으로 한 등방성 커널 내의 점 p_j의 가중치 $w_i(p_j)$는 g_j의 값과 n_i, n_j의 유사도로 결정된다.

$$w_i(p_j) = g_j \cdot normalize(d_i \cdot d_j) \quad (1)$$

$$c_i = \frac{\sum_{p_j \in \{p_j \mid \|p_i - p_j\| \le h\}} w_i(p_j) \cdot p_j}{\sum_{p_j \in \{p_j \mid \|p_i - p_j\| \le h\}} w_i(p_j)} \quad (2)$$

$$n_i = normalize\Big(\sum_{p_j \in \{p_j \mid \|p_i - p_j\| \le h\}} w_i(p_j) \cdot d_j \Big) \quad (3)$$

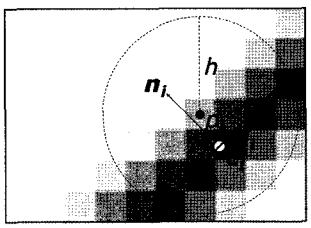


그림 7 맞춤선 정의

맞춤선이 정의되면 해당 맞춤선 l_i가 영향을 주는 영역을 정의한다. 이는 c_i를 원점으로 하여 l_i와 n_i를 각각 장축, 단축으로 두고, p_i를 중심 한 등방성 커널 안에 들어오는 점들을 각 축에 투영하여 가중 평균 낸 값을 해당 축 방향의 크기로 갖는 이방성 커널(anisotropic kernel)을 정의하여 만들어준다(그림 8). 이는 장축, 단축을 나타내는 행렬 D_i와 각 축 방향으로의 크기를 나타내는 A_i를 이용하여 이방성 커널에 해당하는 공분산 행렬(covariance matrix) C_i를 만들어 줌으로써 정의할 수 있다[14]. 구해진 공분산 행렬을 이용하면 해당 이방성 커널에 들어오는 점들은 식 (12)와 같이 정의할 수 있다[13].

$$D_i = \begin{pmatrix} -(n_i)_y & (n_i)_x \\ (n_i)_x & (n_i)_y \end{pmatrix} \quad (4)$$

$$(q_{ij})_x = \left(\overrightarrow{c_i p_j} \cdot ((D_i)_{11}, (D_i)_{21}) \right) \quad (5)$$

$$(q_{ij})_y = \left(\overrightarrow{c_i p_j} \cdot \left((D_i)_{12}, (D_i)_{22}\right)\right) \quad (6)$$

$$q_i(p_j) = \left((q_{ij})_x, (q_{ij})_y\right) \quad (7)$$

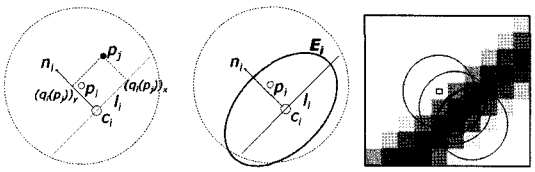
$$(a_i)_{11} = \frac{\sum_{p_j \in \{p_j \mid \|p_i - p_j\| \le h\}} w_i(p_j) \cdot (q_i(p_j))_x}{\sum_{p_j \in \{p_j \mid \|p_i - p_j\| \le h\}} w_i(p_j)} \quad (8)$$

$$(a_i)_{22} = \frac{\sum_{p_j \in \{p_j \mid \|p_i - p_j\| \le h\}} w_i(p_j) \cdot (q_i(p_j))_y}{\sum_{p_j \in \{p_j \mid \|p_i - p_j\| \le h\}} w_i(p_j)} \quad (9)$$

$$A_i = \begin{pmatrix} (a_i)_{11} & 0 \\ 0 & (a_i)_{22} \end{pmatrix} \quad (10)$$

$$C_i = \sqrt{2} \cdot D_i \cdot A_i \cdot D_i^T \quad (11)$$

$$E_i = \{x : (x - c_i)^T \cdot C_i^{-1} \cdot (x - c_i) \le 1\} \quad (12)$$



(a) 장/단축 투영 (b) 이방성 커널 (c) 영상에서의 결과

그림 8 이방성 커널 생성

이렇게 만들어진 이방성 커널과 l_i까지의 거리를 이용하여 l_i의 영향력을 나타내는 지역적 우도 L_i를 정의할 수 있다. l_i에 대한 점 p_j에서의 지역적 우도 값을 $L_i(p_j)$라고 할 때, $L_i(p_j)$는 식 (14)와 같이 정의된다. 이 때, Φ_i는 이방성 커널로부터 정의되는 커널 함수로, c_i에서 최대값을 갖고 타원의 경계 방향으로 갈수록 그 값이 감소하며 경계와 그 외부에서는 0를 갖는다.

$$\Phi_i(p_j - c_i) = 1 - (p_j - c_i)^T \cdot C_i^{-1} \cdot (p_j - c_i) \quad (13)$$

$$L_i(p_j) = \Phi_i(p_j - c_i) \cdot \left(h^2 - \left((p_j - c_i) \cdot n_i\right)^2\right) \quad (14)$$

이를 영상 상의 각 점에 대하여 수행하면, 전체 영상에서 각 점에 대한 맞춤선과 지역적 우도 값을 구할 수 있다(그림 9).

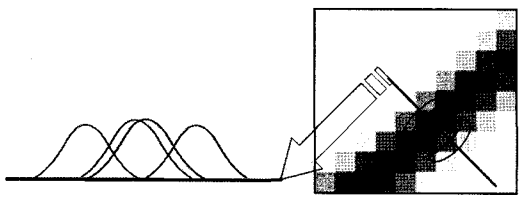


그림 9 지역적 우도 : 영상 상의 네 점에 대하여 지역적 우도를 구한 예

4.5 우도 함수 구하기

4.4장에서 구한 지역적 우도 값을 가중합 하면 영상 전체에 대한 우도 함수 L을 얻을 수 있다(그림 10).

점 p_i의 l_j에 대한 지역적 우도 값은 $L_j(p_i)$로 정의된다. 이 때, 점 p_i에서의 우도 함수 값 $L(p_i)$는 점 p_i를 자신의 이방성 커널 범위 내에 갖는 모든 맞춤선 l_j에 대해 점 p_i의 위치에 대응되는 지역적 우도 값 $L_j(p_i)$의 가중합이 된다. 이 때, 지역적 우도 L_j에 대한 가중치 w_j는, 해당 이방성 커널의 중심인 c_j에서의 그래디언트 값으로 정의된다.

