심 카빙 알고리즘을 이용한 특정 객체 이미지 제거 프로그램

최희수, 이강만
동국대학교 멀티미디어공학과
etheesu@dongguk.edu, gangman@mme.dongguk.edu

A Specific Object Image Removal Program using Seam Carving algorithm

Hee-Su Choi, Gangman Yi
Dept of Multimedia Engineering, Dongguk University, Seoul, 04620, Korea

요 약

이미지의 특정 객체를 제거할 때, 주변 환경을 고려하면서 제거하기에 어려움이 있다. 본 연구는 특정 객체가 제거되면서 생기는 빈자리를 자연스럽게 보완하기 위해서 이미지 내용을 기반으로 이미지를 변경하는 Seam Carving 알고리즘을 이용하여 보다 자연스러운 결과 이미지를 생성하는 프로그램을 구현했다.

1. 서론

기존의 이미지에서 이미지의 크기를 효과적으로 변경하기 위해서는 기하학적인 구속조건을 사용할 뿐만 아니라 이미지의 내용도 고려해야 한다. 이미지의 일부 영역을 포기하고 잘라내는 자르기(cropping)방식이나, 크기 조절(resizing)방식을 이용해 이미지의 전체를 모두 포함시키지만 비율에 맞춰서 이미지를 확대/축소하기 때문에 비율이 변경 될 경우 이미지가 왜곡되어 보이는 방식으로는 이미지의 크기를 효과적으로 변경하기 어렵다. 이런 문제를 해결하기 위해서 이미지 내부 내용을 기반으로 이미지를 변경하는 심 카빙(Seam Carving) 알고리즘이 있다.

심 카빙(Seam Carving) 알고리즘은 이미지에서 비슷한 부분에 대해 근접한 부분이 연속적으로 이어지는 경로는 그렇지 않은 경로에 비해 정보가 적을 것이라는, 중요도가 떨어질 가능성이 높다는 가정을 바탕으로 이미지를 변경하는 방식이다. 이를 이용하면 기존의 이미지 변경 방식보다 중요한 정보를 더 많이 이미지에 담을 수 있다. 하지만 심 카빙(Seam Carving) 알고리즘에서 정의한 중요도에 따른 정보, 에너지(energy)가 우리가 원하는 이미지의 중요도와 일치하지 않을 경우도 있다. 이러한 경우 어떤 부분을 중요한 내용이 있는 부분으로 선택할지에 대한 정확한 솔루션이 제안되지 않았다. 따라서 사용자에 따라 이미지에서 중요도를 판별하는 과정이 필요하다.

본 논문에서 위의 문제를 해결하고자 사용자로 하여금 중요하거나 중요하지 않은 부분에 대해서 직접 표시 할 수 있게 GUI를 제공하며 그에 따라 이미지를 변경할 때 원하는 특정 객체 이미지를 보존 또는 제거할 수 있는 방법을 제안하며 이 방법은 이미지 크기를 변경하지 않고 특정 객체만 자연스럽게 제거할 수 있게 심 카빙(Seam Carving) 기법을 알고리즘을 사용하여 구현하였다.

2. 기능

2.1 Seam Removal

심 카빙(Seam Carving) 알고리즘의 기본인 심 삭제(seam removal)는 중요도가 낮은 심을 찾아서 우선적으로 삭제하는 방식으로, 그 과정은 다음과 같다.

첫째, 기존 이미지에서 어떤 pixel이 중요도가 높고 낮은지를 결정해야 한다. 먼저 이미지의 에너지 지도(energy map)를 생성함으로써 이를 표현하는데 Seam Carving에서는 다음과 같은 에너지 공식을 사용한다.[1]

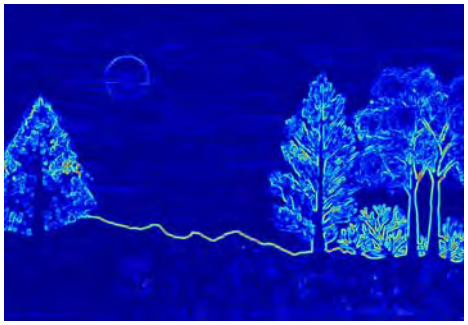
$$e(I) = \left|\frac{\partial}{\partial x}I\right| + \left|\frac{\partial}{\partial y}I\right|$$

