

User Guide for Release 1.1

Dec 2nd, 2011

Contents

1	Introduction	1
2	Usage Information	2
2.1	Procedure	2
2.2	About HOTT.m	3
2.3	Expected input	4
2.4	Software output	4
3	Technical Details	4
3.1	Our MRF energy formulation	4
3.1.1	Overview	4
3.1.2	Relevant Parameters	5
3.2	Optimization	6
3.2.1	Background	6
3.2.2	Relevant Parameters	6
3.3	Discretization of solution space	7
3.3.1	Overview	7
3.3.2	Relevant Parameters	7
3.4	Summary of other user-parameters	7
3.4.1	Trial-dependent parameters	7
3.4.2	Output files and related parameters	8
3.5	Suggestions on parameter-tuning	8
4	Copyright and Code Citation	9
5	Acknowledgments	9

1 Introduction

The code package downloaded from <http://mial.fas.sfu.ca/researchProject.php?s=400> provides a revised implementation of a method for tongue-tracking in 2D+time ultra-

sound (US) sequences as described in [5]. **Currently, it relies on MATLAB¹ and an executable file that can be built and executed on a Windows 32-bit machine.**

Package Contents

- `HOTTsetup.msi`: run this setup if MSVC is not available on the running platform; this will generate an executable file
- `HOTT.exe`: executable that performs energy minimization for the problem specified in an input file and outputs solution (D)
- `HOTT.m`: implements the pipeline of our method
- `createBSplineDeformations.m`: generates a label set of displacement vectors
- `sampDisplacements.m`: discretizes \mathbb{R}^2 ; used in `createBSplineDeformations.m`
- `get2Dnormal.m`: computes normal direction of each vertex of a contour
- `getNpoints.m`: resamples input contour to be of length N (N points)
- `myginput.m`: a simple revision of `ginput.m` that shows selected points on the screen
- `mhdread.m`: reads contents of `.mhd` files
- `mhdwrite.m`: for output to `.mhd` file
- `chkDim.m`, `dockf.m`: misc. files

Please email all inquiries to Lisa Tang at lisat@sfu.ca. If you find this software useful, please kindly cite our work [5] using the following BibTeX entry:

```
@INPROCEEDINGS{MMBIA2010,  
author = {Lisa Tang and Ghassan Hamarneh},  
title = {Graph-based tracking of the tongue  
contour in ultrasound sequences with adaptive temporal regularization},  
booktitle = {Mathematical Methods for Biomedical  
Image Analysis (MMBIA)},  
pages = {1-8},  
year={2010}  
}
```

2 Usage Information

2.1 Procedure

1. Download and unzip the files to `testdir`, a directory of your choice.

¹The m-scripts have only been tested on versions R2008a, R2010b, and R2011.

2. If Microsoft Visual Studio is available on your computer, try to execute `HOTT.exe`; otherwise, the provided `HOTT.exe` will likely need to be replaced. To replace it, run `HOTTsetup.msi` and follow the Setup Wizard.
3. Launch MATLAB
4. Change current directory to `testdir`
5. Perform pre-processing on each US frame:
 - Applying anisotropic-filtering, using e.g. [MATLAB code of \[3\]](#)
 - Denoise frames with, e.g. [MATLAB Toolbox²](#)
 - Crop each frame to the region of interest (as shown in Figure 1)
6. Edit `HOTT.m` (details in Section 2.2) to adjust user-defined parameters.
7. Execute `HOTT.m` to run the algorithm and obtain tongue segmentations.

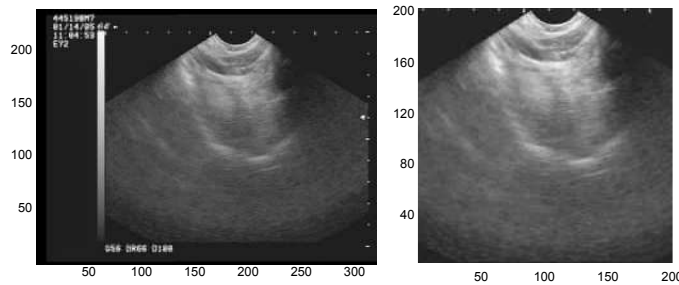


Figure 1: (Left) Original ultrasound frame. (Right) After cropping the frame to region of interest.