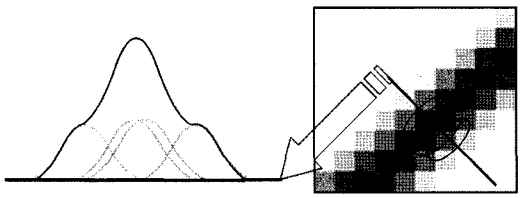


그림 10 우도 함수 : 영상 상의 네 점의 지역적 우도 값을 가중합하여 우도 함수를 구한 예

$$w_j = g_{c_j} \quad (15)$$

$$L(p_i) = \sum_{p_j \in \{p_j \mid \Phi_j(p_i - c_j) > 0\}} w_j \cdot L_j(p_i) \quad (16)$$

이 과정에서 우도 함수 값을 구하는 것과 마찬가지의 방법으로 점 p_i의 법선 방향 n_i를 개선해준다.

$$n_i = normalize\left(\sum_{p_j \in \{p_j \mid \Phi_j(p_i - c_j) > 0\}} w_j \cdot L_j(p_i) \cdot n_j\right) \quad (17)$$

이를 영상 상의 모든 점에 대해 수행하여 각 점의 우도 함수 값과 개선된 법선 방향을 얻을 수 있다. 이 때, 우도 함수는 최대값이 1이 되도록 정규화(normalization)하여 사용한다.

이렇게 구한 우도 함수는 영상에서 어느 부분에 선이 그려지려는 경향이 강한지를 나타낸다.

4.6 법선 방향으로의 극대점 찾기

점 p_i에서의 우도 함수 값 $L(p_i)$가 법선 방향 n_i로 양 옆에 있는 두 가상의 점 p_a, p_b의 우도 함수 값보다 클 경우, 점 p_i는 법선 방향으로의 극대점이 된다. 가상의 점 p_a, p_b의 우도 함수 값은 해당 위치의 양 옆 점의 우도 함수 값을 보간(interpolation)하여 정의한다.

이러한 방법으로 전체 영상에서 법선 방향으로 극대인 점들, 즉 우도 함수를 기하적 관점에서 보았을 때 그림 11과 같이 이랑에 해당되는 점들을 모두 구해준다.

4.7 필터링 결과

그림 12는 필터링 과정에서 나온 결과 영상이다. 그림

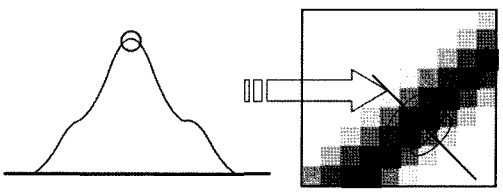
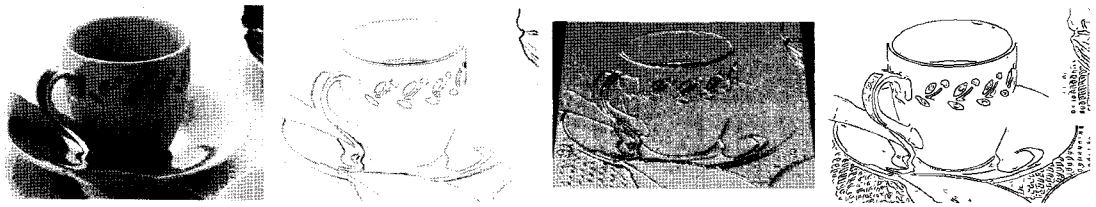


그림 11 법선 방향으로의 극대점



(a) 입력 영상 (b) 우도 함수 명암화 (c) 우도 함수 기하화 (d) 극대점 영상

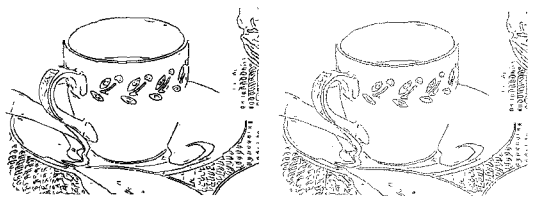
그림 12 필터링 결과 영상

12(a)와 같은 입력 영상에 대해 영상 전체에서의 우도 함수를 구해 그것을 각각 그레이 스케일(gray scale)과 고도장(height field)을 이용해 시각화한 것이 그림 12(b)와 그림 12(c), 필터링 과정의 최종 결과인 법선 방향으로의 극대점들을 나타낸 것이 그림 12(d)이다.

5. 선 연결

이 장에서는 전체 과정 중 선 연결 과정에 대하여 설명한다. 선 연결 과정은 법선 방향으로 우도 함수값이 극대인 점들을 연결하여 라인 스트로크를 만드는 것을 목적으로 한다. 사람이 선을 그리는 과정 중 선을 그려야 한다고 판단한 부분을 가능한한 길고 부드럽게 연결하는 과정에 해당되며, 이를 위해 각 점들간의 연결 관계를 그래프로 정의, 그 위에서 그래프 검색을 수행하는 방법을 정의하였다.

그림 13은 선 연결 과정의 입력과 결과 영상이다. 즉, 그림 13(a)와 같은 필터링 결과가 입력으로 들어왔을 경우, 선 연결 과정을 거치면 그림 13(b)와 같은 라인 스트로크들이 결과로 나온다.



(a) 입력 영상 (b) 선 연결 결과 영상

그림 13 선 연결 과정 입출력 영상

이 장에서는 5.1장에서 먼저 선 연결 과정을 요약한 후, 각 과정에 대하여 5.2장, 5.3장, 5.4장에서 차례로 설명한다. 그리고 선 연결 과정에서 나온 데이타들을 시각화한 영상들을 마지막 5.5장에서 보여준다.

