낮은 SNR에서의 빠른 곡선 검출

이준협

Fast Detection of Curved Edges at Low SNR

Junhyub Lee

요 약

논문 저자가 제시하는 Beam-curve tree 알고리즘 기반 빠른 곡선 검출기를 이해하고 이를 사용해 다양한 low SNR 환경에서 실험을 진행한다. 이후 이에 대한 각각의 성능을 평가 및 분석한다.

Abstract

In this report, we implemented beam-curve tree based algorithms for fast detection of curved edges. There are two type of version in beam curve tree algorithm. Each of their time complexity is O(N^1.5) and O(NlogN) respectively. Experimental analyses are provided to compare with other edge detection algorithms in various SNR environments.

Keyword

Beam-curve tree, Sliding window, Divide and conquer

Github

https://github.com/jjunhyub/computer_vision_project_final

1. 서 론

컴퓨터비전 분야에서 검출이란 상당히 중요한 부분을 차지하고 있다. 특정 점에 대한 검출, edge에 대한 검출, object에 대한 검출 등 다양한 방법과 목적을 가지고 영상 내 검출을 시도한다. 여기서 우리는 특정 점과 더불어 가장 기본이 되는 요소 중 하나인 edge 검출에 대한 연구를 진행하였다. 이미 많은 edge 검출 연구들이 진행되었으며 다양한 edge 검출기들이 존재하지만 (canny, sobel, etc.) 한 종류의 검출기가 모든 환경에서 최고의 성능을 보이는 상황이 아닌 특정 환경에서 고성능을 보이는 검출기들이 각각 있는 상황이다.

좋은 edge 검출기가 되기 위해서는 다음과 같은 측면들이 고려되어야 한다. 첫째, 영상이 들어왔을 때 빠른 속도로 edge를 검출할 수 있어야 한다. 둘째, 다양한 크기, 해상도, 화질에 대해 edge를 검출할 수 있어야 한다. 셋째, edge의 색깔 및 모양에 상관없이 검출할 수 있어야 한다. 저자는 위 3가지 사항을 모두 만족시키는 edge 검출 알고리즘을 제

* 소속

시한다. 해당 알고리즘은 저자가 새롭게 제시하는 beam curve tree 구조가 핵심이 되어 bottom up 방식으로 edge검출을 진행한다.

Edge 검출기 관련 연구로는 sobel operator [1], Marr & Hildreth [2], Canny [3], wavelet based bank of filter [4]가 있다. sobel 과 Marr & Hildreth 검출기는 노이즈에 취약하며 low SNR 환경에서 상당히 낮은 퍼포먼스를 보인다. Canny 는 위 두개의 검출기와는 다르게 노이즈에 어느정도 견딜 수 있었지만 너무 느린것이 단점이다. 노이즈에 강한 모습을 보이기 때문에 본 논문에서는 기본 비교대상을 canny edge detection으로 설정하였다. wavelet based bank of filter 방법은 상당히 빠르지만 곡선 검출에 취약함을 보이고 low SNR 환경에서도 한계점을 보인다. 반면에 저자의 방법은 low SNR에서 빠르고 정확하게 직선뿐만아니라 곡선도 검출해낸다.

2,3장에서는 저자가 제시한 beam curve tree의 구조와 edge 검출 알고리즘의 내부 구조에 대한 설명을 각각 진행하고, 4장에서는 상용화된 검출기들과의 결과 비교 및 분석을 통해 성능 평가를 하겠다. 마지막으로 5장에서는 본 연구에 대한 결론 및 느낌점에 대해 서술하겠다.

2. Beam Curve Structure

본 문단에서는 low SNR에서 빠른 edge 검출을 위해 제안되었던 알고리즘의 뼈대가 되는 beam curve 자료구조를 소개하고 해당 자료구조가 어떻게 사용되는지를 시각 자료와 함께 알아본다.

Beam curve 자료구조는 영상의 특정 순간 일부 영역에 대해 벡터의 형태로 총 4가지 값을 포함하고 있다. 각각의 값을 [R, L, C, P] 로 표현한다. 모든 Beam curve는 현재 영역에 대해 여러 곡선 후보군들을 포함하고 있다. R은 현재 바라보고 있는 영역에 대해 포함하고 있는 각각의 곡선들을 기준으로 2개의 영역으로 나누어지는데 이 때 총 intensity difference를 의미한다. n개의 후보군 곡선이 있는 경우 n개의 intensity difference를 들고있는 vector가 R이 된다. L은 각 후보군 곡선들의 길이를 의미한다. n개의 후보군 곡선이 있는 경우 n개의 곡선의 총 길이를 저장한 벡터가 L이 된다. C는 R을 L로 나눈것으로 전체 intensity difference를 길이로 나눈값이다. 단위 길이당 instensity difference를 의미하게 되며 마찬가지로 n개의 값을 가진다. 마지막으로 P는 각 후보군 곡선들의 모든 픽셀값을 가지는데 벡터안에 벡터 구조로 되어있다. n개의 후보군 곡선에 대해 n개의 벡터가 있으며 각 벡터 내부는 실제 곡선을 이루고 있는 픽셀정보가 들어있다. 이 벡터의 길이는 곡선별로 다르다. 이렇게 RLCP를 정의할 경우 두 Beam curve에 대해 합연산을 매우 간단하게 정의된다.