2.2 About `HOTT.m`

This MATLAB script implements the following pipeline:

1. Loads the first frame of the sequence to be segmented, displays it on screen, and asks for 5 input points from the user through which the template tongue contour C would be drawn. *Notes: the template contour does not need to be exact as the algorithm will adjust it to best fit the image data. Conversely, the length of the template contour will somewhat constrain the length of segmentation contours in all subsequent frames. Thus, when creating the template contour, try to extend the length of the contour as much as reasonable so to facilitate the segmentation of subsequent frames.*

² <http://www.medinfo.cs.ucy.ac.cy/index.php/downloads/toolboxes/10-matlab-software>

2. Discretizes the solution space (\mathbb{R}^2) based on user-defined parameters (c.f. Section 3.3) to generate a set of candidate solutions (aka label set) to an input file (`.bin`). The spatial coordinates of the template contour will also be written to this file.
3. Calls an external executable (`HOTT.exe`) that implements [5] to obtain a set of displacement vectors D from which we derive a set of segmentations for each frame.
4. Displays the segmentation result and generate other plots for evaluation purposes.

2.3 Expected input

Each 2D frame of a US sequence must be written to individual files in a single folder, e.g.

```
USsequence02/001.mhd,
USsequence02/002.mhd,
...
USsequence02/399.mhd
```

Note: `HOTT.m` assumes use of `.mhd` format; if other file formats are used (e.g. `.hdr`, `.tiff`, etc.), ensure that the image orientation is correct.

2.4 Software output

When `HOTT.m` is executed, it will generate an input file `.bin` to be read by the executable. When the executable is called, it will create some output files to be read by `HOTT.m`, which then uses to generate some visualizations, including the obtained segmentations as shown in Figure 2.4. For details on the various output, please see code in `HOTT.m`.

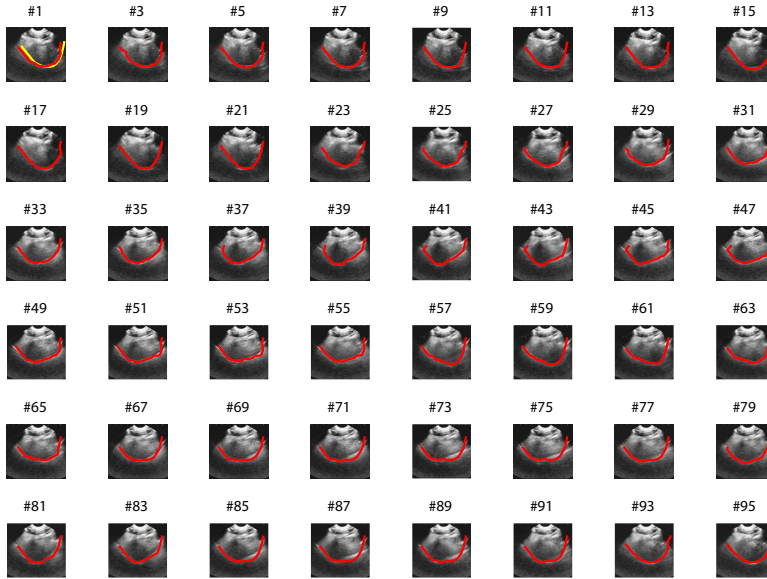
3 Technical Details

3.1 Our MRF energy formulation

3.1.1 Overview

Our method [5] formulates tracking as an MRF energy minimization problem. Our energy terms may be constructed with unary, pairwise, or ternary potentials; these include:

1. E_{data} : Data term that drives segmentation contours to the optimal positions based on image features; available implementations are:
 - (a) Image-gradient [unary] (as presented in [5])



Sample Output (template in yellow, obtained segmentations in red)

Figure 2: Obtained segmentations (red contours) on US frames. Note that the initial template contour does not need to be exact (shown in yellow on the first frame).

- (b) Profile-based data term [ternary]. This term is motivated by the observation that the intensity profiles normal to groundtruth segmentation contours may follow the shape of a Gaussian function. For each control point on a segmentation contour, the correlation between its 1D normal profile and a Gaussian kernel is estimated such that it returns a low penalty value when the estimated correlation is maximal.

