2018 VCLab 동계 미디어 융합 연구

Researcher 한 창 헌* 정 승 찬† 김 예 진‡ 이 주 석§ Professor 신 현 준¶

AJOU UNIVERSITY

Department of Digital Media, Ajou University

*ehwjs1914@ajou.ac.kr, †tmdcks6628@ajou.ac.kr, ‡dpwls273@ajou.ac.kr, §dljoo@ajou.ac.kr, ¶joony@ajou.ac.kr

digitalMEDIA

Introduction

2학기를 마치고, 동계 방학 동안 연구실에서 미디어 융합 연구를 할 수 있는 좋은 기회를 얻게 되었다. 연구의 첫 주에는 각자 구현해보고 싶은 논문을 골랐고, 두 번째 주에 세미나를 진행하여서 논문에 대한 이해를 높였다. 그 이후에는 논문을 기반으로 논문에서 제안한 방법들을 구현했다. 논문 구현은 약 두 달 동안 진행되었으며, 구현 내용을 지금부터 간략하게 설명하고자 한다.

GrabCut[1]

1.1. Abstract

본 논문에서는 사진에 사각형과 불완전한 지시선을 주었을 때, 전경과 배경을 구분해주는 방법을 제안했다. 이 과정에서 사진의 각 픽셀이 노드인 그래프를 만들고, Max-flow min-cut 알고리즘을 반복적으로 적용해서 전경과 배경을 분리한다. 그리고 마스크에서 경계 주변의 영역에 해당하는 부분에 대하여 투명도가 연속적인 값을 가지기 위해서 Border matting 방법을 적용하였다.

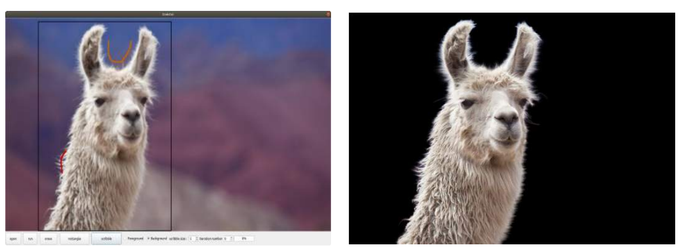


그림 1. GrabCut에서는 사용자의 불완전한 지시선으로 경계를 추출한다.

1.2. GrabCut

사진에서 전경과 배경을 분리하기 위해 기존의 방식인 Graph-Cut을 반복적으로 사용했다. Graph-Cut은 사진의 각 픽셀을 그래프의 한 노드로 하고, Source 노드와 Sink 노드가 있는 그래프에서 에너지 함수를 정의하고, Max-flow Min-cut 알고리즘이 적용하는 방식이다. Graph-Cut에서는 Data term과 Smoothness term의 합으로 정의되는 에너지 함수가 있고, 이를 최소화하는 Cut을 찾아서 전경과 배경을 분할한다. 흑백 이미지를 다룬 기존의 Graph-Cut을 컬러 이미지에 적용하기 위해, Gaussian Mixture Model(이하 GMM)을 사용하였고, 기존의 Data term을 수정하였다. Data term을 통해서 비슷한 색상이 일관되게 전경 혹은 배경으로 할당되게 할 수 있었다. Smoothness term을 통해서 이웃하는 픽셀 간의 픽셀값의 차이가 클 때 다른 영역으로 구분할 수 있었다. 그리고 입력받은 사각형과 불완전한 지시선을 토대로 K-Means를 사용하여 배경과 전경의 5개의 Full-Covariance를 갖는 GMM을 생성하고 주어진 에너지 함수를 최소화하기 위해 Graph-Cut을 반복적으로 함으로써 GMM을 학습한다.