$$\phi(\gamma_1+\gamma_2) = [R_1+R_2, L_1+L_2, \frac{R_1+R_2}{m(L_1)+m(L_2)}, P_1 \cup P_2]. \quad (1)$$

Beam curve각 각 후보군 곡선들을 빠르게 더하고자 하는 목적으로 정의된 자료구조이다.

3. Beam Curve Binary Tree

본 문단에서는 위에서 정의된 Beam curve를 어떻게 활용하여 빠른 edge 검출을 하는지에 대한 전체적인 흐름 설명 및 트리의 노드를 합치는 부분의 최적화 과정을 설명한다.

이미지의 크기가 기준 길이()29pixel1보다 작을 경우 바로 edg e검출을 진행하면 되지만 이보다 클 경우 sliding window로 이미지를 잘라 multi thread 방식으로 동시에 진행한다. 따라서 beam curve tree는 초기 입력 이미지의 길이가 129 pixel로 동일하다.

이미지가 들어오면 divide and conquer 방식으로 이진 트리를 구성해나간다 (그림1 참조). 가로가 길면 가로를 반으로 자르고 세로가 길면 세로를 반으로 자른다. 이렇게 양쪽 픽셀의 길이가 5픽셀 이하가 될때까지 분할해준다. 5픽셀 이하의 이미지에 대해서는 각 모서리의 gradient가 급격하게 변하는 지점을 전부 찾아준 뒤 모든 경우에 대해 선분을 만들어 후보군 곡선을 만들고 이를 가지고 beam curve를 구성한다.

만들어진 beam curve를 합쳐나가면서 conquer해주는 방식이 저자가 고안한 fast curved edge detection

이다. 두 하위 노드를 보았을 때 자른 단면에 대해 후보군 곡선이 시작되는 점들을 파악하고 그 점들로 부터 존재하는 곡선들의 조합을 모두 적용하여 새로운 후보군 곡선을 만들어준다 (그림2 참조). 다만 모든 경우를 계산하는 방법은 시간이 너무 오래 걸리기 때문에 후보군 곡선을 만들어 본 뒤 intensity difference와 특정 threshold를 비교해 한번 거르고 시작한다. 이때 모든 후보군쌍의 경우를 전부 해주는 경우가 original 버전이고 절단면을 기준으로 가장 intensity difference가 컸던 점들에 대해서 후보군 곡선을 찾아주는 방법이 optimized version이다. 당연히 모든 후보군을 해보는 방식이 퍼포먼스가 약간 더 좋았지만 두 방식의 time complexity가 달라지기 때문에 저자는 후자의 방식을 권장한다. Original 방식은 약 O(N^1.5) 이며 optimized 방식

은 약 O(NlogN)이다. 이렇게 합쳐나가는 과정은 원본 이미지 크기까지 해주면 edge detection이 종료된다.

