이미지 기반 스타일리스틱 렌더링

임훈[1],이종원[2]
세종대학교 컴퓨터 공학부[1 2]
terehun@naver.com[1], jwlee@sejong.ac.kr[2]

Image based Stylistic Rendering

Hun Im[1], Jong Weon Lee [2]
Sejong Univ. [1 2]

요 약

최근 비실사렌더링의 여러 가지 기술들이 발전하고 있다. 특히 회화적 느낌을 지닌 많은 연구들이 이루어지고 있으며, 그로 인해 2 차원과 3 차원 양쪽에서 여러 가지 방법을 통해서 좀더 스타일리스틱하며 아티스틱한 비실사렌더링이 연구되고 있다. 본 논문은 아티스트가 직접 그린 이미지에서 회화적 정보를 얻어와 스타일리스틱한 결과를 내는 새로운 방법을 제시한다.
본 논문에서 제시한 방법으로 이루어진 비실사렌더링의 결과는 단순히 모델이나 사진에 종속적인 색감에서 벗어나 아티스트가 원하는 색과 느낌을 가질 수 있게 된다. 그러므로 아티스트 개인의 예술적 표현을 비실사렌더링에 적용할 수 있게 된다.
Keyword : 비실사렌더링, 스타일리스틱 렌더링

.

1. 서 론[1]

비실사 렌더링(Non-photorealistic rendering)은 짧은 역사에도 불구하고 엔터테인먼트 산업의 새로운 요구에 발맞춰 많은 발전을 하고 있다.

비실사 렌더링은 2 차원 이미지를 사용하는 부분과 지오메트리 모델을 사용하는 부분으로 나뉘어진다. 2 차원 이미지 기반의 경우 일러스트레이션 기법[1][2][3][9]이 있으며, 지오메트리를 사용하는 부분은 에지 디텍팅[5], 셀셰이딩[10], 스타일리스틱 렌더링[4][6], 해칭[7] 등을 포함한다. 본 논문은 지오메트리 기반의 스타일리스틱 렌더링의 새로운 방법을 제시한다.

기존에 연구되어 왔던 방식은 3 차원 폴리곤 모델 경우 폴리곤이 변화하는 방향으로 스트로크의(Stroke) 방향을 정하고, 폴리곤(Polygon)의 크기에 맞춰서 스트로크의 크기를 정하고, 또 정반사와 난반사 성분이 합해진 모델의 색을 기준으로 스트로크를 정했다[6]. 2 차원의 경우 사진에서 직접 이미지 프로세싱을 통해서 스트로크의 방향성 및 크기를 얻어내고 색을 판단하여 렌더링 하는 방식이었다[9]. 그로 인해 소스가 되는 모델이나 사진의 색과 소스의 방향성에 종속된 결과가 나온다. 이런 방향성과 색의 종속성은 결과물이 패턴화가 된다는 문제점을 안고 있으며 아티스트의 직접적인 영향을 미치는 부분이 극히 제한적이다. 그러므로 아티스트가 원하는 방향으로 결과물이 나올 수 없다. 본 논문에서 제안하는 새로운 이미지 기반 스타일리스틱 렌더링의 가장 큰 특징은 아티스트의 그림을 가지고 그것을 분석하여 3 차원 모델에 적용시키는 것이다. 그렇게 함으로써 모델에 종속적인 표현에서 벗어나 아티스트의 예술적 감각을 살린 비실사 렌더링을 구현 해 낼 수 있다.

본 논문에서 제안하는 방식은 다음과 같은 순서로 이루어진다.

* 본 연구는 광주과학기술원 문화연구센터의 지원을 받아 수행되었음.

1. 스트로크 추출 및 메쉬 매칭: 아티스트가 그린 이미지에서 스트로크를 추출한다. 그 후 메쉬 모델의 색과 거리를 이용하여 각각의 스트로크가 메쉬 모델의 어떤 서브 셋에 해당하는지를 알아낸다.
2. 스타일리스틱 렌더링: 추출된 스트로크들을 스타일리스틱하게 부여하기 위하여 분산맵과 노멀맵, Tex ID 맵 등을 구한 후, 스트로크를 매칭시킨다.