5.1 선 연결 과정 요약

선 연결 과정은 사람이 선을 그려야 한다고 판단한 점들 중 비슷한 성향을 띄는 점들을 묶어서 하나의 선으로 표현하는 과정이다. 사람이 봤을 때 점들이 서로 비슷한 성향이라고 판단하는 기준은 선을 그리려는 방향이 얼마나 비슷한지, 거리가 얼마나 가까운지 등이다. 따라서 이러한 기준을 적용하여 K-최근린 그래프(K-nearest neighbor graph)를 만들고 클러스터링을 수행하여 하나의 선이 되려는 점들을 각각의 클러스터로 나누어주면 된다. 이러한 각 클러스터 내의 점들에 대해 사람은 가능한한 부드럽고 길게 연결된 선을 그려주려고 한다는 점을 적용, 그래프 상에서 거리가 멀리 떨어진 두 점이 최소한의 비용으로 연결될 수 있도록 그래프 검색(graph search)을 통해 라인 스트로크를 얻는다.

설명한 선 연결 과정을 간단히 요약하면 다음과 같다. 일단 K-최근린 그래프를 만들어 점들간의 연결 관계를 정의해준다(5.2장). 그 후 클러스터링을 통해 하나의 선으로 연결될 부분들이 하나의 클러스터에 들어가도록 해준 후(5.3장), 각각의 클러스터에서 가능한한 길고 부드럽게 연결된 라인 스트로크를 만들어낸다(5.4장).

5.2 K-최근린 그래프 생성

4장에서 설명한 필터링 결과 나온 점들은 선으로 연결되어야 할 부분이 픽셀 단위로 나열된 것에 불과하다. 이를 부드러운 라인 스트로크로 만들기 위해서는 일단 각 점들 사이의 연결 관계 및 연결 비용을 정의해 주어야 한다.

두 점 p_i, p_j에 대해 두 점을 연결하는 벡터를 $s_{ij} = \overrightarrow{p_i p_j}$, 법선 방향의 유사도를 $\delta_{i,j} = 1 - normalize(n_i \cdot n_j)$라 둘 때, 두 점 사이의 연결 비용 $cost_{ij}$는 식 (18)과 같이 정의된다.

$$cost_{ij} = \|s_{ij}\| \cdot \delta_{i,j} \cdot max(|n_i \cdot s_{ij}|, |n_j \cdot s_{ij}|) \quad (18)$$

각 점에 대해 연결 비용이 가장 작은 K개의 점들을 찾아 연결하면 K-최근린 그래프를 만들 수 있다.

5.3 클러스터링

연결 관계가 정의되어 그래프로 표현된 점들은 그래프 검색를 통해 라인 스트로크로 만들어질 수 있다. 하지만 그래프 상에는 분기(branch)나 급격하게 꺾이는 점들 사이에도 연결 관계가 있기 때문에 그대로 하나의 라인 스트로크로 연결하여서는 원하는 라인 스트로크를 모두 찾아낼 수가 없다. 따라서 라인 스트로크를 찾기에

앞서 하나의 선으로 연결되어야 할 부분이 하나의 클러스터에 들어가도록 클러스터링 해주는 과정이 필요하다.

초기 시드(initial seed)는 그래프 상에서 연결 관계가 있는 각 부분(connected component)에 하나씩 생성하여 준다. 그래프 상에 연결 부분이 N개 있다면 N개의 클러스터가 있는 상태에서 시작되는 것이다. 그리고 각 클러스터의 대표선(proxy line) $proxy_k$를 이용하여 분기나 급격하게 꺾이는 부분이 있는지 확인한다. 이는 클러스터 내의 모든 점들을 대표선에 투사하여, 가까운 위치에 투사된 점들의 높이 차가 특정 값 이상인지를 확인하면 알 수 있다. 이러한 부분이 존재할 경우 해당 부분에 시드 $seed_k$와 $seed_{k'}$을 재생성하고, 해당 시드들로부터 연결 비용이 적은 이웃 점들을 같은 클러스터로 병합하여 간다. 대부분의 경우 분기나 급격하게 꺾인 부분에서 연결 비용이 크기 때문에 해당 부분을 경계로 서로 다른 클러스터로 나뉘게 된다. 이 과정은 그림 14와 같다. 전체 그래프 상에서 클러스터의 수가 더이상 늘지 않을 때까지 이를 반복하면 분기나 급격하게 꺾인 부분이 모두 다른 클러스터로 분리되어 있는 클러스터링 결과를 얻을 수 있다.

다음은 클러스터링 과정에서 필요한 세부 연산 방법들이다.

클러스터 k에 대한 대표선 $proxy_k$는, 클러스터 k에 포함되어 있는 모든 점들의 위치와 법선 방향을 평균내어 나온 위치 c_k와 법선 방향 n_k에 대하여, c_k를 지나고 n_k에 수직인 직선으로 정의된다. 해당 클러스터의 시드 $seed_k$는 클러스터 내에서 위치와 법선 방향이 각각 c_k, n_k와 가장 유사한 점으로 정의된다.

클러스터링 과정에서 어떤 점 p_i가 어느 클러스터에 포함될지는, p_i와 연결 관계가 있는 점들 중 클러스터링 비용(clustering cost)이 가장 작은 점이 어느 클러스터에 포함되어 있는지에 따라 결정된다. 따라서 한 클러스터 내의 점들은 중간에 끊어지는 일 없이 연결될 수 있다[15]. 이 때, p_i의 p_j에 대한 클러스터링 비용 $cost_i(p_j)$는 식 (19)와 같이 p_i와 p_j의 법선 방향 유사도 및 p_i로부터 p_j를 지나고 법선 방향에 수직인 직선 s_j까지의 거리가 짧은 정도에 따라 결정된다(그림 15).

$$cost_i(p_j) = |\overrightarrow{p_i p_j} \cdot n_j| \cdot \left(1 - normalize(n_i \cdot n_j)\right) \quad (19)$$

그림 16은 간단한 영상의 클러스터링 예이다. 그림 16(a)와 같은 영상에 대해, K-최근린 그래프를 생성(5.2장)한 직후 연결 관계가 있는 부분을 같은 색상으로 시각화 해보면 그림 16(b)와 같다. 이는 ㅅ모양으로 된 부분이 서로 연결관계를 갖고 한 클러스터에 포함되어 있기 때문에 그대로는 하나의 라인스트로크를 만드는 것이 불가능하다. 이에 대해 클러스터링을 수행하면 그림 16(c)와 같이 해당 부분이 서로 다른 클러스터로 나누어지는 결과가 나오고 각 클러스터에 하나의 선을 그려주면 원하는 결과를 얻을 수 있게 된다.