2. Regularization terms:

- (a) $E_{temporal}$ [pairwise]: ensures temporal neighbours move coherently³
- (b) E_{length} [pairwise]: ensures length of segmentation contours is preserved
- (c) $E_{curvature}$ [ternary]: penalizes irregularities in segmentation contours
- (d) $E_{spatial}$ [pairwise]: ensures spatial neighbours move coherently; this term may replace E_{length} and $E_{curvature}$ (see first note in Section 3.1.2)

3.1.2 Relevant Parameters

Parameters for energy terms listed in 3.1.1 include:

³Neighboring vertices on a given contour are viewed as neighbours; see [5] for details of our graph-based formulation.

- `Gradient_DE_weight`: weight on gradient-based data energy term; suggested range [0,0.5]
- `Profile_DE_weight`: weight on profile-based (higher-order) data energy term; suggested range [0.5,1.25]
- `npix`: length of the intensity profiles to be drawn; suggested range [4,12]
- `Gaussian_ker_sigma`: sigma of Gaussian kernel; should be tuned to “sharpness of edges”; suggested range [0.1,0.45]
- `Corr_Thres`: correlation between Gaussian kernel 0.65
- `Temp_smthness_weight`: weight on $E_{temporal}$; suggested range [0,0.25]
- `Spat_smthness_weight`: weight on $E_{spatial}$; suggested range [0,0.05]
- `Preserve_length_weight`: weight on $E_{spatial2}$; suggested range [0,0.05]

Notes

- Not all regularization terms are necessary; this largely depends on specific segmentation problem at hand; i.e. $E_{spatial}$ plays similar roles of E_{length} and $E_{curvature}$ and may replace them if the expected segmentation contours do not deviate significantly over time.
- `Temp_smthness_weight` should be tuned in relation to `temp_res` (i.e. low or no temporal regularization if `temp_res` is large).
- E_{data} may be a sum of profile-based and gradient-based data energy terms; this is achieved by setting both `Gradient_DE_weight` and `Profile_DE_weight` to non-zero values.

3.2 Optimization

3.2.1 Background

We employ the higher-order clique reduction (HOCR) technique [1] and the Quadratic Pseudo Boolean Optimization (QPBO) algorithm of [2] to minimize our energy function.

QPBO operates by iteratively examining two sets of labelings, each of which may consist of arbitrary values. Given an initial labeling and an arbitrary (or “proposed”, e.g. use of B-spline deformations) labeling on a set of pixels, QPBO operates by choosing for each pixel to either retain its current label or adopt the proposed one in the arbitrary labeling such that the overall MRF energy is approximately minimized. At times, QPBO may produce a partial labeling such that some graph nodes are unlabeled, in which case, rules are designed to decide the values of unlabeled nodes.

Our energy minimization framework involves two nested loops; the inner loop involves iteratively comparing the current labeling with one candidate solution in the label set and the outer loop repeats this cycle again until the overall change in \mathbf{D} drops below an empirically determined convergence threshold (`convergence_thres`). We call the completion of one inner loop as one optimization cycle.

3.2.2 Relevant Parameters

- `improve`: 0 or 1; if 1, use the extension of [4] when dealing with partial labelings.
- `convergence_thres`: defines one condition of termination of optimization; when the average change in D is below the value of this parameter, the algorithm terminates.
- `nitns`: maximum number of optimization iterations allowed; its value is problem-specific; suggestion: start with `nitns=2`, then examine how solution changes after each optimization cycle to decide if more iterations are needed

3.3 Discretization of solution space

3.3.1 Overview

The discretization of \mathbb{R}^2 for the generation of the label set \mathcal{L} can be done in two ways:

1. Uniform sampling: Under this sampling scheme, each $l \in \mathcal{L}$ is a translation vector. When the entire template contour C is assigned to the same l , C is globally translated. This discretization approach is parameterized by the number of steps of along the radial axis and the number of steps along the angular axis.
2. B-spline sampling. Each $l \in \mathcal{L}$ is a *vector of displacements*, i.e. l is a $2 \times \text{npoints}$ array. These displacements are computed by displacing a subset of vertices of C along the directions normal to C .