본 논문은 2 장에서는 관련 연구를 소개하고 3 장에서는 연구의 전체적인 개관을 살펴본다. 4 장에서는 스트로크의 추출과 메쉬 매칭에 대해서 설명하고 5 장에서는 스타일리스틱 렌더링 방법을 소개한다. 마지막 6 장에서는 실험 결과와 결론으로 이루어져 있다.

2. 관 련 연 구

본 논문에서 사용되는 스트로크는 Haeberli [1]의 개념을 채용하였다. Haeberli's 는 마우스를 이용하여 2D 캔버스 위에서 스트로크를 적용하는 방식이였다. 스트로크의 색은 적용 이미지에서 선택하고, 나머지 방향과 크기는 모두 유저 입력으로 조정 하였다.

Meier[6]는 3 차원 지오메트리 모델(Geometry Model)의 법선 값을 이용하여 스트로크를 제작하고, 드로잉 순서는 깊이 값을 이용하여 아티스틱 렌더링을 하였다. 이 방법은 지오메트리 모델의 깊이 값과 법선 값을 사용하기 때문에 2D 에는 적용되지 않는 방식이다.

Hertzmann [2]는 스플라인 곡선을 이용하여 스트로크를 표현하였다. 이 방법은 몇 개의 레이어에서 이미지를 섞는다. 각각의 레이어는 너비가 동일한 스트로크들이며, 특별한 차이가 있을 때 아래 레이어와 섞게 된다. 이 것을 반복하다 보면 소스 이미지와 결과가 완전히 혼합되어 결과가 나오는 방식이다.

이외에 지오메트리 모델에서는 대부분의 경우 에지[5]를 찾거나, 펜슬을 이용한 일러스트레이션이 대부분이다[4][7].

이러한 기존의 연구들은 대부분 모든 정보를 적용될 이미지에서 추측하게 된다. 이러한 방식들은 좋은 결과를 낼 수 있지만 적용될 이미지나 지오메트리 모델에서 정보를 얻어와 결과를 내기 때문에 전체적으로 일정 패턴을 같게 된다거나, 점차 컴퓨터 그래픽스에서 중요시 되는 아티스트의 느낌을 전혀 넣을 수 없는 단점이 있다. 그러나 제안하는 방식은 아티스트의 그림에서 정보를 얻어와 3 차원 모델에 적용시키기 때문에 위의 단점들을 극복 할 수 있다.

3. 프로그램 개관

본 논문의 방법은 기본적으로 4 가지의 입력 데이터가 필요로 한다. 아티스트가 직접 그린 소스 이미지와 그 소스이미지를 그릴 때 사용한 붓의 터치인 스트로크 이미지가 필요하다. 또 실제 아트 렌더링이 진행되는 지오메트리 데이터가 필요하다. 이 3 차원 모델은 소스 이미지를 기반으로 제작되어야 한다. 마지막으로 필요한 소스는 소스 이미지에서의 하이라이트 위치가 필요 하다.