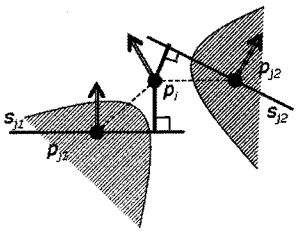


그림 15 클러스터링 비용 결정

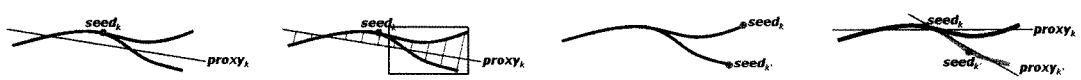
(a) 입력 영상 (b) 클러스터링 이전 (c) 클러스터링 이후

그림 16 클러스터링의 예

5.4 라인 스트로크 생성

라인 스트로크는 클러스터링(5.3장) 결과로 나온 클러스터들에서 각각 하나씩 생성해준다. 가능한한 부드럽고 길게 연결된 라인 스트로크 생성이 목적이므로 그에 맞는 조건으로 그래프 검색을 수행한다. 이는 그래프 상에서 최단 경로(shortest path)를 찾는 Dijkstra의 알고리즘(Dijkstra's algorithm)을 확정되지 않은 다수의 목표점에 대하여 수행할 수 있도록 확장하는 방법으로 구현되었다[16].

그래프 검색의 시작점은 해당 클러스터 내에서 선이 되려는 경향이 가장 큰 점, 즉 우도 함수 값이 가장 큰 점으로 잡는다. 또한 클러스터 내의 어떤 점 p_i는 시작점까지의 연결 비용 합(total cost) x_i를 저장하고 있고, 이 값은 무한대로 초기화 되어 있다. 이러한 조건에서 검색 리스트에 시작점 p_s를 넣은 후 x_s를 0으로 초기화 시키고 검색 리스트가 빌 때까지 다음 알고리즘을 반복하여 수행하면 된다.



(a) 분기가 있는 클러스터 (b) 분기 검출 (c) 시드 재생성 (d) 클러스터 분리

그림 14 클러스터링 과정

검색 리스트에 들어있는 점들 중 연결 비용 합이 가장 작은 점 p_i에 대하여, p_i와 연결 비용 $cost_{ij}$(5.2장의 식 (18)에서 정의됨)의 연결 관계를 갖는 점 p_j의 시작점까지 연결 비용 합은 $x_i + cost_{ij}$와 x_j 중 작은 값이 된다. p_i와 연결 관계가 있는 모든 점에 대하여 이 값을 갱신한 후, 검색 리스트에서 p_i를 제거하고 p_i와 연결 관계가 있는 점들 중 검색 리스트에 들어있지 않은 점들을 검색 리스트에 추가하여 준다.

이 과정을 거치면 해당 클러스터 내의 모든 점들은 시작점까지 가장 부드럽게 연결된 경로와 그 때의 연결 비용 합을 갖게 된다. 연결의 끝에 있는 다수의 점들 중 서로간의 거리(p_s까지 연결 경로의 합)가 가장 큰 두 점 p_{d1}, p_{d2}를 선택하여 시작점 p_s까지의 연결 경로를 따라 라인 스트로크를 만들면 해당 연결 비용이 최소가 되어 가능한한 부드러우면서도 길게 연결된 라인 스트로크를 얻을 수 있다(그림 17).

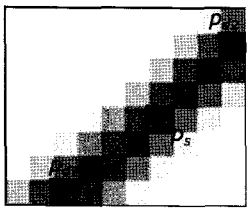


그림 17 라인 스트로크 생성

5.5 선 연결 결과

그림 18은 선 연결 과정에서 나온 결과 영상이다. 그림 18(a)와 같이 입력으로 들어온 필터링 결과 영상에 대하여 클러스터링을 수행하고 각 클러스터를 각각 다른 색상으로 시각화한 것이 그림 18(b), 선 연결 과정의 최종 결과인 라인 스트로크가 그림 18(c)이다. 그리고 그림 18(b), 그림 18(c)를 부분만 확대한 것이 각각 그림 18(d), 그림 18(e)이다.

6. 스타일화

이 장에서는 전체 과정 중 마지막 단계인 스타일화 과정에 대하여 설명한다. 스타일화 과정은 선 연결 단계에서 구한 라인 스트로크를 곡선 근사하고 그 위에 여러 가지 텍스쳐를 입혀 사람이 손으로 그린 것 같은 영상을 만드는 것을 목적으로 한다. 사람이 선을 그리는 과정 중 실제 재료를 이용하여 선을 그리는 과정에 해당되며, 이를 위하여 곡선 근사 및 텍스쳐 매핑을 이용하였다.

그림 19는 스타일화 과정의 입력과 출력 영상이다. 즉, 그림 19(a)와 같은 선 연결 결과가 입력으로 들어왔을 때, 그림 19(b)와 같이 사람이 그린 것과 같은 라인 드로잉이 결과로 나온다.

이 장에서는 6.1장에서 먼저 스타일화 과정을 요약한 후, 각 과정을 6.2장, 6.3장에서 설명한다. 마지막 6.4장에서는 스타일화 과정에서 나온 영상들을 보여준다.

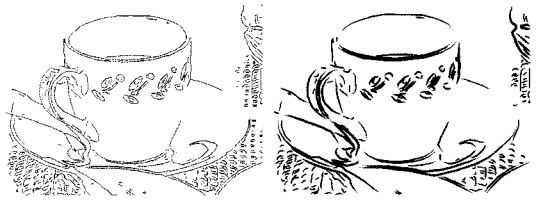
6.1 스타일화 과정 요약

스타일화 과정은 라인 스트로크를 따라 실제 선을 그리는 과정이다. 하지만 픽셀 단위로 이루어진 이차원 영상의 한계 때문에 연결된 라인 스트로크는 실제 사람이 그리는 선에 비해 부드럽지 못하다. 따라서 이를 해결하기 위해 곡선 근사를 먼저 수행해준다. 그리고 실제 사

(a) 입력 영상 : 필터링 결과 영상 (b) 클러스터링 영상 (c) 라인 스트로크 영상

(d) 클러스터링 영상 확대 (e) 라인 스트로크 영상 확대

그림 18 선 연결 결과 영상



(a) 입력 영상 (b) 스타일화 결과 영상

그림 19 스타일화 과정 입출력 영상

람이 사용하는 재료로 인해 그려지는 선은 두께를 갖고 있으므로 근사된 곡선을 법선 방향으로 일정 폭만큼 늘려서 삼각형 스트립(triangle strip)을 이용, 실제 선의 텍스쳐를 매핑 해준다. 이 과정을 수행하면 영상 상에 선으로 표현되어야 할 부분에 원하는 재료의 부드럽게 연결된 선이 그려진다.