<수식 1> Energy Formula

위의 에너지 공식을 이용하면 pixel 값이 많이 변하는 영역, 즉 더 많은 정보를 포함하는 영역을 측정할 수 있다. 위 공식은 pixel의 에너지를 측정하는 방법 중 하나로 각 pixel을 주변 pixel과 대조하여 그 차이를 확인할 수 있다. 본 논문에서는 Scharr Filter를 적용하여 이미지의 x방향과 y방향에서의 기울기(gradient)의 절대 값을 합산하여 에너지 값을 계산한다. 또한 이미지의 pixel은 세 개의 채널(B,G,R)을 가지고 있기 때문에 이를 분리해서 계산한 후 합산하여 에너지 지도(energy map)를 만든다. <그림

2>에서 확인할 수 있듯이 에너지 지도(energy map)는 <그림 1>의 입력 이미지와 같은 차원의 2D 이미지로 표현된다.

<그림 1> 입력 이미지(1)



<그림 2> 에너지 지도(energy map)

둘째, 누적 비용 행렬(cumulative cost matrix)을 만든다. 누적 비용 행렬은 에너지 지도(energy map)의 상위 가장자리에서 시작하여 행을 반복하며 구성한다. 누적 비용 행렬의 pixel 값은 누적 비용 행렬에서 세 개의 상단 이웃 pixel(상단 왼쪽, 상단 중앙, 상단 오른쪽)의 최솟값에 에너지 지도의 해당 pixel 값을 더한 것과 동일하다. 누적 비용 매트릭스의 맨 위 행의 값은 에너지 지도(energy map)와 동일하다. 왼쪽이나 오른쪽 가장자리 때문에 인접 pixel을 사용할 수 없는 경우, 해당 인접 pixel은 계산에 사용하지 않는다. 위 과정을 <수식 2>와 같은 공식으로 표현한다.[1]

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

<수식 2> Cumulative Energy Formula

위의 방식은 결과적으로 인접한 pixel 중 가장 작은 에너지 값을 가지고 있는 pixel의 위치를 경로에 추가하여 전체 이미지를 아래로 통과하는 수직 심(vertical seam)을 만든다. 이 단계에서 동적 프로그래밍(dynamic programming)을 이용하여 구현한다.

셋째, 하단 행에서 상단 행으로 역추적(backtracking)을 통하여 <그림 4>와 같은 최적의 심(optimal seam)을 찾는다. 누적 비용 행렬의 하단 줄에서 최솟값 pixel을 찾고 누적 비용 행렬의 상단 열까지 추적하여 최소 심 좌표를 저장한다. 저장된 심 좌표를 가지고 해당 pixel을 제거한 후 각 행의 모든 pixel 중 최소 심보다 큰 인덱스를 가지고 있는 경우 왼쪽으로 이동한다. 이미지의 너비가 정확히 1 pixel이 감소한다.

넷째, 이 과정을 원하는 너비가 될 때까지 반복한다. 이 과정이 끝나면 덜 중요한 내용이 우선적으로 삭제되는 심 삭제 결과를 얻을 수 있다. 너비가 아닌 높이를 줄일 경우, 이미지를 90° 회전시킨 후 위 과정을 반복한다.

<그림 5>에서 확인할 수 있듯이 심 삭제 결과 이미지는 입력 이미지<그림 3>과 비교해서 내용적인 부분은 비율이 줄어들지도 삭제되지도 않았다.

<그림 3> 입력 이미지(2)

<그림4> 최적의 심(optimal seam)

<그림 5> Seam Removal을 진행한 이미지

2.2 Seam Insertion

심 삽입(Seam Insertion)은 심 삭제와 과정이 매우 유사하다. 복제된 입력 이미지를 만들어 심 삭제를 수행할 때와 같이 최적의 심(optimal seam)을 찾고 제거한다. 제거할 때, 이와 동일한 순서로 제거되는 pixel의 모든 좌표를 저장 후 기록된 좌표에 새로운 심을 삽입한다. 새로 삽입된 pixel은 주변 이웃 pixel로부터 받은 평균값을 입력한다. 이 과정을 원하는 너비가 될 때까지 반복한다.