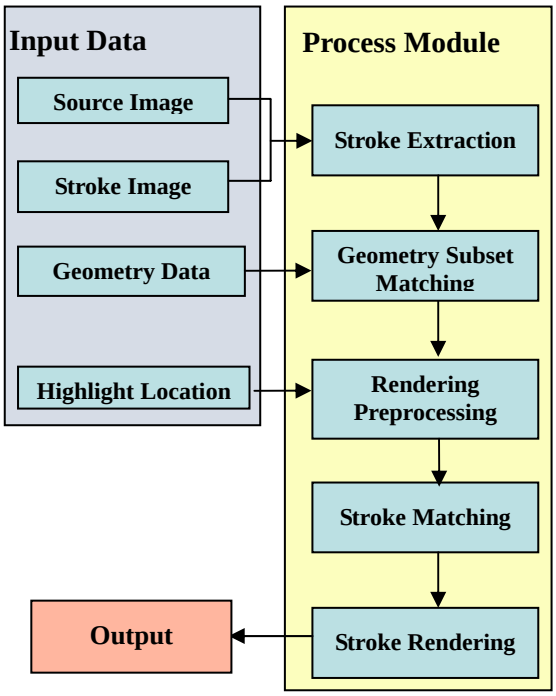

그림 1 프로그램 레이아웃

제안된 방법은 그림 1 과 같이 프로세스가 진행된다. 먼저 소스 이미지를 로드 하여 스트로크

를 추출한다. 스트로크는 사용자 입력 값 ***S*** 를 받아 ***S***×***S*** 값으로 자른 이미지로 스트로크를 추출한다. 이 이미지를 타겟 추출 이미지라 한다. 타겟 추출 이미지에서 유사 칼라 값을 이용하여 스트로크 데이터를 추출하며 추출된 스트로크 데이터를 이용하여 스트로크를 미리 텍스쳐에 렌더링 한다.

그 후 지오메트리 데이터를 로드하여 폴리곤의 난반사(Diffuse) 값과 스트로크의 칼라 값, 그리고 2D 이미지에서의 스트로크 위치와 지오메트리 데이터에서의 스트로크 위치를 계산하여 지오메트리 모델 안에 서브셋 중 어떤 부분에 스트로크가 매칭되는지 검사한다.

이로써 스트로크의 기본적인 데이터 및 정보는 모두 저장되었고 스타일리스틱 렌더링을 하게 된다.

스타일리스틱 렌더링은 먼저 렌더링 전처리(Preprocessing)에서 시작된다. 우선 렌더링을 위해서 깊이와 법선 값을 가지고 있는 법선-깊이맵(Normal-Depth Map)과 그것을 기반으로 한 분산맵(Distribute Map), 하이라이트를 위한 하이라이트 맵(Highlight Map), 서브셋 구분을 위한 TexID 맵을 만들게 된다. 만들어진 각각의 맵들은 스트로크의 드로잉 위치 매칭에 사용되며, 매칭된 결과를 스트로크 드로잉에 적용시킨다.

4. 스트로크 추출 및 지오메트리 데이터 매칭

4.1. 추출 타겟 이미지 분석

입력 이미지인 아티스트가 그린 이미지가 들어오면 우선 바탕 부분을 범람 채우기(Flood Fill) 기법으로 구분해 낸다. 단순하게 RGB(255, 255, 255)만을 채우는 것이 아니라 색의 평균값이 250 이상인 경우를 모두 채우게 된다. 그렇지 않으면 그림 주변의 흰 부분이 남게 된다. 그 후 배경을 제외한 전경만을 사용자 지정한 크기인 *S* 크기만큼의 사각형으로 이미지를 자르고 저장한다. 그 이미지를 추출 타겟 이미지라고 부른다.

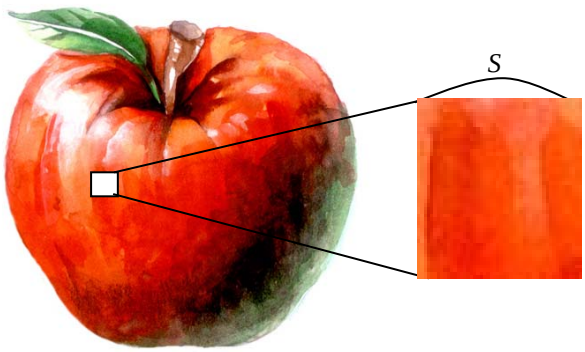

그림 2 소스 이미지와 추출 타겟 이미지

추출 타겟 이미지를 만드는 이유는 소스 이미지 전체에서 칼라 값을 이용하여 스트로크를 추출하면 스트로크가 너무 크게 되는데, 이것은 아티스가 그리는 것과 같은 방식이 아니다. 아티스트는 여러 번의 터치로 그렸을 이미지가 하나의 스트로크로 대체 되기 때문이다.