단, 우도 함수값이 지나치게 작거나 너무 짧은 라인 스트로크의 경우, 이를 그려주지 않는 것이 결과 영상의 질에 도움이 된다. 따라서 사용자로부터 우도 함수값에 대한 임계값 $T_{likelihood}$와 라인 스트로크 길이에 대한 임계값 T_{length} 값을 입력받아 각 라인 스트로크를 결과 영상에 그려줄지 결정하는 기준으로 삼는다.

요약하면, 스타일화 과정에서는 일단 입력으로 들어온 라인 스트로크 중 사용자로부터 입력받은 임계값 기준을 만족시키는 라인 스트로크들을 부드러운 곡선으로 근사한다(6.2장). 그리고 생성된 각 점에서 법선 방향으로 일정 폭만큼 늘려 만든 공간에 원하는 텍스쳐를 매핑하여 다양한 스타일의 라인 드로잉을 만든다(6.3장).

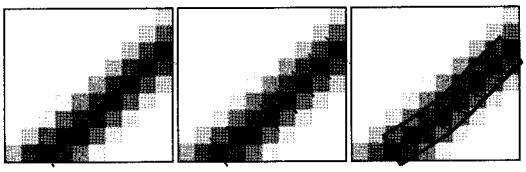
6.2 곡선 근사

5장의 선 연결 결과 나온 라인 스트로크들은 규칙적 격자 위의 픽셀 단위로 얻어진 것이기 때문에 정확한 위치를 표현하는데 한계가 있다. 따라서 라인 스트로크를 그대로 사용하면 결과 라인 드로잉에서 자연스럽게 연결되지 못하고 튀는 부분이 생긴다. 그러므로 좀 더 자연스러운 결과를 위해서는 라인 스트로크로 연결된 점들 중 중요성이 떨어지는 점들을 몇몇 제거하고 곡선 근사하는 과정이 필요하다. 여기서 각 점의 중요성은 해당 부분에서 선의 진행 방향이 얼마나 급격하게 변하는지로 생각할 수 있고, 이것은 법선 방향의 변화로 판단이 가능하다.

곡선 근사를 위해서는 일단 라인 스트로크 상의 점들을 차례로 따라가면서 곡선 근사에 이용한 점들을 추려내야 한다. 두 점 사이의 거리와 법선 방향의 유사도를 고려하여 정의된 연결 비용을 더해가며 그 합이 임계치를 넘을 때마다 해당 점을 선택하고 연결 비용의 합을 다시 0으로 돌려준다. 이러한 방법을 사용하면 선 진행 방향의 변화가 적은 부분에서는 상대적으로 적은 수의 점을 선택하고, 변화가 큰 부분에서는 많은 수의 점을 선택할 수 있으므로 곡선 근사 이전의 라인 스트로크 상에서 특징되는 부분들을 손상시키지 않을 수 있다. 그 후 선택된 점들을 커멀롬 스플라인(Catmull-Rom spline)을 이용하여 곡선 근사하였다[17].

6.3 텍스쳐 매핑

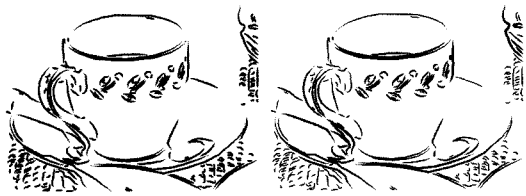
근사된 곡선 상의 점들을 곡선에 수직으로 일정 폭만큼 늘려주면 삼각형 스트립을 이용하여 해당되는 굵기의 선을 텍스쳐 매핑할 수 있다(그림 20).



(a) 근사된 곡선 (b) 폭 늘림 (c) 삼각형 스트립

그림 20 텍스쳐 매핑 과정

이 때, 우도 함수 값을 이용하여 해당 선의 중요도를 판단하고 그에 따라 각 선의 굵기를 달리 해주면 좀 더 사람이 그린 것 같은 좋은 결과를 얻을 수 있다(그림 21). 이는 사람이 실제 선으로 물체를 그릴 때 중요한 부분일수록 눈에 잘 띄게 그려주는 경향이 있기 때문이다.



(a) 선 굵기 일정 (b) 선 굵기 변화

그림 21 선의 굵기 변화에 따른 텍스쳐 매핑 결과 영상

6.4 스타일화 결과

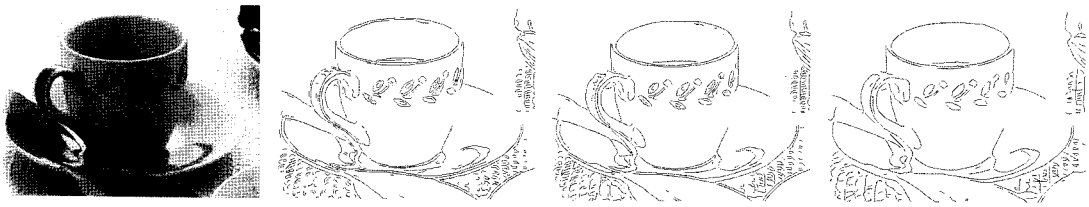
그림 22는 스타일화 과정에서 나온 결과 영상이다. 그림 22(a)와 같이 입력으로 들어온 선 연결 결과 영상에 대하여 곡선 근사한 것이 그림 22(b)이고, 여기에서 각 선의 우도 함수 값을 고려하여 다양한 굵기로 붓 선 텍스쳐를 매핑해준 것이 그림 22(c)이다.

7. 실험 결과

이 장에서는 본 연구의 결과 영상을 살펴본다. 7.1장에서는 사용자의 입력에 따른 결과의 상세도(LOD:level of detail)변화를 살펴보고, 7.2장에서는 다양한 텍스쳐를 이용한 스타일 변화를 확인한다. 그리고 7.3장에서는 본 연구의 결과로 나온 몇몇 영상들을 보여준다.