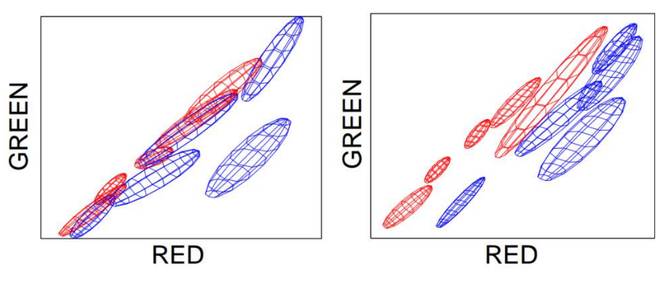
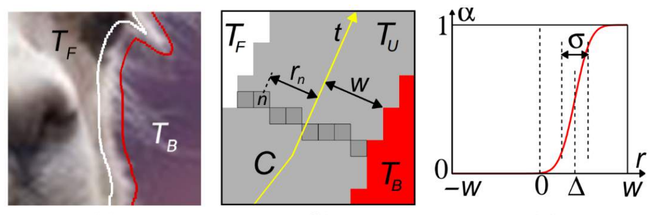


그림 2. Gaussian Mixture Model이 반복적인 Graph-Cut을 통해서 학습된다.

1.3. Border Matting

GrabCut을 통해서 얻은 마스크 이미지에서 깊이 우선 탐색을 통해 이어지는 경계를 얻는다. 경계 부분의 ±6의 영역만큼을 T_U라고 한다. 경계의 픽셀에서의 L * L 크기(L = 41)의 사각 영역 L을 정의하고 T_U안쪽의 전경 영역과의 교집합인 F_t와 배경 영역과의 교집합 F_b의 Gaussian 분포 모델을 생성한다. 경계의 근처가 연속적인 투명도 값을 가지기 위해 (σ, Δ)로 결정되는 soft step function[그림 3(c)]을 정의하고, 자연스러운 경계를 위한 Data term과 Smoothness term의 합인 에너지 함수를 정의한다. Data term은 투명도 값이 사각 영역 L에 대해여 적절한지 판단해주는 역할을 한다. Smoothness term은 직전의 픽셀에서의 (σ, Δ)와 급격하게 변하는 것을 막는 역할을 한다. 그리고 dynamic programming을 하여서 에너지 함수를 최소화 시키는 (σ, Δ)들을 구하였다.



(a) (b) (c)

그림 3. (a) 이미지 위에서 Trimap을 포현한 그림. (b) 경계의 픽셀에서의 만든 직선. (c) 투명도를 정해주기 위한 파라미터(σ, Δ)로 결정되는 soft step function.

1.4. Conclusion

논문을 읽고 직접 구현을 했다는 사실이 뿌듯하였다. Graph-Cut 알고리즘과 Border-matting으로 전경을 자연스럽게 분리하는 것이 신기했다. 그리고 논문을 이해하기 위한 과정에서 Graph-Cut에 대해서 공부도 많이 해서 좋았다. 그리고 구현을 하면서 〈컴퓨터 비전〉 수업에서 배운 내용을 많이 도움이 되었다. 사진에서 전경을 분리하는 작업에서 컴퓨터 비전의 지식이 중요한 역할을 함을 느꼈고, 많은 공부가 되었다.

SPH[2]

2.1. Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics는 연속적인 유체의 흐름을 비연속적인 컴퓨터 환경에서 표현하기 위한 수단이다. 유체를 Particle이라는 수많은 작은 개체로 표현하여, 각 개체의 속도, 위치, 적용되는 힘을 계산한다. 실제로는 물 18g에는 1몰(6.02 x 10^23) 만큼의 분자가 존재한다. 컴퓨터에서 매번 이만큼의 분자들을 계산하는 것은 불가능하므로, 분자보다 큰 단위 (파티클)로 유체를 나누어서 표현한다. 또한 하나의 파티클에 다른 파티클이 미치는 힘을 모두 계산하지 않고, 유의미한 거리 내의 파티클에서만 계산하는데, 이를 위해 Smoothing Kernel을 사용한다. 각 파티클에 적용되는 힘을 계산하기 위해서 Navier-Stokes 방정식을 사용한다. 이번 연구에서는 비압축성 유체(물)를 표현하는 Navier-Stokes 방정식을 사용했다. 유체 내의 파티클에는 압력, 점성, 표면장력, 중력 총 4가지 힘이 작용한다.