추출 타겟 이미지의 분석은 아래와 같은 방법으로 진행된다.

1. 타겟 이미지를 픽셀 검색한다.
2. 검색된 픽셀은 이미 추가된 스트로크 픽셀 데이터 중 수식 1 의 조건에 만족 하는지 알아낸다.
3. 조건에 만족하면 만족된 스트로크 픽셀 데이터에 추가 된다. 만족하지 않으면 새로운 스트로크 픽셀 데이터가 추가된다.

수식 1 에서 $Target_{rgb}$ 는 검색이 진행되는 픽셀의 RGB 값이다. $Data\_i_{rgb}$ 는 스트로크 픽셀 데이터 셋을 말한다.

$$Diff_{rgb} = Target_{rgb} - Data\_i_{rgb}$$
$$MinDiff = \min(Diff_b, \min(Diff_r, Diff_g))$$
$$MaxDiff = \max(Diff_b, \max(Diff_r, Diff_g))$$
$$|MinDiff| + |MaxDiff| < T\_COLOR$$

Target – 검색된 픽셀, Data 스트로크 픽셀 데이터, ***T_COLOR*** 차이 구분값

수식 1 스트로크 픽셀 구분

T_COLOR 가 20 이하일 경우는 스트로크로 생기는 블러링이 추가되지 않고, 40 이상인 경우는 다른 색조를 같은 스트로크로 추가하는 현상이 생

겼기 때문에 본 논문은 ***T_COLOR.***는 30 으로 적용하였다.

4.2. 스트로크 생성

이렇게 추출된 스트로크 픽셀 데이터를 이용하여 스트로크 데이터[1]인 크기, 각도, 색, 위치를 얻어오게 된다. 각 데이터를 얻어오는 방식은 이미지 모멘트(Image Moment)[8]를 이용하여 되는데, 그 방법은 Michio[9] 방식을 변형한 것이다.

그림 3 (a) - 스트로크 픽셀 데이터 값 (b) - 이미지 모멘트 식을 이용하여 구한 대응 사각형

각각의 데이터는 수식 2 로 얻어온다.

$$M_{lm} = \sum_{y}\sum_{y} x^{l} y^{m}$$

$$x_c = \frac{M_{10}}{M_{00}}$$

$$x_y = \frac{M_{01}}{M_{00}}$$

$$\theta = \frac{\tan^{-1}\left(\frac{b}{a-c}\right)}{2}$$

$$w = \sqrt{6\left(a+c-\sqrt{b^2+(a-c)^2}\right)}$$

$$l = \sqrt{6\left(a+c+\sqrt{b^2+(a-c)^2}\right)}$$

수식 2 스트로크 데이터 획득 식

a, b, c 는 수식 3 과 같이 정의 된다.

$$a = \frac{M_{20}}{M_{00}} - x_c^{\,2}$$

$$b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)$$

$$c = \frac{M_{02}}{M_{00}} - y_c^{\,2}$$

수식 3 스트로크 데이터 획득시 변수

수식 2, 3 에서 $n = l + m$ 이라 할 때 M_{lm} 은 n 차원의 이미지 모먼트라[]고 불려진다. 여기서 x_c , y_c 는 스트로크의 중점이며 이를 ***C*** 라 한다. θ 는 회전각, ***w*** 는 너비 ***l*** 은 길이가 된다. 이 정보들이 들어있는 스트로크 데이터를 ***SD*** 라 한다.

$l \times w$ 값에 전체 스트로크 픽셀 데이터 숫자를 나눠 준다. 이것은 스트로크의 밀집도이며 이 밀집도로 스트로크의 블렌딩 값을 정한다.

이렇게 생성된 정보로 스트로크를 텍스쳐화 한다. 이때 스트로크는 색의 휘도에 따라 레벨링을 한다. 레벨링은 스트로크를 부여할 때 밝은 것부터 드로잉을 하는 아티스트들의 기본적인 드로잉 방식을 적용하기 위해서이다.