(a) 입력 영상 : 선 연결 결과 영상 (b) 곡선 근사 영상 (c) 텍스쳐 매핑 영상

그림 22 스타일화 결과 영상



(a) 입력 영상 (b) h = 1.5 (c) h = 3 (d) h = 5

그림 23 h에 따른 상세도 변화 ($T_{likelihood}$ = 0.01, T_{length} = 5)

7.1 사용자 입력 변수를 이용한 상세도 변화

전체 과정 중에서 사용자가 입력하는 변수는 4.4장에서 주위에 함께 고려할 픽셀들의 범위를 지정하기 위하여 사용되는 등방성 커널의 크기 h와 스타일화 단계에서 우도 함수값이 지나치게 작거나 라인 스트로크의 길이가 너무 짧은 경우를 제외하기 위해 이용되는 임계값 $T_{likelihood}$와 T_{length}, 총 세 가지가 있다. 이들을 적당히 조절하면 결과 영상에서 각 부분을 상세하게 그려주는 정도, 즉 상세도를 변화시킬 수 있다.

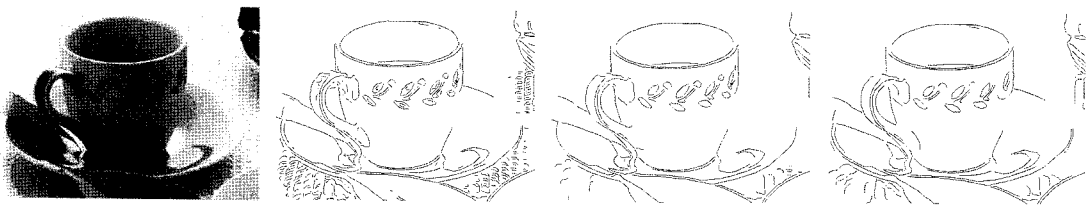
그림 23은 사용자 입력 변수 중 등방성 커널의 크기 h 값을 변화시킨 결과이다. 결과를 알아보기 쉽도록 텍스쳐 매핑 이전의 곡선 근사 상태로 표현하였고, $T_{likelihood}$는 0.01, T_{length}는 5로 고정해둔 상태이다. h 값이 커질수록 결과 영상이 단순화되는 것을 알 수 있다.

그림 24는 h값을 3으로 고정해둔 상태에서 그려줄 최소 우도 함수 값 $T_{likelihood}$와 최소 선 길이 T_{length}를 바꾸어본 결과이다. 마찬가지로 $T_{likelihood}$, T_{length}의 값이 커질수록 결과의 상세도가 떨어지는 것을 알 수 있다. 결과를 알아보기 쉽도록 텍스쳐 매핑 없이 곡선 근사만 수행하였다.

그림 24의 결과 영상들은 400 × 294 영상에 대하여 3.6GHz Pentium 4 프로세서, 2GB 메모리의 윈도우즈 환경에서 얻은 것으로, 결과를 얻기까지 총 15초 정도가 소요 되었다. 그 중 필터링 과정에서 3초, 선 연결 과정에서 12초 정도가 소요 되었고, 스타일화 과정은 거의 실시간으로 이루어졌다.

7.2 텍스쳐 종류에 따른 스타일 변화

본 연구의 주요 목적 중 하나인 라인 스트로크를 이용하여 다양한 스타일의 텍스쳐를 매핑하면 원하는 기법의 라인 드로잉 결과를 얻을 수 있다. 그림 25는 같은 곡선 근사 결과에 다양한 선 텍스쳐를 매핑해본 결과이다. 그림 25(a)를 입력 영상으로 하여 등방성 커널의 크기 h는 3, 최소 우도 함수 값 $T_{likelihood}$는 0.01, 최소 선 길이 T_{length}는 7로 두었을 때, 먹선 텍스쳐를 입힌 결과 영상이 그림 25(b), 크레용 텍스쳐를 입힌 결과 영상이 그림 25(c)이다. 그림 25(d)는 수채 물감 선 텍스쳐를 매핑한 결과이고, 그림 25(e)와 그림 25(f)는 각각 연필선 텍스쳐와 유화를 나이프로 그린 선 텍스쳐를 근사된 곡선에 입힌 결과 영상이다. 각 결과에서 사용된 텍스쳐는 결과 영상의 상단에 함께 제시되었다.

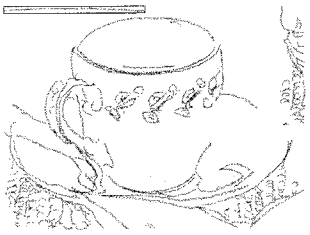


(a) 입력 영상 (b) T_{like} = 0.01, T_{length} = 5 (c) T_{like} = 0.12, T_{length} = 5 (d) T_{like} = 0.01, T_{length} = 15

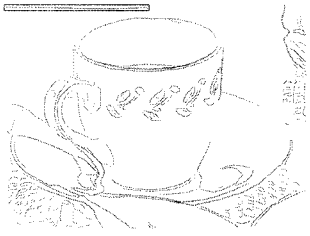
그림 24 $T_{likelihood}$ 및 T_{length}에 따른 상세도 변화 (h = 3)


