OPEN PLATFORM FOR ULTRASOUND LOCALIZATION MICROSCOPY: PERFORMANCE ASSESSMENT OF LOCALIZATION ALGORITHMS

AUTHORS: Arthur Chavignon, Baptiste Heiles, Vincent Hingot. CNRS, Sorbonne Universite, INSERM.
Directed by: Olivier Couture, Research Director CNRS, Sorbonne Universite, INSERM
Laboratoire d'Imagerie Biomedicale, Team PPM. 15 rue de l'Ecole de Medecine, 75006, Paris
Code Available under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (see https://creativecommons.org/licenses/by-nc-sa/4.0/)

ACADEMIC REFERENCES TO BE CITED

Details of the code in the article by Heiles, Chavignon, Hingot, Lopez, Teston and Couture.
Performance benchmarking of microbubble-localization algorithms for ultrasound localization microscopy, Nature Biomedical Engineering, 2021, in press.

General description of super-resolution in: Couture et al., *Ultrasound localization microscopy and super-resolution: A state of the art*, IEEE UFFC 2018.

CORRESPONDING AUTHORS

- Article: Baptiste Heiles (baptiste.heiles@gmail.com)
- Materials, scripts, and codes: Arthur Chavignon (arthur.chavignon.pro@gmail.com)
- Collaborations, rights and others: Olivier Couture (olivier.couture@sorbonne-universite.fr)

ABSTRACT

Ultrasound Localization Microscopy (ULM) is an ultrasound imaging technique that relies on the acoustic response of sub-wavelength ultrasound scatterers to map the microcirculation with an order of magnitude increase in resolution. Initially demonstrated in vitro, this technique has matured and sees implementation in vivo for vascular imaging of organs, and tumors in both animal models and humans. The performance of the localization algorithm greatly defines the quality of vascular mapping. We compiled and implemented a collection of ultrasound localization algorithms and devised three datasets in silico and in vivo to compare their performance through 18 metrics. We also present two novel algorithms designed to increase speed and performance. By openly providing a complete package to perform ULM with the algorithms, the datasets used, and the metrics, we aim to give researchers a tool to identify the optimal localization algorithm for their usage, benchmark their software and enhance the overall image quality in the field while uncovering its limits.

This article provides all materials and post-processing scripts and functions.

1. PATH AND LOCATIONS

Before running scripts, two paths are required and have to be set in `PALA_SetUpPaths.m` to your computer environment:

- `PALA_addons_folder` : the addons folder with all dedicated functions for PALA
- `PALA_data_folder` : root path of your data folder

2. EXAMPLE SCRIPT

Script name `/PALA_scripts/PALA_InVivoULM_example.m`
We provide a simple framework for ultrasound localization microscopy. We encourage the community to try it on various datasets of any kind of organs. Data are loaded, microbubbles are detected, localized and paired to generate a list of trajectories. The script generates 4 final images:

- Image density based on microbubbles counts: pixel intensity codes the number of microbubbles crossing this pixel
- Image density with axial color encoding: pixel intensity codes the number of microbubbles crossing upward/downward
- Velocity magnitude image: pixel intensity represents the average bubble velocity in *mm/s*
- PowerDoppler image for comparison

3. MAIN SCRIPTS

For each major datasets (*In Silico PSF*, *In Silico Flow*, and *In Vivo Brain*), scripts are separated in two parts: one for processing and one for displaying. After computation, the processing script will launch displaying script. All scripts are located in `/PALA_scripts/` folder.

- Data processing scripts performing detection, localization (and tracking): (`PALA_SilicoPSF.m`, `PALA_SilicoFlow.m` , `PALA_VivoBrain.m`)
- Displaying scripts for results analysis: (`PALA_SilicoPSF_fig.m`, `PALA_SilicoFlow_fig.m`, `PALA_VivoBrain_fig.m`)

After running the 3 main scirpts, the gobal score can be computed and display with `PALA_GlobalScore.m` . This script loads scores from the 3 major datasets and computes the global score.

For the 3 supplementary in vivo datasets (*RatBrainBolus*, *MouseTumor*, and *RatKidney*), we added a routine script (`PALA_VivoMulti.m`) with few input parameters. The user can select each datasets by changing the value of `DataSetNumber` .

4. TOOLBOX FOR ULTRASOUND LOCALIZATION MICROSCOPY

The 3 main functions required for ULM processing are provided in the folder `/PALA_addons/ULM_toolbox/`

- `ULM_localization2D.m` : localizing microbubbles in a stack of images
- `ULM_tracking2D.m` : pairing a list of microbubbles into a set of tracks

- `ULM_Track2MatOut.m` : generates a density image (`MatOut`) by accumulating tracks occurrences in a pixel grid

5. PALA DEDICATED FUNCTIONS

Other functions, dedicated to performance assessments and metrics are localized in the folder `/PALA_addons/PALA_functions/` . It contains a list of functions embedding localization, tracking, and analyzing functions in order to build all metrics presented in this article. One can easily add a new localization kernel and keep all the comparison framework provided by this work.

6. LOCALIZATION ALGORITHMS

In this article we compared 7 different localization algorithms:

- *no-shift* : no localization, bubble taken in the center of the max intensity pixel
- *weighted average* : weighted average localization base of intensities
- *interpolation based scheme* : using cubic, Lanczos or spline kernels
- *Gaussian fit* : using a minimization algorithm and with a Gaussian kernel
- *radial symmetry* : based on the minimization of the intensity gradient

You can easily add your custom localization kernel in the function `ULM_localization2D` (and `ULM_localization2D_mesh`) by adding a new `LocMethod` .
Your custom function must take a square image (5x5 or 7x7 pixels) centered on the bubble, and return the localization of the bubble in the pixel grid. If the bubble is located at pixel {3; 3} at the top left-hand corner, the function must return something like {3.5; 3.3}.

7. REQUIREMENTS

Specific functions have been added to the toolbox but can be downloaded from MathWorks Community :

- `simpletracker` : Jean-Yves Tinevez (2020). simpletracker (https://www.github.com/tinevez/simpletracker], GitHub. Retrieved April 22, 2020. (https://fr.mathworks.com/matlabcentral/fileexchange/34040-simpletracker)
- `munkres.m` : by Yi Cao at Cranfield University on 17th June 2008 (http://csclab.murraystate.edu/bob.pilgrim/445/munkres.html] (https://fr.mathworks.com/matlabcentral/fileexchange/20328-munkres-assignment-algorithm)
- `tight_subplot.m` : Pekka Kumpulainen (2020). `tight_subplot(Nh, Nw, gap, marg_h, marg_w)` (https://www.mathworks.com/matlabcentral/fileexchange/27991-tight_subplot-nh-nw-gap-marg_h-marg_w), MATLAB Central File Exchange. Retrieved April 23, 2020

A few MATLAB toolboxes are required to run the code: *Communications*, *Bioinformatics*, *Image Processing*, *Curve Fitting*, *Signal Processing*, *Statistics and Machine Learning*, *Parallel Computing*, *Computer Vision Toolbox*.
Codes have been run with MATLAB 2019a, and 2017b for Vantage 4.0.0 Verasonics simulations. Please refer to Verasonics Vantage documentation for requirements specific to the Verasonics Vantage Ultrasound Platform.

8. DISCLAIMER