4.3. 지오메트리 서브셋 메칭

지오메트리 서브셋 메칭은 지오메트리 모델을 로드 한 후에 전혀 다른 색을 가지거나 너무 먼 곳에 있는 서브셋에 적용되지 않게 하기 위해서이다.

여기서 서브셋이란 지오메트리 모델에서 매터리얼(Material) 값이 다른 폴리곤 덩어리를 말한다.

지오메트리 서브셋과 스트로크 매칭은 아래와 같이 이뤄진다.

1. 버텍스 단위로 되어있는 3 차원 좌표계와 픽셀 단위 2 차원 좌표계의 차이를 스케일링을 통하여 맞춰준다.
2. 각 서브셋의 충돌 사각형(Bound Box)을 구한다.
3. 외각 사각형과 스트로크와 거리를 잰다.
4. 두 개 이상의 충돌 사각형 안쪽에 있거나, 어

떤 외각 사각형의 안쪽에 포함되지 않는 경우 지오메트리의 난반사와 스토로크의 칼라의 차이를 이용하여 스크로크가 어떤 서브셋에 포함되는지 결정한다.

5. 스타일리스틱 렌더링

스타일리스틱 렌더링을 위해서는 스트로크들이 적당한 위치에서 드로잉 되어야 한다. 그를 위해 스트로크의 위치를 결정하는 분산맵을 만들고 분산맵에 스트로크들을 매칭시켜야 한다. 매칭을 위해 2 차원과 3 차원의 스트로크 위치에서 하이라이트 방향 벡터를 비교하게 된다.

5.1. 3 차원 모델 전처리

3 차원 모델의 전처리 과정은 일반적인 Transports & Light(T&L) 과정이 아닌 방식으로 렌더링된다. 가장 먼저 렌더링되는 것은 법선-깊이맵이다. 법선 깊이 맵은 r,g,b 값에 법선 값을 저장하고 a 값에 깊이 값을 저장하여 나온 결과이다. 이 결과를 가지고 분산맵을 만들게 된다.

분산맵이란 어떻게 스트로크가 분산되느냐를 결정하는 중요한 맵이다. 이것은 깊이 값의 변화량과 법선 백터의 변화량을 가지고 결정하게 된다. 아티스트가 드로잉을 할 때 복잡한 부분에 더 많은 터치를 하게 되므로, 법선-깊이 값의 변화량이 큰 부분에 더 많은 스트로크가 들어가야 한다. 이를 위해 아래와 같은 방식으로 법선-깊이맵에서 분산맵을 구하게 된다.

1. 3×3 필터로 법선값을 각각 내적한다.
2. 만약 필터 중 하나의 두 노멀값의 차이 내적을 ***Dot*** 이라 할 때 ***(n − AddNormal)*** > ***Dot*** 이면 스트로크 위치로 확정한다.
3. 법선 벡터의 사이각으로 인해 스트로크 위치로 확정되지 않은 경우 각각의 내적값에서 평균을 내어 ***AddNormal*** 에 대입한다.
4. 깊이 값의 검사는 3×3 필터에서 차이의 합을 ***D*** 라 할 때 ***AddDepth+D>m*** 이면 스트로크 위치로 확정한다.
5. 깊이 값의 차이에 의해 위치로 확정되지 않은 경우 현재 차이 값을 ***AddDepth*** 에 저장한다.

우리는 ***n*** 값에 0.9 를 ***m*** 값에 0.5 를 부여 하였다. 이렇게 나온 분산맵은 그림 4 와 같다.