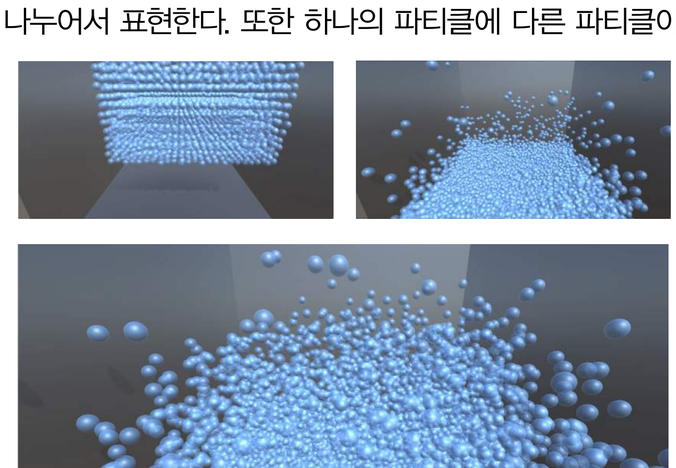


그림 4. Smoothed Particle Hydrodynamics 로 구현한 물.

2.2. SPH 구현

단순히 Single thread에서 동작하는 SPH는 구현이 간단하다. Update 루프를 돌면서 각 파티클에 적용되는 모든 힘을 계산하고, 힘을 계산한 후 계산한 힘의 방향대로 작은 거리를 이동한다. 하지만 Single thread에서 동작할 경우 속도가 느리고 많은 파티클을 사용할 수 없다는 단점이 있다. Multi thread 환경으로 구현할 경우 이 문제를 해결할 수 있다. Entity Component System은 Multi thread 프로그램을 작성할 수 있는 하나의 Paradigm으로 이번 연구에서 ECS를 이용하여 Multi thread 환경을 구현하였다. Single Thread에서 120 파티클에서 평균 8fps를 보여준 것에 반면, Multi Thread에서는 5,000파티클에서 평균 60fps의 결과를 확인할 수 있었다.

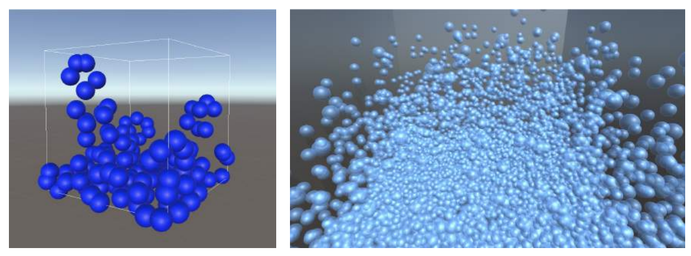


그림 5. Single Thread 환경의 SPH와 Multi Thread 환경의 SPH.

2.3. 상수 값 설정

이상기체 상태방정식을 예를 들면, PV = nRT의 식에서 R는 기체 상수라고 한다. 이처럼 다양한 물리식은 특정 상숫값을 갖는다. 비 압축성 유체를 표현하는 Navier-Stokes 방정식에는 기체 상수나 점성 계수와 같은 다양한 상숫값이 등장한다. Simulation에서는 이 상숫값들을 물리적으로 올바른 값을 사용하거나, 물리적으로 올바르진 않지만 미적으로 아름다운 결과를 얻기 위해 임의의 값을 선택하기도 한다. 아래는 Multi Thread 환경에서 물리적으로 올바른 Simulation과 미적으로 아름다운 결과를 얻을 수 있는 Simulation을 보여준다.

