멀티미디어 시스템 9조 프로젝트 보고서

Canny Edge & Stereo Depth

2011104023 송현수

2011104043 장원엽

2014104019 이인홍

2010104028 김지후

목차

1. 프로젝트 주제

2. 프로젝트 설계

3. 각 기능의 이론 및 구현

a. Stereo Depth

b. Canny Edge 및 Labeling

c. Application

4. 역할 분담 및 일지

5. 결론

**1. 프로젝트 주제**

첫 프로젝트 주제는 에지와 뎁스를 통한 오브젝트 디텍션으로, 에지를 통해 각 경계를 파악 후, 이 경계에 따라 레이블링하고, 각 레이블의 경계 (에지) 부분에서의 뎁스 값이 비슷한 경우 같은 오브젝트로 인식하고 레이블을 병합하여 하나의 오브젝트로서의 레이블링을 완성하는 방향으로 간단한 오브젝트 디텍션을 하고자 함.

그러나 발생한 여러가지 문제로 인해 주제를 조금 바꾸어 canny edge와 depth map을 직접 구하는 것으로 선회하였음. 이에 따라 오브젝트 디텍션을 축소하고, 케니와 depth가 잘 추출될 경우 어플리케이션 차원에서 무엇이 가능한지에 대한 예시로서 보이는 것으로 변경.

**2. 프로젝트 설계**

C#

OpenCV

Emgu

Form

Image Processing

Application

윈도우 프로그램으로 제작하기 위해서 C# 윈도우 어플리케이션 폼 기반 개발.

C# 내에서 이미지 프로세싱을 위해 OpenCV를 사용. 이를 위해 Emgu를 이용하였음.

프로그램 크기가 작고 순차적인 작업을 하기 때문에 간단한 구성을 가지도록 하였다.

이미지 Import

윤곽선 추출

Depth 추출

Labeling

Label 통합

Application

**3. 각 기능 이론 및 구현**

**a. Stereo Depth**

뎁스를 구하는 방법은 크게 스테레오 비전과 하드웨어적인 뎁스 카메라 이용 두가지 방법으로 나눌 수 있다. 본 프로젝트에서는 스테레오 비전을 통해 depth를 구하기로 결정함. 고정된 위치와 방향에서 촬영한 영상을 활용하는 경우에 대한 구현 시도

|  |  |  |
| --- | --- | --- |
| Left Image | Right Image | 직접 구한 Depth |

예시 영상에서 좌우 영상간 에피폴라 라인에서, SSD (Sum of Squares of Distance) 알고리즘으로 매칭점을 찾아 disparity map을 구하였다. 다소 노이즈가 심하였으나 충분한 결과을 얻었다고 판단. 그러나 이는 고정된 두개 이상의 카메라에서 좌우 영상을 촬영해야 하기 때문에 실제 상황에서 사용자가 단일 카메라로 찍은 영상에서는 사용이 불가능하다는 문제점을 발견하였다.

따라서 SSD보다 강인한 피쳐 디텍터를 이용해 매칭점을 찾고, 이에 대한 disparity를 구하는 방법으로 뎁스를 구하고자 하였다. 빠르고 강인한 디텍션을 위해 SURF (Speed Up Robust Feature)를 사용하게 되었다.

|  |  |
| --- | --- |
|  |  |
| 이미지들 중 기준 이미지 | 직접 구한 뎁스 |

SURF는 openCV에서 제공하는 라이브러리를 사용하였다. 다수의 이미지에서 SURF 특징점을 찾은 후, 원하는 이미지와 나머지 이미지들 간의 피쳐 매칭값이 일정 값 이상인 경우, SURF에서 제공하는 호모그래피 트랜스폼 이후 두 특징점 간의 거리를 측정하여 disparity로 선정하였다. 고정된 위치에서 촬영하여 에피폴라 라인에서 찾은 매칭점보다 심각하게 sparse한 결과를 얻게 되었으나, 일반 단일 카메라에서 촬영한 영상에서도 뎁스를 얻을 수 있게 되었다.

**b. Canny Edgy 및 Labeling**

b.1. 에지 검출이란?



위 사진과 같이, 2차원 이미지에서 사물의 윤곽선을 찾는 작업을 말한다.

b.2. 왜 에지를 검출하는가?

