Writeup for Problem Set 4 of CMSC733

Jason Filippou

November 26, 2014

Contents

1 Deliverables

There are three subdirectories in this submission. At the top-level, we also provide the project description and this writeup. `images` contains all the different images used or produced. `papers` contains papers relevant to the problem of stereo matching with graph cuts or graph cuts in Computer Vision in general. Finally, `code` contains our code. `code` contains the precompiled maxflow libraries and wrappers for both Windows 64-bit and Linux 64-bit, and the top-level script `ps4.m` that runs the entire submission checks the running computer's architectire before inluding the appropriate directory in the path. This was necessary for us since we were changing from a Windows to a Linux computer very often during implementation of the assignment.

2 Data Term

2.1 Implementation

Our implementation of the data term follows the suggestions of the project description and can be found in MATLAB function file `unary_cost.m`. The input consists of both images, the maximum value of the disparity, as well as a value for "infinity" which we compute based on visual and algorithmic cues. The output is a three-dimensional array, where the first two dimensions correspond to image dimensions and there is one "slice" for every disparity value. The comments in the source file provide further information.

The interesting thing about our implementation is that we essentially disallow matchings between pixels that are in one image but are not in the other (all pixels for which $j - d < 0$). We do so by applying a cost of infinity to all those cells in the data term matrices. It is also important to mention that this unary cost computation cannot, by nature, sum over the smoothness costs of edges connecting pixels in $\mathcal{G}_{ab}$ with pixels outside $\mathcal{G}_{ab}$, simply because at the time that they are computed we haven't actually run a-b swap to generate $\mathcal{G}_{ab}$ yet. Incorporating these costs occurs later in the code, in function `ab_swap.m`.

2.2 Testing

We include a MATLAB script called `test_data_term.m` for testing purposes of the data term. The script contains two different tests that are run on custom binary images. Refer to the comments on the script for further information.

3 Smoothness Term

3.1 Implementation

Our implementation of the smoothness term is contained in the MATLAB function `potts_cost.m`. We once again follow the guidelines contained in the description. Our inputs consist of the left image, the Potts model parameter K and our "infinity" value for disallowing impossible matches. The output is a 3D matrix with 4 image-sized slices. The first slice contains the downward edge costs, the second one the upward ones, the third one the rightward ones, and the fourth one the leftward ones. Our aim is to provide a straightforward mapping between the slices generated by `potts_cost.m` and the edge sets generated by the routine `edges4connected`, included in the MATLAB wrapper which we downloaded.

3.2 Testing

Similarly to the data term, we include an assertions-based script that tests `potts_cost.m`. This script is called `test_smoothness_term.m` and can be run to verify that everything is working correctly.

4 Initialization

To initialize our disparities, we follow the guidelines and initialize based on a run of stereo matching without considering the smoothness terms. MATLAB provides for a very elegant one-liner solution based on `min`, which we have included in file `initialize.m`.

5 Full stereo

Our full stereo method is implemented in MATLAB function `stereo.m`. It takes the two images, L and R as input, and generates the computed disparity map. The method performs the following tasks: sets algorithm parameters ($MAXD$, K, infinity), computes the data and smoothness terms, initializes the disparity and runs a-b swap by calling the function `ab_swap.m`. The latter function is where the meat of the algorithm lies, and contains many comments and helper functions.

6 Running the code / Results

We provide a top-level MATLAB script called `ps4.m` which can be used to run the code. The user can simply run that particular script and everything takes care of itself.

Unfortunately, our results are not particularly good. We are making some kind of mistake in the optimization procedure, since every run of maxflow produces labelings that are 100% balanced over the two possible disparity labels a and b. Our intuition is that the smoothness term is to blame. We have run out of time to find the cause of this error and opted to not wait further because of other Vision-related obligations. Our results can be seen in figure 1.

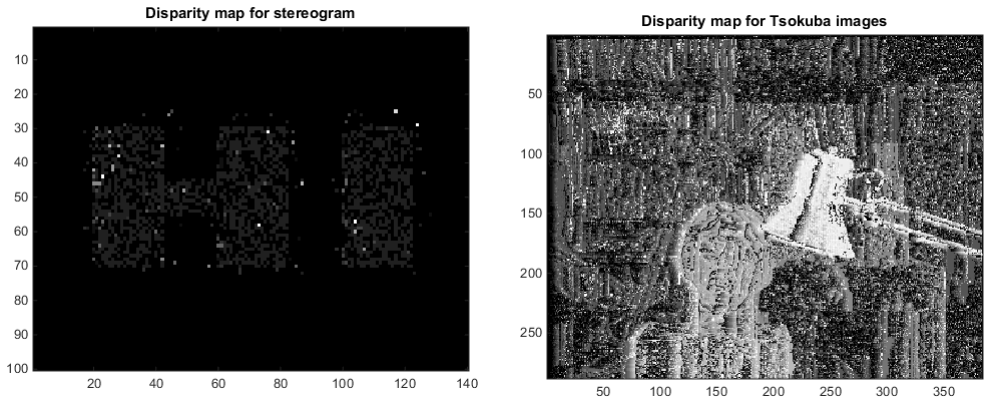

Figure 1: Results of a-b swap.