Computer Vision II - Homework Assignment 4

Stefan Roth, Tobias Plötz, Uwe Schmidt, GRIS, TU Darmstadt

June 18, 2014

This homework is due on July 1, 2014 at 20:00.

Goals.

This assignment will deepen your understanding of inference on Markov random fields using graph cuts and α-expansion for multi-labeling problems.

Please note:

- Since your code needs to be fast, points will be subtracted for unnecessary usage of loops. Please use matrix/vector operations whenever possible.
- Please include one file/sheet with your written solutions. Please also write *all* comments or observations into this file/sheet.

Problem 1 - Iterated Graph Cuts - 35 points

As a first application of graph cuts, we want to implement a simplified version of the GrabCut algorithm [RKB04] for figure-background segmentation. The inputs for this algorithm are an image as well as a bounding box annotation drawn around the object. The output is a binary label for each pixel either classifying it as foreground or background. Pixels outside the bounding box are fixed to be in the background. With I, Θ and S denoting the image, the parameters of the color distribution and the segmentation, respectively, the segmentation energy for pixels inside the bounding box is given by

$$E(S \mid I, \Theta) = E_d(S \mid I, \Theta) + \lambda E_s(S \mid I) \tag{1}$$

where E_d and E_s are the data term and the smoothness term, respectively. The data term

$$E_d(S \mid I, \Theta) = \sum_i -\log \mathrm{GMM}(I_i \mid \theta_{S_i}) \tag{2}$$

measures how well each pixel fits to the color distribution of foreground or background pixels (depending on the segmentation). Color distributions are represented as Gaussian Mixture Models with 5 components. $\mathrm{GMM}(\cdot \mid \theta)$ denotes the probability density of a Gaussian mixture with parameters θ.

The spatial smoothness term favors segmentations that align to image boundaries by using contrast sensitive Pott's potentials on a 4-connected MRF.

$$E_s(S \mid I) = \sum_{(i,j)\in\mathcal{N}} \exp(-\beta \|I_i - I_j\|_2^2) \cdot \delta_{S_i \neq S_j} \tag{3}$$

The iterated graph cut algorithm that we will implement here now proceeds by initializing the color distributions from user annotation (e. g. a bounding box) and then iterating between minimizing the segmentation energy and re-fitting the color distributions. In the original GrabCut paper, a joint energy over S and Θ is defined, but we will ignore this for simplicity.

Tasks:

- Download and extract the file `asgn4_data.zip`. Follow the instructions in `README.TXT` to install the graph cuts optimization framework. Should you have problems compiling `MEX`-files, look into the help section for building and troubleshooting `MEX`-files. Run `GCO_UnitTest` to test your installation, save the output to `GCO_Check.txt` and attach it to your solution.

  1 points

- Implement the function

  ```
  Ed = edges4connected( height, width )
  ```

  which creates edges for a four-connected grid graph, *i.e.* a graph where each node is connected to its four neighbors. The result should be a $|E| \times 2$ Matrix, where $|E|$ is the total number of edges. Each row indicates the linear index of the two nodes which are connected by that edge.

  5 points

- Implement the function

  ```
  W = contrast_weight( I, Ed )
  ```

  which computes the weight term for the contrast sensitive Pott's potential. Set the missing parameter β as follows:

  $$\beta = \left(\frac{1}{|\texttt{Ed}|} \sum_{(i,j)\in\texttt{Ed}} \|I_i - I_j\|_2^2 \right)^{-1}$$

  `W` should be a column vector with the same number of rows as `Ed`.

  7 points

- Implement the function

  ```
  S = smoothness_term( n, Ed, W, lambda )
  ```

  which assembles the edge information, the accompanying contrast weights and the smoothness factor λ into a $n \times n$ matrix S representing the strength of the Pott's potentials. S has a non-zero entry in $S_{i,j}$ for each (i,j) in `Ed`. In your Matlab code, S should be a sparse matrix in order to save memory.

  2 points

- Implement the function

  ```
  [ fg_gmm, bg_gmm ] = fit_color_distribution( I, labels, k )
  ```

which estimates Gaussian mixture models `fg_gmm` and `bg_gmm` for foreground and background pixels, respectively. Use the built-in function `gmdistribution.fit` for approximate maximum likelihood estimation of the parameters. Here, `I` denotes a RGB color image that is segmented with the mask `labels` into foreground (label 1) and background (label 2). The number of mixture components is passed by the variable `k`. Consider setting a regularization parameter if you experience numerical instabilities while fitting the GMMs.