2.4. Conclusion

SPH로 유체(물) 시뮬레이션을 만들어보았다. 만들어 보면서 가장 크게 느꼈던 것은 시뮬레이션을 디버깅하는 과정이 쉽지 않다는 것이다. 일반적인 프로그램은 문제가 발생했다는 것을 '원하는 결과' 와 '확인한 결과' 를 비교함으로써 알 수 있다. 하지만 시뮬레이션에서는 수치적으로 계산하지 않으면 '원하는 결과' 가 무엇인지 정확하게 알 수 없었다. 또한 멀티 스레드환경도 디버깅 하는 것을 더욱 어렵게 만들었다. 설정한 상숫값을 재확인하고 코드를 디버깅하기 쉽게 코드를 리팩토링 하면서 문제를 해결할 수 있었다. 이번 연구를 통해 까다로운 디버깅을 침착하게 수행하는 법을 배운 것 같아서 보람차고 뜻 깊었다.

2018 VCLab 동계 미디어 융합 연구

Researcher 한 창 헌* 정 승 찬† 김 예 진‡ 이 주 석§ Professor 신 현 준¶

AJOU UNIVERSITY

Department of Digital Media, Ajou University
*ehwjs1914@ajou.ac.kr, †tmdcks6628@ajou.ac.kr, ‡dpwls273@ajou.ac.kr, §dljoo@ajou.ac.kr, ¶joony@ajou.ac.kr

digitalMEDIA

Image Completion[3]

3.1. Abstract

본 논문을 참고하여, opencv를 이용해 이미지의 특정 지워진 부분을 자연스럽게 채워 넣는 프로그램을 만들었다. 논문에따라 Fast approximation, Confidence map and Level set, Adaptive neighborhood, Search, Composite의 순서대로 진행하였다. 먼저 Fast approximation과 Confidence map을 토대로 어떤 픽셀을 먼저 채울지 결정한다(Level Set). 그리고 그 픽셀 주변을 어느 정도의 크기로 채울지(Adaptive neighborhood) 결정한 다음, 그 빈칸을 어떻게 채워 넣을지를 해당 이미지의 나머지 부분에서 찾는다(Search). 그러고 나서 '채울 부분(target fragment)'과 '참고할 부분(source fragment)'을 적절하게 합성(composition)하여 빈칸을 채워 넣는다. 각 단계별 역할 및 구현 내용은 다음과 같다.

3.2 Fast approximation과 Level set

이 단계는 search 단계를 수행할 때에 target fragment와 비슷한 source fragment를 찾기 위해서 비어있는 target fragment에 적절한 값을 임시로 넣어놓는 작업이다. 약 3회만큼 이미지를 pyramid down 시키고 up하는 것을 반복하여 주변의 픽셀과 비슷한 값들이 빈 부분에 채워지게 된다. Confidence map and Level set에서는 각 픽셀의 신뢰도와 target fragment의 후보를 결정한다. 이미 채워져 있는 부분은 confidence값으로 1을 가지게 되고, 나머지 픽셀은 자신과 주변의 픽셀들이 얼마나 채워져 있는지를 바탕으로 confidence값을 가지게 된다. level set은 다음으로 채워질 target fragment를 결정하기 위한 값으로, confidence값이 1보다 작으면서 비교적 높은 confidence값을 가진 픽셀이 다음 타겟으로 결정되게 된다.