3.3.2 Relevant Parameters

- `proposalType`: 0 = uniform sampling scheme; 1 = B-spline sampling scheme
- `max_dist`: the maximum expected magnitude of D
- `samplingFac`: parameterizes size of label set, given `max_dist`; default values provided in `HOTT.m`

We suggest users to use the B-spline sampling when the expected displacement is large, e.g. `max_dist > 20`.

3.4 Summary of other user-parameters

3.4.1 Trial-dependent parameters

- `start_fr`: the i -th frame on which the template contour was defined
- `end_fr`: the last frame to segment
- `temp_res`: temporal resolution of the 2D+T segmentation
- `npoints`: length of the template contour

3.4.2 Output files and related parameters

- `write_each`: 0 or 1; if 1, then write out D after each optimization cycle
- `debug`: {0, 1, 2, 3}; different levels of verbosity; if 3, shows misc. information (e.g. statistics on mean values of different penalties). Suggested usage: turn on to examine appropriateness of ranges of each penalty terms and adjust the weights outlined in Section 3.1 accordingly. As output to `std::cout` increases runtime, it should be turned off when `nframes > 50`.
- `resf`: file prefix of output generated by `HOTT.exe`; some of which include
 - `test_x.txt`: contains the x component of the final D
 - `test_y.txt`: contains the y component of the final D
 - `test_x0.txt`: contains the x component of D obtained after the 1st iteration
 - `test_y0.txt`: contains the y component of D obtained after the 1st iteration
 - `test_stats.txt`: contains values of critical parameters and statistics relating to success of energy minimization, e.g.:
 1. Energy before the “Improve” heuristics [4] (IMPROVE) is performed
 2. Energy after IMPROVE is performed
 3. Fraction of labels unlabeled before IMPROVE is performed
 4. Fraction of labels unlabeled after IMPROVE is performed
 - `test_template.txt`: saves the template contour and values of key parameters used

3.5 Suggestions on parameter-tuning

- Use the B-spline sampling scheme if large deformations are expected. Otherwise, start with uniform sampling scheme and switch to B-spline sampling scheme if optimization speed needs to be increased.
- Start with high weight on the data likelihood term, then decrease the weight and/or refine regularization weights based on inspection of results (e.g. increase regularization if contours are highly irregular)
- Solution usually does not change significantly in each optimization iteration, so one can start with minimal number of `nitns` (e.g. 1); once optimal parameters are found for a given problem, then refine results with more number of optimization iterations
- Optimality of parameters may not generalize (e.g. those that worked well for a case of 25 frames may not work well with a case of 100 frames, and vice-versa)

4 Copyright and Code Citation

This code is licensed under the GNU General Public License as published by the Free Software Foundation. A copy of this license is included with the software. If you found this software useful, please kindly cite our work [5] using the following BibTeX entry:

```
@INPROCEEDINGS{MMBIA2010,  
  author = {Lisa Tang and Ghassan Hamarneh},  
  title = {Graph-based tracking of the tongue  
  contour in ultrasound sequences with adaptive temporal regularization},  
  booktitle = {Mathematical Methods for Biomedical  
  Image Analysis (MMBIA)},  
  pages = {1-8},  
  year={2010}  
}
```

5 Acknowledgments

We sincerely thank authors of [1,2] for their software code.

References

- [1] H. Ishikawa. Higher-order clique reduction in binary graph cut. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 20–25, 2009. 6, 9
- [2] V. Kolmogorov and C. Rother. Minimizing non-submodular functions with graph cuts - a review. 29(7):1274–1279, 2007. 6, 9
- [3] P. Kovesi. Anisotropic diffusion smoothing. <http://www.csse.uwa.edu.au/pk/Research/MatlabFns/Spatial/anisodiff.m>. 3
- [4] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing Binary MRFs via Extended Roof Duality. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007. 6, 8
- [5] L. Tang and G. Hamarneh. Graph-based tracking of the tongue contour in ultrasound sequences with adaptive temporal regularization. In *Mathematical Methods for Biomedical Image Analysis (MMBIA)*, pages 1–8, 2010. 2, 4, 5, 9