(a) 입력 영상


(b) 먹선 텍스쳐 매핑


(c) 크레용 텍스쳐 매핑


(d) 수채 물감 선 텍스쳐 매핑


(e) 연필선 텍스쳐 매핑


(f) 나이프 선 텍스쳐 매핑

그림 25 다양한 텍스쳐 매핑 결과

7.3 여러 가지 결과 영상

다음은 다양한 영상에 여러 가지 텍스쳐를 매핑하여 라인 드로잉 결과를 얻은 것이다. 그림 26은 상자에 놓인 캔 사진에 펜선 텍스쳐를 매핑하여 라인 드로잉 한 결과이다. 캔 앞면에 있는 그림이 가느다란 펜선으로 잘 표현된 것을 알 수 있다. 그림 27은 집 사진에 대해 먹선 텍스쳐를 입혀 라인 드로잉 한 결과이다. 집의 구조는 굵은 직선으로 잘 표현되고 그 외의 수풀 등 자잘한 부분은 가는 선으로 표현된 것을 알 수 있다. 그림 28은 곡선과 직선이 두드러진 실내 사진에 대하여 마커 텍스쳐를 이용하여 라인 드로잉 한 결과이다. 부드러운 곡선과 직선이 확실히 드러나며 약간 투명한 느낌의 마커 질감이 잘 살아있다. 그림 29는 몇몇 소품을 찍은 사진을 수채 물감 선 텍스쳐를 이용하여 라인 드로잉 한 것이다. 주요한 선 외에 리본의 주름 등도 잘 표현되어 있다. 그림 30은 식물 열매와 잎을 목탄 텍스쳐를 이용하여 라인 드로잉한 것이다. 열매의 줄기 부분과 잎이 목탄 특유의 부드러운 질감으로 잘 표현되었다. 그림 31은 접사 촬영된 꽃 사진에 먹선 텍스쳐를 매핑하여 라인 드로잉 한 것이다. 꽃과 줄기, 잎 부분이 잘 표현되었고, 배경의 산 부분도 하나의 선으로 부드럽게 그려졌다. 그림 32는 컵과 식물이 함께 있는 사진을 간단한 사인펜 선으로 라인 드로잉 한 결과이다. 사인펜의 투박한 느낌의 선이 중요한 부분을 잘 표현해주고 있다. 그림 33은 시계탑 내부에서 바깥을 향하여 찍은 사진에 대하여 매직펜선으로 라인 드로잉 한 결과이다. 가까이에 있는 구조물들이 굵은 직선으로 잘 표현되었고 외부의 건물 풍경도 어느 정도 구조를 갖고 가는 선으로 그려져 있다.

8. 결론 및 향후 연구


(a) 입력 영상

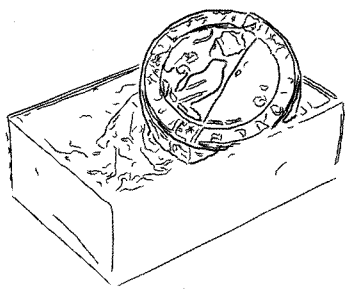

(b) 펜선 텍스쳐 매핑 결과 영상

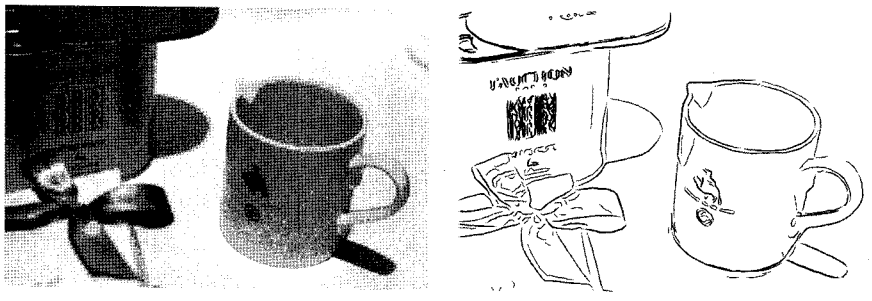
그림 26 펜선 라인 드로잉

(a) 입력 영상 (b) 먹선 텍스쳐 매핑 결과 영상

그림 27 먹선 라인 드로잉

(a) 입력 영상 (b) 마커 텍스쳐 매핑 결과 영상

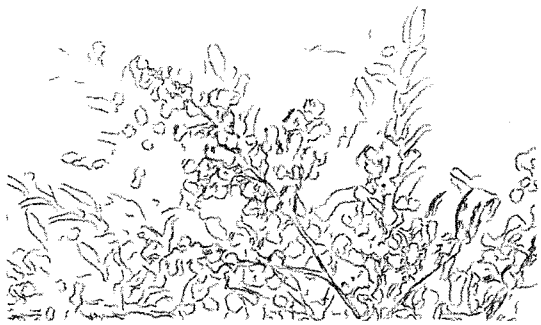
그림 28 마커 라인 드로잉



(a) 입력 영상 (b) 수채 물감 선 텍스쳐 매핑 결과 영상

그림 29 수채 물감 라인 드로잉

(a) 입력영상



(b) 목탄 텍스쳐 매핑 결과 영상

그림 30 목탄 라인 드로잉

(a) 입력영상

(b) 먹선 텍스쳐 매핑 결과 영상

그림 31 먹선 라인 드로잉

8.1 결 론

본 연구는 이차원 영상을 자동으로 원하는 스타일의 라인 드로잉으로 표현해주는 방법을 제시하였다. 이를 위하여 실제 사람이 이차원 영상으로부터 라인 드로잉 하는 과정을 분석하고 그에 따라 알고리즘을 구상하였다. 이는 크게 사람이 어디에 선이 놓일지 판단하는 단계에 해당하는 필터링, 판단된 부분을 가능한한 길고 부드럽게 연결하는 단계에 해당하는 선 연결, 해당 선을 따라 재료의 특성이 살려 실제로 선을 그리는 단계에 해당하는 스타일화로 나뉜다.

연구에서 중요한 초점이 되는 부분은 크게 두 가지이다. 첫째, 라인 드로잉을 위해 특화된 필터링 방법을 제시하였다. 이는 특징선에 해당되는 픽셀을 찾는 단계에서부터 이후 해당 픽셀들이 선으로 연결되어야 한다는 것을 생각하고, 이를 고려하여 정의한 우도 함수를 이용하여 구현되었다.

둘째, 이차원 영상에서 라인 스트로크를 구하여 그 위에 다양한 텍스쳐 매핑을 함으로써 여러 가지 재료의 특성을 살린 라인 드로잉이 가능하게 하였다. 이는 사람이 가능한한 길고 부드럽게 선을 연결한다는 것을 고려, 그에 따른 점들 간의 연결 관계 및 연결 비용을 정의하여 그래프를 생성하고 그 위에서의 그래프 검색을 통해 구현하였다.

본 연구의 결과를 이용하면 삼차원 모델을 입력으로 하던 기존의 방법과는 달리, 이차원 영상을 이용하여 원하는 스타일의 라인 드로잉 결과를 얻을 수 있다.

8.2 향후 연구

본 연구의 향우 연구 과제는 다음과 같다.