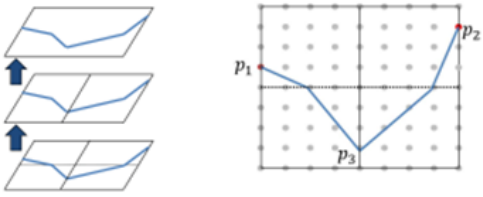

그림1

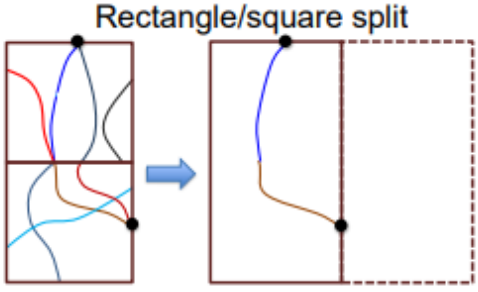

그림2

4. 결과 분석

본 문단에서는 논문에서 제시된 이미지들과 새로운 이미지들의 결과물들을 다른 edge detection 알고리즘들과 비교 및 분석을 진행한다. 그림3에서 윗줄은 아래줄보다 노이즈가 더 큰 경우이다. 왼쪽에서부터 차례대로 원본, beam curve, canny, PMI의 결과물이다. 성능적인 측면에서 canny 가장 흡사하지만 S위에있는 짧은 직선을 canny는 검출하지 못하지만 본 논문의 방법은 검출한다. 여러 SNR 환경에서 성능이 더 뛰어난것을 보여준다. 그림3이 의도적으로 만들어진 이미지에 대한 결과분석이라면 그림4는 실제 데이터에 대한 실행 결과물이다. Beam curve의 결과물이 canny보다 더 세밀한 edge들을 검출하는 것을 확인할 수 있다. 그림 5는 논문에서 제시하지 않은 데이터에 대해 실행한 결과물이다. low SNR에서는 잘되지만 오히려 high SNR 환경 즉 노이즈가 없는 깔끔한 환경에서는 오히려 edge 검출이 잘 되지 않는 모습을 보여주었다. 이러한 부분은 언급되지 않은 논문의 한계점이라고 생각하다.

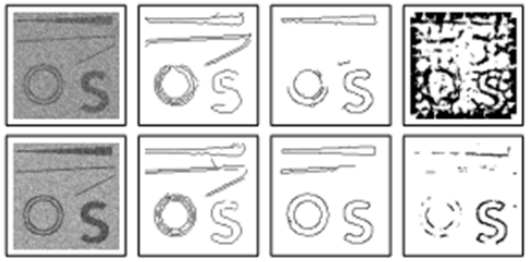

그림3

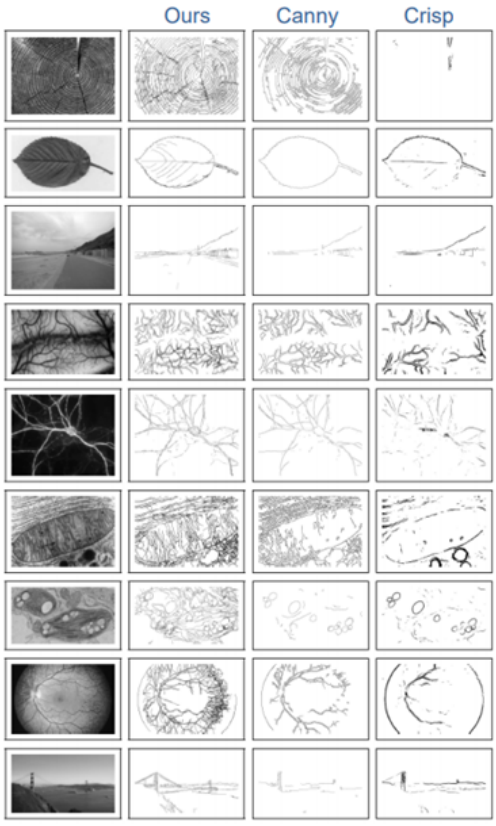

그림4

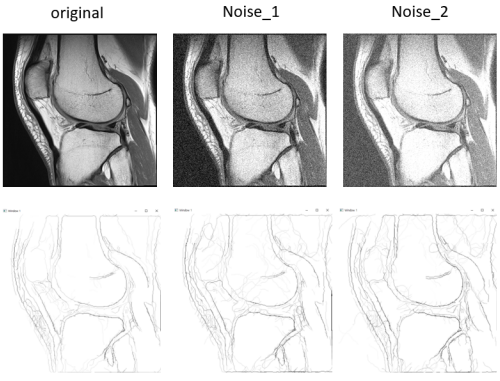

그림5

5. 결론 및 느낀점

실제로 논문을 읽고 난 뒤 구현 및 실행을 해보고 결과를 분석해본것은 처음이었다. 처음에는 글이 잘 읽히지도 않았지만 여러번 읽어보며 제시된 사이트에서 포스터 및 코드를 직접 보며 이해해보니 도움이 많이 되었다. 머신러닝으로 많은 문제를 해결하려는 요즘 흐름에서 이렇게 알고리즘적 아이디어만으로도 이렇게 훌륭한 결과물을 내는 논문을 썼다는 것이 가장 인상깊었다. 분할정복이라는 아이디어는 쉽게 나올 수도 있었겠지만 이를 구현하고 다양한 이미지에 대해 성능이 좋게 나오는데까지 발전시키는 것은 정말 어려운 작업이라고 생각한다. 유익한 경험이었고 추후 대학원 진학에 있어 상당히 좋은 밑거름이 되었다고 생가한다.

참 고 문 헌

[1] I. Sobel. Camera models and machine perception. In PhDthesis, Electrical Engineering Department, Stanford University., 1970

[2] Marr and E. Hildreth. Theory of Edge Detection. Proceedings of the Royal Society of London. Series B, BiologicalSciences, 207(1167):187-217, 1980

[3] J. Canny. A computational approach to edge detection. IEEETrans. Pattern Anal. Mach. Intell., 8(6):679-698, June 1986.

[4] P. S. P. Wang and J. Yang. A review of wavelet-based edgedetection methods. IJPRAI, 26(7), 2012