1. Graph Cuts on GPU

1.1 Push-Relabel algorithm

// describe push-relabel algorithm

1.2 Push-Relabel implemented on CUDA

1.2.1 Graph construct on CUDA

Symbols A(i), B(i) are respectively gray level values at pixel i corresponding to two parts of the image that are determined to be the area overlapping.
Formula that calculate weight between 2 adjacency edges taken from graphcut textures paper of Vivek Kwatra Arno Schodl Irfan Essa Greg Turk Aaron Bobick.

$$c(u,v) = c(v,u) = A(u)\text{-}B(u) + A(v)\text{-}B(v) \tag{1}$$

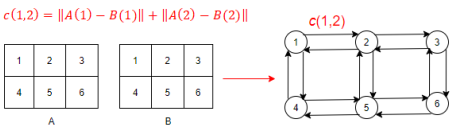

Figure 1: Figure 1: Graph construction

1.2.2 Data representation (unify memory)

We construct a 4-connectivity graph, so we need 4 arrays with each array size equal —V—, V is set of vertices of graph G to hold 4 weights for each node corresponding it's 4 adjacency edges. To access all weights of node i, we access i-th element of all 4 arrays. If the node lacks any adjacency edge, the weight on that side equal 0. One array of size —V— contain excess flow of all vertices and one array of size —V— hold heigh of all vertices. Relabel kernel ollow the push kernel. Relabel kernel needs to know which nodes are in need of relabeling, so it is necessary to have a relabel mask array of size —V— to mark the nodes that need relabel.

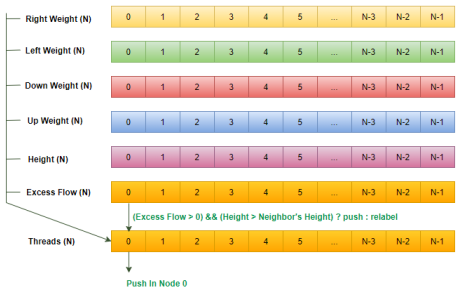

Figure 2: Figure 2: Data for push kernel

Figure 3: Figure 2: Data for push kernel

1.2.3 Push kernel

1.2.4 Relabel kernel

1.2.5 Global relabel CPU

1.2.6 Overall graphcut algorithm

2. Result and evaluating

2.1 Result

2.2 Improved method

2.3 Result after using improved method