5 points

- Implement the function

  ```
  D = igc_data_term( I, fg_gmm, bg_gmm )
  ```

  that returns a $2 \times n$ matrix holding in each column the data term for each pixel. Here, n is the number of pixels in I.

  5 points

- Implement the function

  ```
  labels = iterated_graphcuts( I, window, lambda, k )
  ```

  that puts together the previously implemented components into the final segmentation algorithm. Here, I denotes an RGB color image on which a bounding box `window = [top, height, left, width]` is annotated. The pixels should be labeled as 1 if they belong to the foreground and 2 otherwise. Initialize all pixels within the bounding box as foreground and do 10 iterations of graph cuts and color distribution estimation. The final output should comprise labels for *all* pixels, so you may have to extend your label map before returning. Also `labels` should have the same number of rows and columns as I. Display the current segmentation estimate after each iteration.

  8 points

- Run the script `a4p1` that calls your iterated graph cuts algorithm with a test image and predefined bounding box and shows the result. Comment on the quality of your result. If you see artifacts, why might they be there? What could be done to improve the results?

  2 points

Notes:

- In order to foster modularity, each function is thought to use as little prior information about the parameters as possible. Hence all functions but `iterated_graphcuts` do not know about the bounding box.
- The color distributions should be fitted by using all pixels, i. e. the pixels inside as well as outside the boundary.
- If you have trouble with the GCO framework (compilation errors, parameter semantics, etc.) feel free to ask on Moodle and more importantly feel free to answer questions of others if you know a solution.
- You *must* adhere to the function signatures and semantics of inputs/outputs. Put up a question on Moodle if you have any doubts.
- Implement each function in its own file!

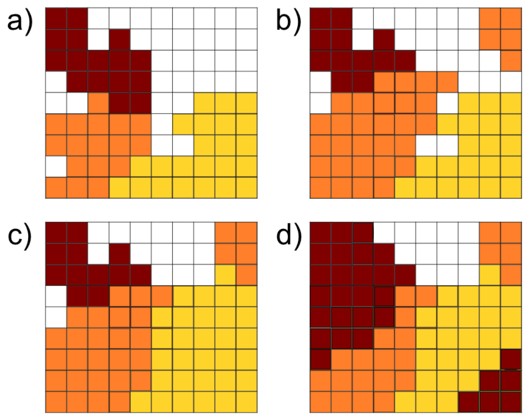

Figure 1: The alpha-expansion algorithm breaks the problem down into a series of binary sub-problems. At each step, we choose a label α and we expand: for each pixel we either leave the label as it is or replace it with α. This sub-problem is solved in such a way that it is guaranteed to decrease the multilabel cost function. a) Initial labeling. b) Orange label is expanded: each label stays the same or becomes orange. c) Yellow label is expanded. d) Red label is expanded. (source: [Prince2011])

Problem 2 - α-expansion - 15 points

Recall the α-expansion algorithm [BVZ01] presented in class. It provides a way of doing inference on a Markov random field with multiple labels by breaking the solution down into a series of binary problems, each of which can be solved exactly.

The main idea is to iterate through each label, where at each iteration all nodes are given the choice to either keep their current label or to switch to the new label (see Figure 1 for an illustration).

Here, we will implement an approximate version of the α-expansion algorithm.

Tasks:

- Implement the function

  ```
  labels = alphaexpansion(...)
  ```

  which implements the α-expansion algorithm and returns a labeling for a given set of labels. Use the GCO-optimization library to perform the binary graph cut at each iteration. Please refer to `README.TXT` for further instruction on its usage. You should use the Laplacian likelihood from assignment 1 for the unary potentials. Use the standard Potts model for the pairwise potentials, where a constant penalty is imposed on edges with different labels:

  $$E_{pq}(\alpha, \beta) = \begin{cases} 0, & \text{if } \alpha = \beta \\ \lambda > 0, & \text{otherwise.} \end{cases} \tag{4}$$

  9 points

- Test your implementation on the Tsukuba dataset and display the current labeling after each iteration. Please make sure your implementation of the α-expansion converges. A coarse structure is outlined in `stereo.m` and should serve as a starting point. Try starting from a uniform labeling (*e.g.* $\mathbf{x} \equiv 0$) and from a randomly generated disparity map. Comment on the differences between the two.

  3 points

- Compute and display the percentage of falsely labeled pixels with respect to the ground truth. Try different values for the smoothness parameter λ and report the best one.

  3 points

Hints:

- Do not forget to call `GCO_Delete()` after each expansion step to avoid memory leaks.
- Commands useful for this problem include `pause, reshape, find, sparse, addpath`.

References

[BVZ01] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 309–314, New York, NY, USA, 2004. ACM.