An Implementation of the “Live-Wire” Image Segmentation Tool

Based on the papers “Interactive live-wire boundary extraction” and “Interactive Segmentation with Intelligent Scissors” by William A. Barrett and Eric N. Mortensen

James Gomez

CS 269

Final Project Report

Professor Terzopoulos

*Abstract*—Fully automated image segmentation for general imagery is still an unsolved problem. On the other hand, manual segmentation is tedious, time-consuming, and often inaccurate. **In their paper “Interactive live-wire boundary extraction”, Barrett and Mortensen introduced “live-wire”, an interactive tool for fast, accurate, and reproducible image segmentation via mouse gestures. They refined their tool further in 1998, renaming the tool “intelligent scissors. This paper outlines an implementation of the live-wire tool using Java and the OpenCV computer vision library.**

# Introduction

Image segmentation via manual boundary tracing is tedious, time-consuming, and often inaccurate. On the other hand, accurate, fully-automated general image segmentation is still not a fully-solved problem. As a result, it is desirable to have interactive tools available that enable users to extract segments of interest from images. These types of tools are especially desirable in the medical field, where extraction of specific objects from medical imagery is of high interest and importance. In 1996, Barrett and Mortensen introduced Live-wire, an interactive tool for fast and reproducible image segmentation [1]. Live-wire, also known as “intelligent scissors”, frees users from the tedious job of extracting segments with manually drawn boundaries and improves both speed and accuracy.

This paper outlines a specific implementation of the Live-wire tool using the Java language and the OpenCV computer vision library. The remainder of this paper is organized as follows: in Section II, the previous work by Barrett and Mortensen is presented. In section III, the methodology and implementation details of this work are presented. In section IV, results of the implementation are presented and discussed. Finally, in section V, the paper is concluded.

# Previous Work

Barrett and Mortensen’s live-wire tool distinguishes itself from previous methods because it searches the entire image for piece-wise optimal paths as the user interacts with the boundary in real-time [1].

## Boundary Extraction

The authors outline several steps that Live-wire follows in order to extract boundaries. First, a local cost field must be generated such that those pixels with strong edge features are associated with low costs and vice versa. Secondly, the local costs must be expanded from a user specified seed-point in order to generate shortest paths from all pixels back to the seed-point. Next, real-time boundary detection and drawing is accomplished by traversing the shortest path from the user’s current cursor position back to the seed-point. Finally, the application detects closure of a boundary and extracts the pixels of that boundary.

### Local Costs

Barrett and Mortensen describe the local cost function as the weighted sum of gradient magnitude, gradient direction, and Laplacian zero-crossing features.