• 재료에 따른 다양한 스타일화 방법 연구

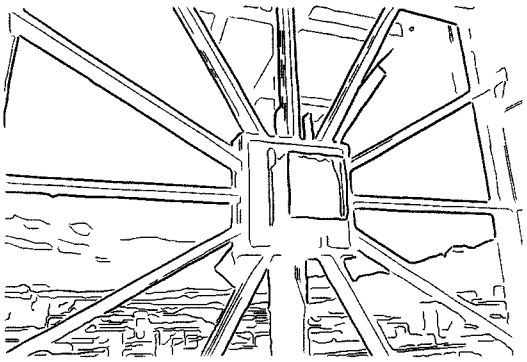
실제 사람이 라인 드로잉을 할 때 나타나는 재료나 선의 특성을 단순하고 일괄적인 텍스쳐 매핑만으로 모두 표현하기에는 무리가 있다. 예를 들어 재료가 붓일 경우 6.3장에서 언급한대로 중요한 선일수록 굵게 그려주는 것이 좋은 효과를 나타낼 수 있지만, 재료가 연필일 경우에는 같은 경우 굵기 변화보다는 선을 진하게 표현해주는 것이 효과적일 것이다. 또한 재료에 따라서 한 번에 선을 완성하기보다는 같은 부분에 조금씩 어긋나는 선을 여러 번 겹쳐 그려 완성하려는 경향을 보이기도 한다. 한편 손에 들어가는 힘에 민감하게 반응하는 재료의 경우, 선의 곡률(curvature)에 따라 하나의 선에서도 선의 굵기가 다양하게 변하는 효과가 난다. 이렇게 재료에 따른 다양한 특성을 파악

(a) 입력 영상

(b) 사인펜 텍스쳐 매핑 결과 영상

그림 32 사인펜 라인 드로잉

(a) 입력 영상



(b) 매직펜 텍스쳐 매핑 결과 영상

그림 33 매직펜 라인 드로잉

하여 각각에 대해 그 특성을 최대화할 수 있도록 특화된 스타일화 방법을 연구할 필요가 있다.

- 복합적 요소를 통합한 상세도 조절

  사용자의 입력 변수를 이용한 상세도 조절에 대해서는 7.1장에서 설명하였다. 하지만 일반적으로 사람이 사용하는 상세도 변화에는 해당 요소들을 비롯하여 선의 밀도가 얼마나 높은지, 선의 길이가 얼마나 긴지 등도 복합적으로 고려된다. 이러한 복합적 요소들을 하나의 변수로 통합하여 상세도 변화를 줄 수 있는 방법에 대한 연구가 필요하다.

- 다수의 영상을 이용한 결과 개선

  동일한 사진을 찍더라도 초점이나 빛의 방향 등의 여러 조건에 따라 결과로 나오는 이차원 영상이 달라진다. 이렇게 환경을 바꿔가며 찍은 사진을 함께 이용하면 렌즈에 의한 왜곡을 보정하고, 물체에 대해 더 많은 정보를 얻을 수 있다. 이러한 정보를 이용하여 결과를 개선할 수 있을 것이다.

- 예술가의 라인 드로잉을 분석하여 이용

  실제 예술가는 보이는 것을 그대로 그리는 것이 아니라 자신만의 기준으로 라인 드로잉을 하고 그것이 개성적인 스타일로 나타난다. 따라서 예술가가 그린 라인 드로잉을 분석, 적용할 수 있는 방법이 필요하다.

- 이차원 영상에 대한 라인 드로잉을 비디오로 확장

  속도 향상 및 시간 일관성(temporal coherence)에 대한 고려를 통해 본 연구의 결과를 비디오로 확장할 수 있다.

참 고 문 헌

[1] Bruce Gooch and Amy Gooch. *Non-Photorealistic Rendering*. A K Peters, 2001.

[2] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-time nonphotorealistic rendering. *ACM Computer Graphics (Proc. SIGGRAPH '97)*, pages 415-420, 1997.

[3] J.D. Northrup and Lee Markosian. Artistic silhouettes: A hybrid approach. *Non-Photorealistic Animation and Rendering (Proc. NPAR 2000)*, pages 31-38, June 2000.

[4] Robert D. Kalnins, Lee Markosian, Barbara J.

Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. Wysiwyg npr: Drawing strokes directly on 3D models. ACM Computer Graphics *(Proc. SIGGRAPH 2002)*, pages 755-762, 2002.

[5] Christian Rössl and Leif Kobbelt. Line-art rendering of 3D-models. In *Proceedings of Pacific Graphics 2000*, pages 87-96, 2000.

[6] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[7] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. ACM Computer Graphics *(Proc. SIGGRAPH '98)*, pages 453-460, 1998.

[8] D. DeCarlo and A. Santella. Stylization and abstraction of photographs. ACM Computer Graphics *(Proc. SIGGRAPH 2002)*, pages 769-776, 2002.

[9] J. Wang, Y. Xu, H.-Y. Shum, and M.F. Cohen. Video tooning. *ACM Computer Graphics (Proc. SIGGRAPH 2004)*, pages 574-583, 2004.

[10] Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. Real-time video abstraction. *ACM Computer Graphics (Proc. SIGGRAPH 2006)*, pages 1221-1226, 2006.

[11] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. *Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 60-65, 2005.

[12] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[13] Oliver Schall, Alexander Belyaev, and Hans-Peter Seidel. Robust filtering of noisy scattered point data. In *IEEE/Eurographics Symposium on Point-Based Graphics*, pages 71-77, 2005.

[14] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In European Conference on *Computer Vision (ECCV'04)*, volume 2, pages 238-249, 2004.

[15] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Computer Graphics (Proc. SIGGRAPH 2004)*, pages 905-914, 2004.

[16] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. ACM Computer Graphics *(Proc. SIGGRAPH '95)*, pages 191- 198, 1995.

[17] Edwin Catmull and Raphael Rom. A class of local interpolating splines. *Computer Aided Geometric Design*, pages 317-326, 1974.

손 민 정

2005년 2월 포항공과대학교 컴퓨터공학과(학사). 2007년 2월 포항공과대학교 컴퓨터공학과(석사). 2007년 3월~현재 포항공과대학교 컴퓨터공학과 박사과정 재학 중

이 승 용

1988년 2월 서울대학교 계산통계학과(학사). 1990년 2월 한국과학기술원 전산학과(석사). 1995년 2월 한국과학기술원 전산학과(박사). 1995년 3월~1996년 9월 미국 City College/CUNY 박사후연구원 2003년 8월~2004년 8월 독일 MPI Informatik 방문선임연구원. 1996년 10월~현재 포항공과대학교 조교수/부교수