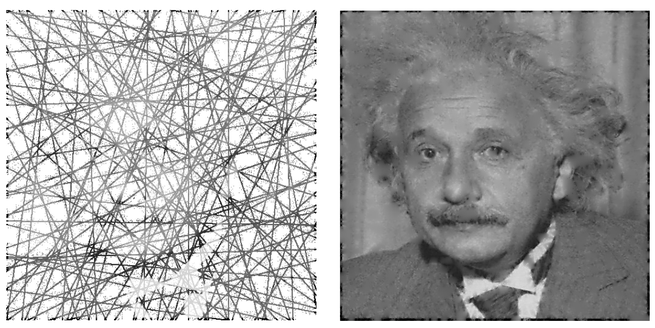


그림 6. Fast approximation 전과 후

3.3. Seach and Composition

어떤 부분을 채울지가 결정되면 Adaptive neighborhood 에서는 그 픽셀 주변 몇 개의 픽셀들을 함께 채울지를 결정한다. 즉 target fragment의 크기를 결정한다. 자신 포함 주변 픽셀들 간의 값 차이가 큰 픽셀(주로 edge에 해당하는 부분)은 작은 fragment size를 가지게 되고, 주위 픽셀들이 비슷한 값을 가지고 있는 부분은 큰 size로 결정된다. Search에서는 fast approximation에서 채워 넣어진 값을 바탕으로 target fragment와 가장 비슷한 source fragment를 찾는다. target과 source fragment의 각각 대응되는 픽셀들끼리의 color, luminance, gradient 값이 비슷해야하며 source가 target보다 높은 confidence값을 가지고 있어야한다. 마지막 Composition은 source와 target fragment를 합성하는 단계이다. 이때 target fragment 중 이미 값을 알고 있는 부분은 지워지면 안되기 때문에 inverse alpha값이 1이 아닌 부분만 합성이 되도록 한다. 또, 합성 경계선이 뚜렷하게 나타나지 않도록 가우시안 필터를 사용한다.

그림 7. Image Completion 결과

3.4. Conclusion

Fast approximation을 통해 일부 데이터로 실제 그림과 유사한 그림을 만들 수 있다는 게 놀라웠다. 어려웠던 부분은 Search 단계 부분이다. 단순히 대응되는 픽셀 값들을 비교하는 것만으로는 비슷한 fragment를 찾기 힘들다는 것을 깨달았다. 공부를 통해 영상 데이터에 대한 통찰력을 키워서 더 좋은 Search 알고리즘을 찾고 싶다. 일련의 단계들을 하나씩 수행 하면서 Image Completion 프로그램을 그럴듯하게 만들어냈다는 것이 기쁘다. 컴퓨터 비전에 대한 공부를 더 해보고 싶다는 생각이 들었다.

Morphable 3D Face Model[4]

4.1. Morphable Face

Morphable Face는 data set을 이용하여 morphable(변형 가능한) 얼굴이다. 본 논문에서는 총 200개의 전처리한 face model을 사용하여 하나의 morphable face model을 만들었고, 200개의 data를 이용해 주성분을 추출하여 얼굴의 특징을 바꾸거나 다른 어떤 얼굴에도 합성할 수 있는 변형이 가능한 face model을 구현했다.

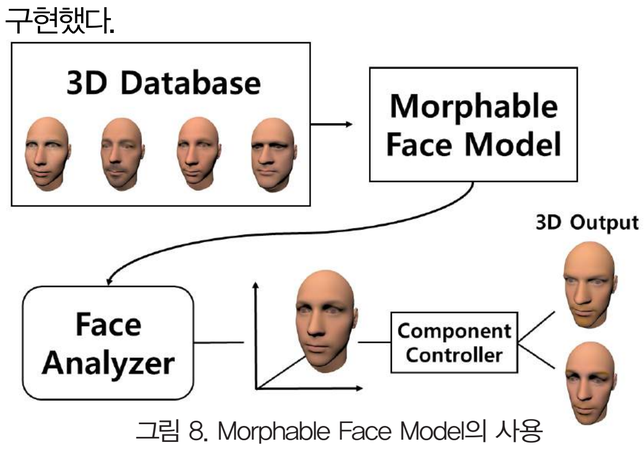


그림 8. Morphable Face Model의 사용

4.2. 3D Model Visualization (by Open GL)