### Boundary Traversal via Graph Search

## Additional Features

### Path Cooling

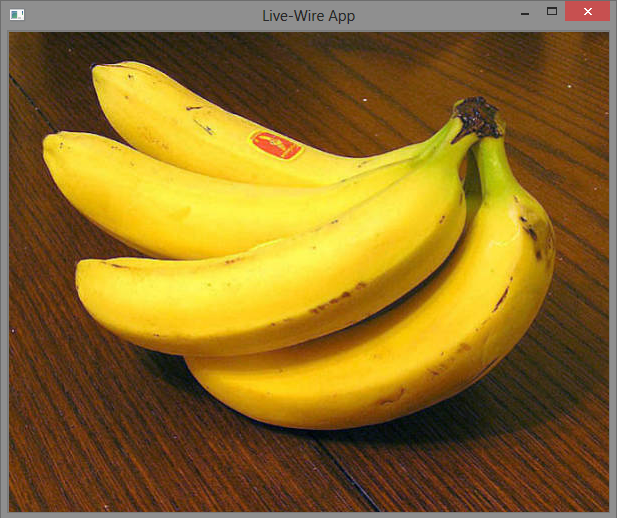
### On-the-fly Training

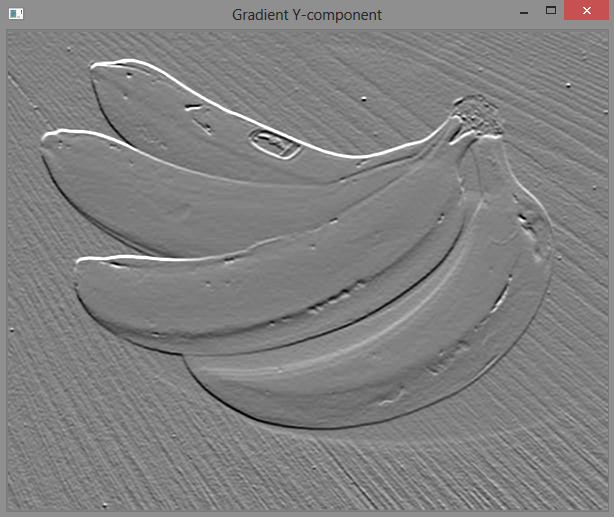
# Methodology and Implementation

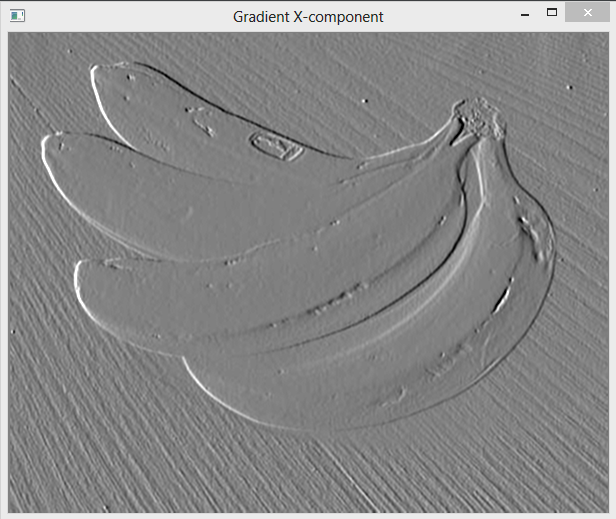
In order to keep from re-inventing the wheel, it was desirable to use a software library or platform with common image processing and computer vision algorithms readily available. It was also desirable to use a language and environment that provides both good application performance (especially during graph expansion) and implementation speed and safety. Matlab, provides an easy to use environment with high implementation speed and under-the-hood memory management, but does not offer the real-time performance required by this application. On the other hand, OpenCV, though powerful, has a standard C++ API interface. Though C++ has the quality of creating high performance native code, it provides a tedious implementation experience and memory must be explicitly managed by the programmer. Thus, this live-wire implementation is based in Java and interfaces the OpenCV libraries via the JavaCV wrapper APIs. Java provides the real-time performance needed for this application while offering a safer and higher speed implementation environment than C++.

The process of extracting a closed boundary involves several steps. First, local costs for the image pixels must be generated such that those pixels with strong edge features are assigned low costs and vice versa. Secondly, the user must select a starting point from which to calculate optimal paths to all other pixels. This involves a process called graph-expansion. Next, live-wire boundaries must be generated and drawn in real-time such that the boundary follows the shortest path from the user’s current cursor position in the image back to the original starting point specified by the user. Finally, upon drawing a closed boundary, the application must detect the closed boundary and extract the boundary pixels and the image segment enclosed within the boundary.