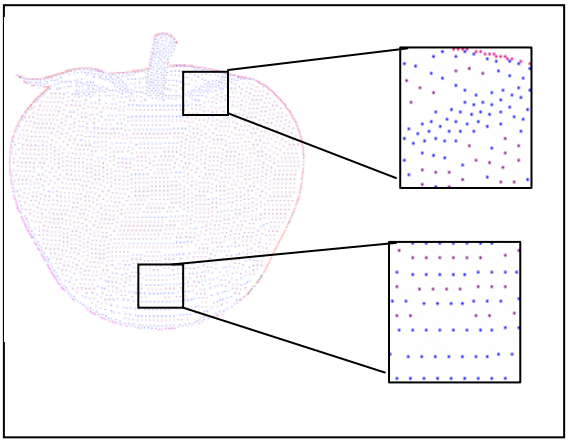

그림 4 분산맵

그림 4 는 깊이 값으로 인해 스트로크 위치가 결정된 경우 r 값에 저장하고, 법선의 사이각으로 결정된 스크로크의 위치의 경우 b 값에 저장 한 결과이다. 이렇게 결정된 스트로크의 위치를 ***T*** 라고 한다.

분산맵을 구한 후 매칭에 필요한 맵들을 더 만든다. 각각의 맵은 그림 5 에 나와있다.

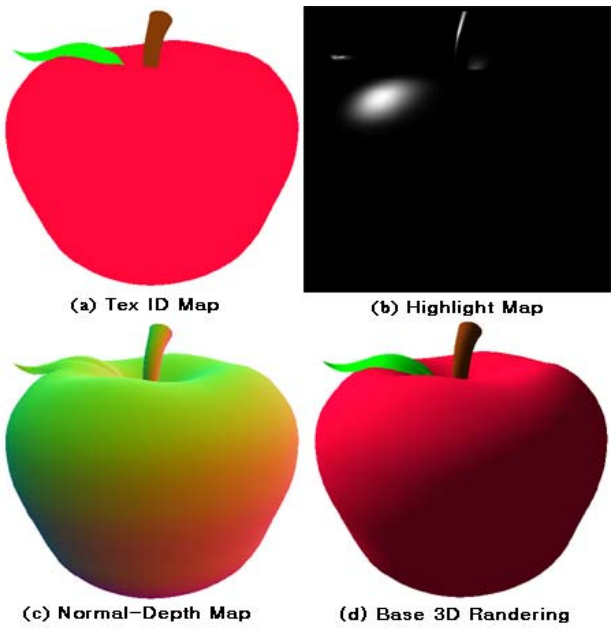

그림 5 각 종류의 맵들 (d) 는 3D 렌더링 결과

Tex ID 맵은 난반사 성분에 전혀 조명을 입히지 않은 지오메트리 값 그 자체이다. 이 맵은 4.3 지오메트리 서브셋 매칭에서 매칭된 스트로크들이

다른 서브 셋을 침범하지 못하도록 만든다.

하이라이트 맵은 Phong Shading 결과로 인해 나온 맵이다. 이 맵은 하이라이트 매칭과 하이라이트 부여에 사용된다. 하이라이트 맵 중 0 이 아닌 부분들을 범람 채우기로 채워 그 중심을 모델 하이라이트로 정한다.

노멀-깊이맵은 분산맵을 만드는데 이외에 분산맵에서 확정된 스트로크 위치에 최저 휘도를 구하는 데도 사용된다. (Base 3D Rendering 에 관하여 추가하세요.)

5.2. 스트로크 위치 매칭 및 페인팅

분산맵의 각 점으로 표현된 ***T*** 에 알맞은 스트로크를 매칭 시키는 과정이 필요하다. 이때 3 차원 모델과 2D 이미지가 완전히 같을 수 없고, 카메라의 조정 등으로 결과가 달라질 경우를 대비하여 선 매칭을 하게 된다.