2.3 Object Removal

객체 제거(Object Removal)는 자신이 지정해놓은 객체를 우선적으로 삭제할 심으로 만들 수 있다. 이를 위해서는 사용자가 제거할 부분을 표시한 마스크(mask)가 필요하다. 에너지 지도를 생성할 때 마스크로 보호되는 지역은 매우 높은 음의 값으로 가중치를 부여한다. 높은 음의 가중치를 가지고 있는 마킹(marking)된 부분은 최적의 심을 찾을 때 우선적으로 포함된다. 최적의 심을 삭제하는 과정의 심 삭제와 동일하다. 마스크에 표시된 부분이 모두 삭제될 때까지 이 과정을 반복한다. 반복할 때마다 정확한 에너지 지도를 얻기 위해서는 심 제거 단계에서 최적의 심을 제거할 때 좌표를 저장하여 마스크에서도 동일하게 삭제한다.

객체를 제거하기 위해서 심 삭제를 진행한 만큼 이미지의 너비는 줄어든다. 줄어든 이미지의 너비를 다시 입력 이미지의 너비의 크기로 만들어 주기 위해 심 삽입을 수행한다. <그림 6>의 입력 이미지와 <그림 7>의 마스크 이미지를 가지고 <그림 8>의 Object Removal이 수행된 이미지를 얻을 수 있다.

<그림 6> 입력 이미지(3)

<그림 7> 마스크(Mask) 이미지

<그림 8> Object Removal을 진행한 이미지

2.4 Object Preservation

객체 보존(Object Preservation)은 자신이 지정해놓은 객체를 심 삭제에 포함시키기 않는다. 사용자가 보존시키고 싶은 객체를 표시한 부분을 마스크로 만든다. 에너지 지도를 생성할 때 마스크로 보호되는 지역은 매우 높은 양의 가중치를 부여한다. 높은 양의 가중치를 가지는 부분은 최적의 심을 찾을 때 포함되지 않는다.

3. 구현

본 논문의 프로그램은 기본 구성은 <표 1>이다. 심 카빙(Seam Carving) 알고리즘의 기능은 OpenCV를 이용하여 구현했다. GUI 부분은 Python 표준 GUI 인터페이스인 tkinter를 사용해서 사용자가 이미지에 직접적으로 입력을 할 수 있게 구현했다. 출력 결과로는 입력 이미지와 결과 이미지를 정확하게 비교할 수 있게 Matplotlib를 통해 시각화 기능을 제공한다.

본 논문의 프로그램은 위에 기술한 4가지 기능들을 사용할 수 있다. 객체 제거(Object Removal)와 객체 보존(Object Preservation)은 <그림 9>와 같이 입력 이미지 위에 사용자가 직접 객체에 대한 영역을 정하고 그에 따른 결과 값 <그림 10>을 확인할 수 있다.

구 분	사 양
운영시스템	Windows 10 Education / 64bit
통합개발환경	Pycharm
개발언어	Python
라이브러리	Open Source Computer Vision Library (OpenCV), tkinter, Matplotlib

<표 1> 입력 이미지(3)

<그림 9> GUI를 통해 객체 영역 설정

<그림 10> 출력 결과

4. 결론

본 논문은 이미지의 객체를 삭제할 때 이미지의 내용을 기반으로 처리하는 Seam Carving 알고리즘을 사용하여 이미지의 왜곡을 최소화하면서 사용자가 원하는 객체를 삭제할 수 있는 GUI환경을 제공한다. 자신이 원하는 이미지를 입력하면 그에 맞는 처리 후 처리된 결과 이미지를 얻을 수 있게 구현했다.

Acknowledgments

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학지원사업의 연구결과로 수행되었음"(2016-0-00017)

참고문헌

[1] Shai Avidan , Ariel Shamir, (2007) Seam carving for content-aware image resizing, ACM Transactions on Graphics (TOG), v.26 n.3, July 2007

참고이미지

(1) Acrylic Painting Landscape
https://vidalcuglietta.com/acrylic-painting-landscape/12386/acrylic-painting-landscape-new-easy-landscape-paintings-for-beginners-step-by-step/
(2) Photo by Brent Cox on Unsplash
https://unsplash.com/photos/ydGRmobx5jA
(3) The author is Newton2
File:Broadway tower edit.jpg