Objectives

- Implement the first part of the graph-cut with connectivity priors algorithm presented in [1];
- Apply this algorithm to automated image segmentation;
- Experiment with different edge weight functions aiming to increase segmentation accuracy.

Introduction

There are many approaches to image segmentation, but most of these methods can present fragmented results when dealing with thin, elongated structures and one way of dealing with this problem is to impose connectivity conditions when performing the segmentation. The employed method (Minimum planar multi-sink cuts with connectivity priors [1]) is based on graph-cuts.

The pixels are represented by vertices in the graph, and neighboring pixels are connected by edges (Fig. 2). The connectivity conditions will be given by the user, that will select a pixel in the background (source) and multiple pixels inside the object (sinks) and they will prevent the resulting object from being fragmented.

Algorithm implementation

The following figures illustrate the steps of the algorithm implementation:

- Figure 3: Graph constructed from the image (source and sinks given by the user). Green color highlights lower weight edges.
- Figure 4: Dual graph of the original grid graph. Each face in the original graph is a vertex in the dual graph.
- Figure 5: Red colored edges show the shortest paths between each pair of sinks (Dijkstra’s algorithm).
- Figure 6: The light blue vertices highlight the region of the dual graph containing the source but no vertex from the shortest paths (DFS).

Results

The segmentation below was obtained using six selected pixels as sinks, and the edge weight \( w \) was calculated by:

\[
    w = |l_1 - l_2|^{-1},
\]

where \( l_1 \) and \( l_2 \) are luminance values of the pixels.

Conclusion

The implementation of the first part of the algorithm presented good results, therefore we expect that the complete implementation will improve the quality of the segmentation. Future work includes 1. experimenting with a variety of edge weight functions to increase the segmentation accuracy; and 2. heuristically decrease the running time by testing different pre-processing steps.

References