최종적인 morphable model을 만들어 내기 위해 주어진 data를 3D model로 시각화 해준다. 주어진 데이터들을 3D model로 표현해주기 위해 기본적인 Open GL library와 함께 별도의 .obj/.mtl 파일을 읽어주는 load 기능을 구현해서 얼굴 3D model을 구현해준다. 주어진 obj 파일에는 model의 vertex 좌표 정보, normal 좌표 정보, UV texture 좌표 정보, face index 정보로 구성되어 있다. load 과정에서는 vertex, normal, UV texture 좌표들을 각각 저장해주고 face index 순서로 각 좌표를 정렬하여 buffer object에 전달해주는 작업을 해준다. mtl 파일에서는 diffuse, specular, ambient 색상 정보와 texture image 파일에 대한 정보를 따로 저장하여 사용한다.

4.3. Blend Faces & PCA

주어진 model을 표현하는데 사용한 vertex, normal, UV texture 좌표들과 texture image들을 하나의 model로 표현하기 위해 섞어주는 작업을 해준다. 각 좌표들은 혼합 비율에 맞게 계산을 하여 새로운 좌표로 계산하고, texture image는 blending 함수를 이용하여 혼합 비율에 맞게 새로운 image를 만든다. 여기까지는 단순히 평균적인 face model을 만드는 과정이고, 더 나아가 모든 model에 대해 각 vertex data를 이용하여 matrix를 만들어 Open CV의 PCA(Principal Component Analysis)를 이용해 PCA model을 만들어 준다. PCA model로부터 shape(x, y, z)에 대한 eigen vector를 구하여 원하는 혼합 비율을 곱하여 morphable model의 모양을 만들어 낸다. 마찬가지로 texture image(R,G,B)에 대한 PCA model로 eigen vector를 구해서 morphable 모델의 texture를 결정해 주도록 한다.

$$S_{mod} = \sum_{i=1}^{m} a_i S_i, T_{mod} = \sum_{i=1}^{m} b_i T_i$$

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m} b_i = 1$$

수식 1. 기존 morphable model을 구하는 방법

$$S_{model} = \bar{S} + \sum_{i=1}^{m-1} \alpha_i s_i$$

$$T_{model} = \bar{T} + \sum_{i=1}^{m-1} \beta_i t_i$$

수식 2. PCA를 이용하여 model을 구성하는 방법
(s와 t는 PCA로 구한 주성분 vector, α β 는 혼합 비율)

그림 9. 20개의 face model로 만들어진 morphable face model

그림 10. 주성분을 이용한 변형된 face model (4, 6, 11 주성분)

4.4. Conclusion

그림 10에서 각 주성분이 얼굴에서 어떤 특성과 경향성을 나타내는지 확인할 수 있다. 안와, 눈, 턱 등 얼굴의 특징이 주성분에 따라 다르게 영향을 받는다. 이를 이용해 morphable face로부터 원래의 얼굴을 얻어낼 수도 있고 완전히 새로운 얼굴을 만들어 낼 수도 있는 진정한 morphable face model을 만들어 낼 수 있다. 즉, 논문과 같이 input image를 통해 합성된 새로운 3D 얼굴 모델을 도출 해내는 기능을 구현하는 쪽으로 생각할 수 있다. 최적화 방식을 더 공부하여 이 기능도 구현해 볼 수 있을 것 같다.

Reference

1. Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. "GrabCut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. 23, 3 (August 2004), 309–314.

2. Matthias Muller, David Charypar, and Markus Gross. 2003. Particle–based fluid simulation for interactive applications. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '03). Eurographics Association, Aire–la–Ville, Switzerland, Switzerland, 154–159.

3. Iddo Drori, Daniel Cohen–Or, and Hezy Yeshurun. 2003. Fragment–based image completion. In ACM SIGGRAPH 2003 Papers (SIGGRAPH '03). ACM, New York, NY, USA, 303–312.

4. Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3D faces. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99). ACM Press/Addison–Wesley Publishing Co., New York, NY, USA, 187–194.