## Feature Extraction

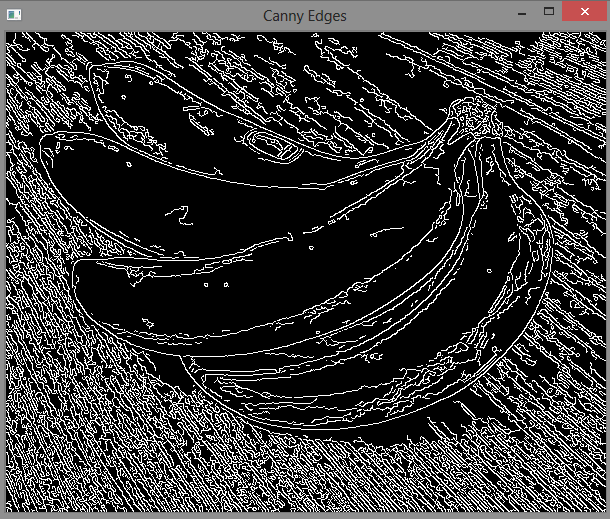


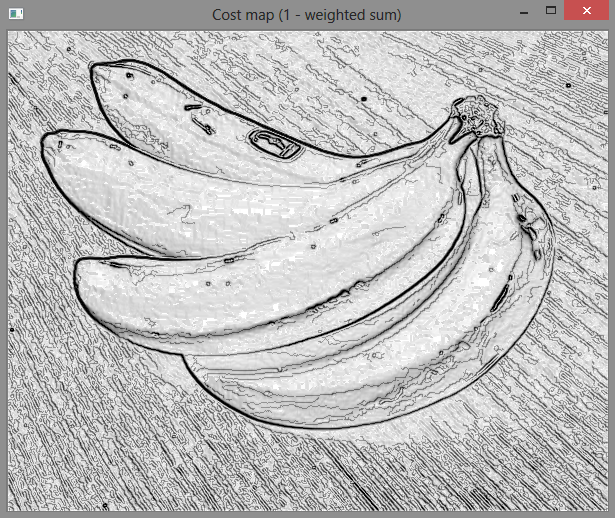












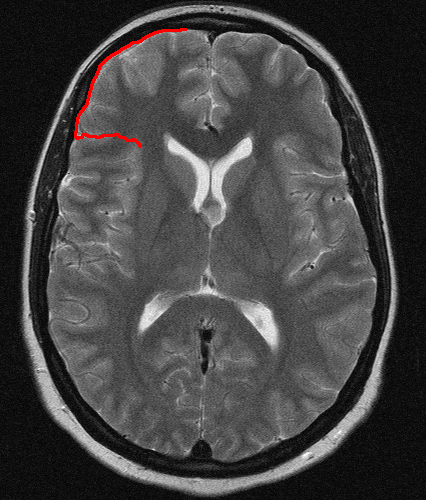
## Local Costs

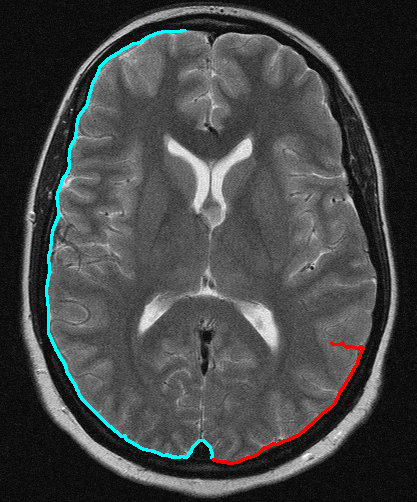
TODO

## Graph Expansion

TODO

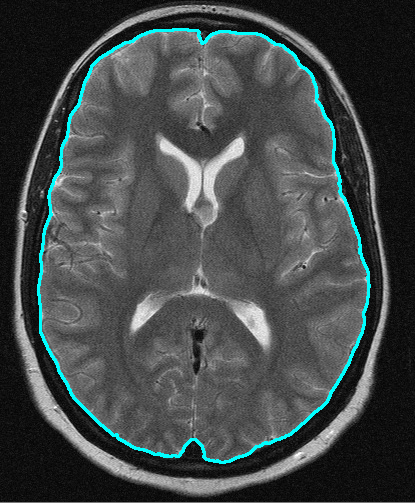
## Live-wire boundary



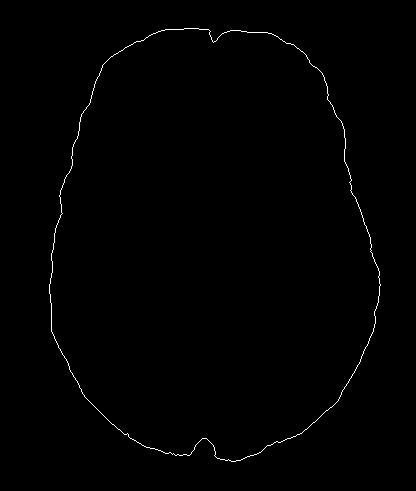


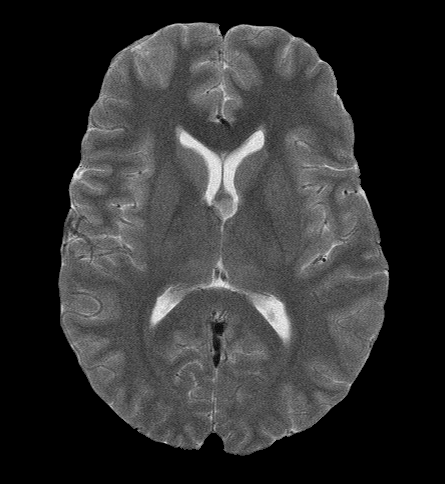
TODO

## Closed Boundary Detection



## Boundary and Segment Extraction





# Results

# Conclusion

# References

|  |  |
| --- | --- |
| [1] | W. A. Barrett and E. N. Mortensen, "Interactive Segmentation with Intelligent Scissors," *Graphical Models and Image Processing,* pp. 349-384, 1998. |
| [2] | W. A. Barrett and E. N. Mortensen, "Interactive live-wire boundary extraction," *Medical Image Analysis,* vol. 1, pp. 331-341, 1996/7. |