영상을 통해 특정 물체를 인식하기 위해서 우리는 검출된 에지를 활용할 것이다.

b.3. Canny edge detection algorithm

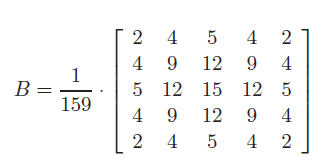
캐니 에지 검출 알고리즘은 1986년 John F. Canny에 의해 개발 된 알고리즘이다.

명확하게 하나의 선으로 에지를 찾을 수 있다는 점과 상, 하한 임계 값 설정을 통하여 다양한 환경에 적용할 수 있다는 장점이 있다. 구현이 복잡하고 실행 시간이 타 에지 검출 알고리즘에 비하여 길다는 것이 단점이다.

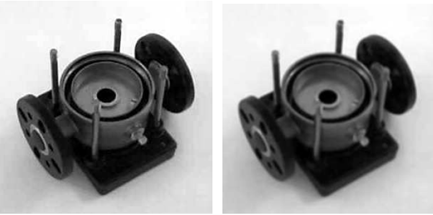
b.4. Canny Edge Detection Algorithm의 구현

1. Smoothing

이미지로부터 노이즈를 제거하기 위한 작업이다. 이를 위해서는 가우시안 필터를 적용한다. 가우시안 필터 값은 다음과 같다.



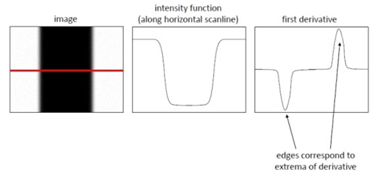
필터 적용 전과 후의 이미지 변화 결과는 다음과 같다.



확실히 이미지가 부드러워진 것을 확인할 수 있다.

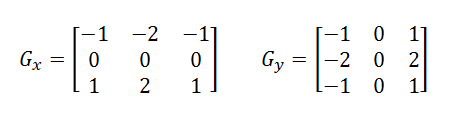
1. Finding gradients

* 이미지로부터 에지를 검출한다. 우선 예시로 아래 그림을 보자.



첫 번째 이미지에서 가로로 빨간 선을 그린 부분의 값을 그래프로 표현하면 두 번째 이미지와 같은 모양의 그래프가 나온다. 저 그래프를 미분한 것이 세 번째 이미지의 그래프이다. 눈 여겨 보아야 할 점은, 흰색에서 검은색으로 변하는 경계부분과 검은색에서 흰색으로 변하는 경계부분에서의 미분 값이 각각 극 값을 나타낸다는 점이다.

이 극 값을 Extrema라고 하는데, Extrema의 검출을 위해서는 Sobel Mask를 이용한다. Sobel Mask필터의 X축과 Y축에 대한 인자 값은 아래와 같다.



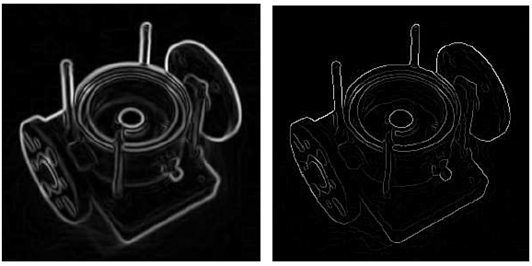
이렇게 X축과 Y축에 대해서 미분을 하면 각각 Extrema를 얻을 수 있고 결과는 아래와 같다.



1. Non-maximum suppression

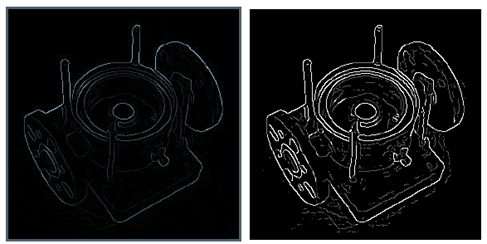
에지가 아님에도 에지로 검출 된 영역이 존재할 수 있다. 그러한 영역을 Non-maximum이라고 하며 이를 제거해야 한다. 이는 각각의 픽셀에 대하여 주변 픽셀의 값과 비교하여 현재 픽셀의 에지 강도를 비교 판단하는 것이다. 현재 픽셀의 에지 강도가 가장 크다면 Maximum, 아니라면 Non-maximum이 된다.

이를 적용한 이미지는 다음 사진과 같다.



