1 CudaMultilabelOptimization Software

The code can be used to a) solve multi-label optimization problems based on the Potts model by means of continuous optimization and the primal-dual algorithm, b) compute a powerful data term for interactive segmentation.

2 Corresponding Papers and Citations

This code implements the following papers:

1 C. Nieuwenhuis and D. Cremers, Spatially Varying Color Distributions for Interactive Multi-Label Segmentation, Transactions on Pattern Analysis and Machine Intelligence, 2013

2 C. Nieuwenhuis and E. Toeppe and D. Cremers, A Survey and Comparison of Discrete and Continuous Multi-label Approaches for the Potts Model, International Journal of Computer Vision, 2013

3 C. Zach, D. Gallup, J. Frahm and M. Niethammer, Fast global labeling for realtime stereo using multiple plane sweeps, Vision, Modeling and Visualization Workshop (VMV), 2008

4 A. Chambolle, D. Cremers, T. Pock, A Convex Approach to Minimal Partitions, SIAM Journal on Imaging Sciences, 2012

5 T. Pock, D. Cremers, H. Bischof, A. Chambolle, An Algorithm for Minimizing the Piecewise Smooth Mumford-Shah Functional, ICCV 2009

If you use this software for research purposes, YOU MUST CITE the corresponding papers in any resulting publication.
The optimization routines are based on the algorithms given in [2] in formula (8) (which is based on the relaxation in [3] and the primal-dual algorithm in [5]) and formula (11) (which is based on the relaxation in [4] formulated according to (9) and (10) in [2] and based on the primal-dual algorithm in [5]).

3 License

4 How to Compile

The code is based on a linux system and cuda 5.0 and requires a GPU for executing. It builds on the cimg library for image handling. The file QMakeFile.pro can be used to create a Makefile. Adapt the CUDA_DIR path and the INCLUDEPATH to your cuda and cudaSDK directory. The Makefile creates an executable called cudaMultilabelOptimization.

5 How to Execute the Program

To specify parameters adapt the file parameters.txt (lines starting with are ignored by the program). Then run the executable cudaMultilabelOptimization. You can either run the code interactively in case no scribble file is specified in the parameter file, or you can load a saved scribble file by specifying the scribble file in the parameter file. If no scribble file is specified the program runs interactively letting you draw scribbles on the image and inspect the segmentation result.

5.1 Drawing Scribbles

To draw a scribble use the mouse by left-clicking on the image for the start of the scribble and letting go at the end of the scribble. To assign this scribble to a label press the corresponding digit (between 0 and 9) on the keyboard. Only then the scribble will be drawn onto the image and the segmentation result be computed.

5.2 Parameters

The following parameters can be specified in the parameters.txt file. Lines starting with # are regarded as comments by the program.

General Parameters

- IMAGEFILE: can be all image formats handled by cimg, images must be normalized to the range [0,255]
- SCRIBBLEFILE: (optional) file containing scribbles in the format -1: pixels not contained in a scribble 0 - n: pixels contained in regions 0 to n. If no scribble file is indicated then the program will run interactively, otherwise the scribbles are read from the indicated file
- RESULTSFOLDER: folder so save all results to, i.e. the computed data term, the scribble file and the optimization result

Data Term Parameters

- COLORVARIANCE: variance of Gaussian for color component in Parzen density, rho in the paper, we normalize images to the range [0,..,255]
- SCRIBBLEDISTANCEFACTOR: variance of spatial component in the Parzen density is multiplied by this factor, alpha, in the paper, must be 5 or larger for images of range [0,..,255] to avoid artifacts due to spatial variances close to 0, set to 1000 for turning spatial component off and having purely color based likelihood
- BRUSHSIZE: determines the size of brush for drawing scribbles
- BRUSHDENSITY:]0,1] determines the ratio of points in the scribble that are actually used to compute the data term to save run time and memory, 1 means that all points are used having a dense scribble, small ratios indicate scribbles of low density

Optimization Parameters

- OPTIMIZATIONMETHOD: "zach" is based on [2,3], "chambolle" is based on [2,4]
- SMOOTHNESSWEIGHT: weighting for regularizer for balancing data term and regularization
- NUMSTEPS: number of iterations used in the optimization procedure
- DEBUGOUTPUT: "true" prints debug information on the screen during optimization "false" does not print debug information
- OUTPUTEVERYNSTEPS: step interval for printing debug information

6 Code Structure

The main function is defined in main.cpp.
The code defines the following two classes: imageSegmentation and dataterm.

ImageSegmentation Class The class imageSegmentation handles the general segmentation and optimization process. The function "executeInteractive" performs a loop consisting of 1. adding scribbles to the image by the user followed by 2. optimization of the segmentation problem by calling the "executeAutomatic" function from within the loop. The function "executeAutomatic" carries out optimization based on a given scribble file (either loaded from a file or created during the interactive execution). The parameters of both functions are explained above each function in the imageSegmentation.cpp file. The actual primal-dual optimization routines are implemented in the cudaOptimization.cu file. They both optimize the energy stated in (4) in [2] using different relaxations of the same functional. The method "zachPrimalDual" in the cudaOptimization.cu file implements the algorithm given in formula (8) in [2] based on the relaxation by Zach et al. in [3]. The method "chambollePrimalDual" implements the algorithm given in formula (11) in [2], which is based on the relaxation by Chambolle et al. in [4] but uses the formulation stated in (9) and (10) in [2].

Dataterm Class The dataterm class implements the spatially varying color distribution data terms in [1]. The function "readScribbleFromMap" reads scribbles from a scribbleMap (-1 for unassigned pixels, 0 to MAXNREGIONS for assigned pixels) and stores them for data term computation. The function "computeDataEnergy" then computes the data term based on this scribble information. The actual computation of the data term is implemented in the file cudaDataterm.cu.

Basic Data Structures All data structures are built on the cimg image structure. Energies are given as images consisting of four dimensions. energy(x, y, z, v): x,y spatial coordinates, z not used, i.e. 0, v channels correspond to region labels