1. 유저에 의해 선택된 이미지에서의 하이라이트에서 ***C***(스트로크의 중점)방향으로의 벡터를 구한다. 이 벡터를 ***IV*** 로 정의한다.
2. 하이라이트 맵에서 구한 3 차원 모델 하이라이트 위치와 ***T*** 으로의 벡터를 구한다. 이 벡터를 ***MV*** 로 정의한다.
3. ***MV*** 와 ***IV*** 사이 각 θ 로 $w = \cos\theta$ 를 구한다.
4. ***MV*** 와 ***IV*** 벡터의 길이 ***l*** 을 구한다.
5. ***w*** 의 값이 0.939(-20~20 도) 이하 이면서 ***l*** 의 길이가 가장 작거나 ***l*** 이 **S** 값 이하이면서 ***w*** 의 값이 가장 큰 것을 스트로크로 정한다. 만약 그런 값이 같은 레벨에 여러 개라면 수식 4 번의 ***match*** 값이 가장 작은 것으로 선택한다.
6. 만약 어떤 ***T*** 에도 모든 ***C*** 가 만족하지 못한다면 2 번에서 구해진 백터의 반대 방향으로 매칭되는 ***C*** 를 찾아본다.
7. 6 번 프로세스로도 매칭이 되지 않는 ***T*** 는 가장 근처에 있는 ***T*** 의 매칭 ***C*** 값을 할당 받는다.

$$match = w \times 0.5 + \left(1.0 - \frac{l}{S}\right) \times 0.5$$

수식 4 매칭 웨이팅 값

전처리 매칭 후의 프로세스는 매칭 – 드로잉을 한 세트로 각각의 레벨 별로 이루어 지게 된다.

매칭 프로세스 에서도 위의 1-7 번 까지의 프로세스를 거의 유사하게 따라간다. 각 스트로크 레벨 별로 드로잉 하기 때문에 현재 레벨에 맞는 ***SD***(스트로크의 상세 정보가 들어있는 데이터) 값에서만 1-7 번의 프로세스를 적용시킨다.

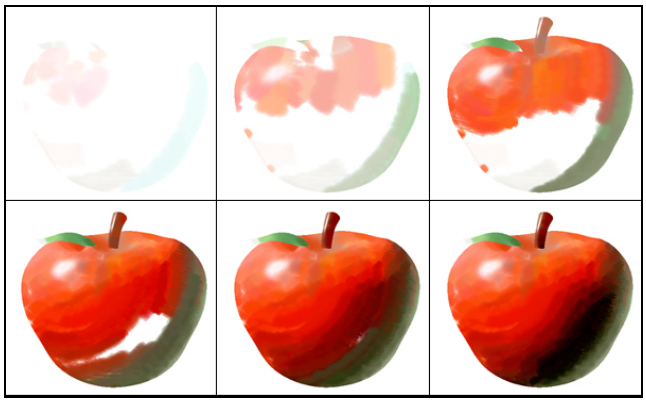

그림 6 레벨 별 스트로크. 왼쪽 위부터 0, 3, 7, 10, 12, 14 레벨

그림 6 은 각 0 레벨부터 14 레벨 까지의 드로잉 진행을 보여주고 있다.

스트로크 드로잉은 각 스트로크의 블렌딩 값에 의해서 합쳐 지는 방식이다. 단, 하이라이트 맵을 참조 현재 위치에 하이라이트 값이 있다면 블렌딩 값이 하이라이트 값만큼 감소된다.

6. 결 론

6.1. 실험 결과

그림 7 입력 스트로크 이미지

그림 7 은 소스 이미지를 그린 스트로크로써 입력 스트로크 이미지로 스트로크 텍스쳐를 만들

어 낸다.

그림 8 소스 이미지

그림 8 은 위에 스트로크로 아티스트가 그린 이미지 이다.

그림 9 추출된 스트로크 이미지

그림 9 는 그림 8 에서 추출된 스트로크 중 일부이다. 왼쪽은 스트로크의 색을 표현한 것이며, 오른쪽은 투명 값을 표현한 알파맵이다. 그림 8 에서 총 스트로크는 637 개가 추출되었다.