1. Double thresholding

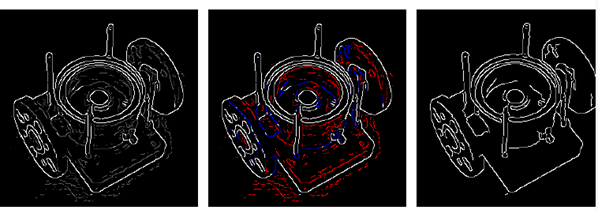
위의 Non-maximum suppression을 통해 얻은 에지들 중에는 실제 에지와 약간의 노이즈가 존재하는데, 이들을 구별하는 작업이 필요하다. 이는 임계 값을 설정하여 구분할 수 있다. 즉, 상한 값과 하한 값을 설정하여 하한 값 아래의 값은 제거하고, 중간 값은 흐리게, 상한 값 이상의 값은 진하게 표현한다. 그 결과는 아래와 같다.



1. Edge tracking

위에서 얻은 약한 에지와 강한 에지의 연관성을 BLOB-analysis를 이용하여 파악하는 단계이다. 즉, 약한 에지가 강한 에지와 연결되어있는가를 확인하는 작업이다. 만약 강한 에지에 연결되어 있다면 에지로 남기고, 그렇지 않다면 제거하는 과정을 거친다.

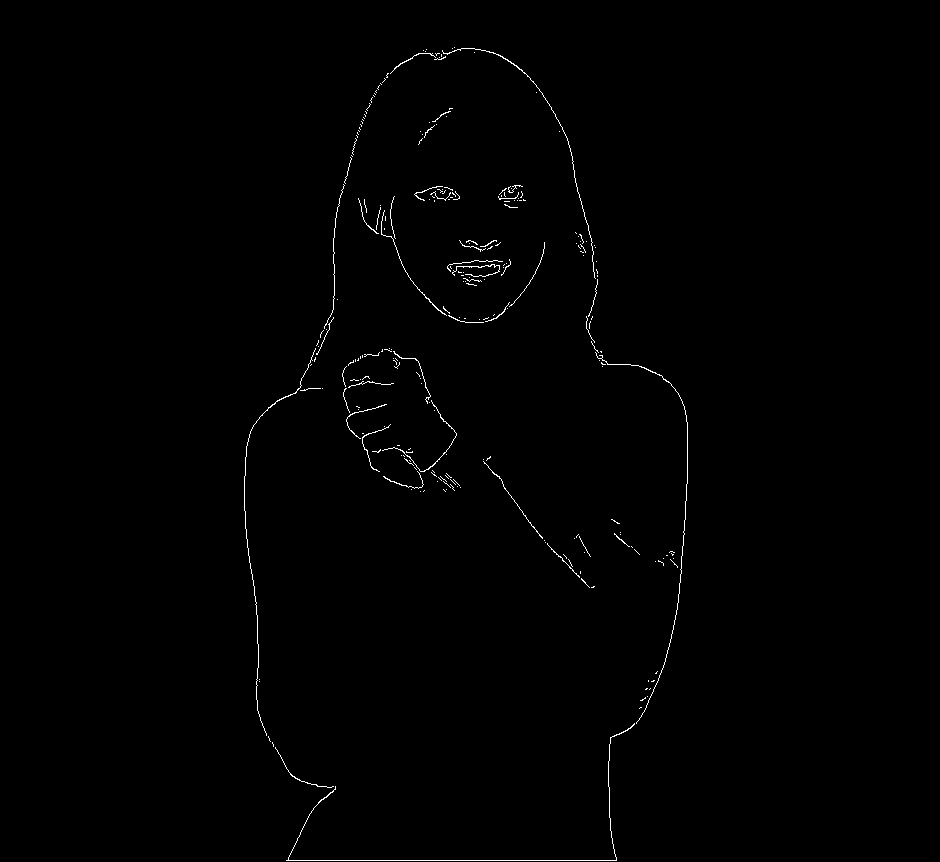
아래 그림을 보자.



첫 번째 그림의 회색 에지에 대하여 BLOB-analysis를 수행한다. 두 번째 이미지는 그 결과인데, 빨간색과 파란색 선이 표시되어 있다. 빨간색 선은 강한 에지와 연관이 없는 에지로 삭제해야 할 에지이며, 파란색 에지는 강한 에지와 연관이 있는 에지로 최종적으로 남게 될 에지이다. 세 번째 이미지는 최종 결과이다. 에지가 깔끔하게 나온 것을 확인할 수 있다.

1. 예시

다음 이미지는 프로젝트 발표에 사용했던 이미지 예제이다.



원본 Canny Edge