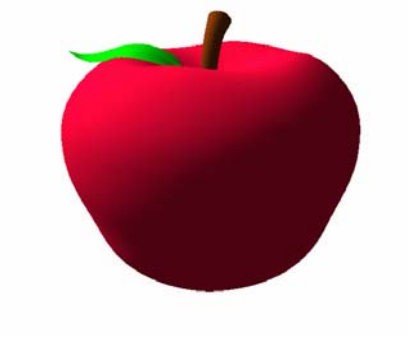

그림 10 소스 지오메트리 모델

그림 10 은 소스 이미지를 바탕으로 만든 지오메트리 모델이다. 이 지오메트리 모델을 이용하여 스타일리쉬 렌더링을 한다.

그림 11 완성 이미지

그림 11 은 완성 이미지이다. S = 25 값으로 설정하였다. 그림 8 의 아티스트가 그린 소스 이미지를 기반으로 가장 어두운 부분에 역광 처리가 검은 녹색으로 처리 되었으며, 명함 단계도 유사하게 표현되었다.

6.2. 결론 및 향후 과제

본 논문에서 제안한 이미지 기반 스타일리스틱 렌더링은 기존에 방법들과 다르게 패턴화되거나 아티스트의 느낌을 줄 수 없는 것에서 벗어나 아티스트의 그림을 기반으로 색감, 붓터치, 명함단계 등을 추출하여 3 차원 지오메트리에 아티스트의 느낌을 부여 할 수 있게되었다.

좀더 좋은 결과를 위해서 아래와 같은 점이 더 보완 되어야 한다.

1. 아티스트가 독특한 방식의 하이라이트 처리를 하였을 경우 그것을 가지고 오지 못한다. 현재 일반적인 방법으로 하이라이트를 합성하기 때문에, 하이라이트가 독특한 방식으로 그려졌다면 결과가 유사 하지 않다.
2. 본 논문에서 기정 사실화 한 두 가지 사실인 '복잡한 곳에 더 많은 터치가 필요하다'와 '밝은 부분부터 먼저 터치를 한다' 가 아닌 그림 역시 아티스트의 느낌을 그대로 따올 수 있어야 한다.
3. 복잡한 그림과 모델의 경우 그림과 모델에 맞춰 새로운 서브셋을 만들어 내야 한다. 그래야 스트로크들이 하나의 서브셋에 너무 많이

몰려 매칭시 에러가 나는 현상을 줄일 수 있다.

8. 참고 문헌

[1] Paul Haeberli. Paint by numbers: Abstract image representations. *Computer Graphics*, 24(4):207–214, August 1990.

[2] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In Michael F. Cohen, editor, *SIGGRAPH98 Conference Proceedings*, Annual Conference Series, pages 453–460, July 1998.

[3] William Baxter, Jeremy Wendt, and Ming C. Lin. IMPaSTo: A Realistic, Interactive Model for Paint. *SIGGRAPH2004.* June 2004.

[4] Brett Wilson, Kwan-Liu Ma. Rendering Complexity in Computer-Generated Pen-and-Ink Illustrations. *SIGGRAPH2004.* June 2004.

[5] J.D. Northrup, Lee Markosian. Artistic Silhouettes: A Hybrid Approach. *Proceedings of NPAR 2000,* June 2000

[6] Barbara J. Meier. Painterly rendering for animation. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 477–484, August 1996.

[7] Pierre-Marc, Jodoin Emric, Epstein Martin, Granger-Pich´e, Victor Ostromoukhov. Hatching by Example: a Statistical Approach. *SIGGRAPH 2002,* June 2002

[8] Berthold K. P Horn. *Robot Vision*. MIT Press, Cambridge 1986, 1986.

[9] Michio Shiraishi, Yasushi Yamaguchi. An Algorithm For Automatic Painterly Rendering Based On Local Source Image Approximation. . *SIGGRAPH2000.* June 2000

[10] Adam Lake, Carl Marshall, Mark Harris, Marc Blackstein. Stylized Rendering Techniques For Scalable Real-Time 3D Animation. *SIGGRAPH2000.* June 2000