**CoverBLIP MATLAB toolbox for accelerated Magnetic Resonance Fingerprint reconstruction**

The CoverBLIP algorithm uses fast ANN searches based on cover trees data structure in order to accelerate iterative matched-filtering for solving Magnetic Resonance Fingerprinting (MRF) inverse problem. The idea of using cover tree searches for the general class of data-driven compressed sensing problems was originally presented in:

*M Golbabaee, ME Davies, “*[*Inexact Gradient Projection and Fast Data Driven Compressed Sensing*](javascript:void(0))*”, IEEE Transactions on Information Theory, 2018 (doi:*[*10.1109/TIT.2018.2841379*](https://doi.org/10.1109/TIT.2018.2841379)*)*

The customized application of this framework to the MRF reconstruction problem (i.e. the CoverBLIP algorithm) appeared in the following articles:

*M Golbabaee, Z Chen, Y Wiaux, M Davies, “CoverBLIP: accelerated and scalable iterative matched-filtering for Magnetic Resonance Fingerprint reconstruction”, arxiv, 2018. (Extended version)*

*M Golbabaee, Z Chen, Y Wiaux, M Davies, “*[*Cover tree compressed sensing for fast MR fingerprint recovery*](javascript:void(0))*”, IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP), 2017, (doi:*[*10.1109/MLSP.2017.8168167*](https://doi.org/10.1109/MLSP.2017.8168167)*).*

If you find this toolbox useful in your research, please cite the relevant article(s) from above.

*Thnx!*

© Mohammad Golbabaee, Zhouye Chen (2018)

**To run:**

reconstruction\_demo.m

The main demo file comparing MRF reconstruction using three methods: 1) TM: non-iterative template matching [Ma et al., Nature’13], 2) BLIP: iterative matched-filtering with brute-force searches [Davies et al., SIAM’14], and 3) CoverBLIP iterative algorithm based on cover tree ANN searches.

We use in this simulation: one slice of the segmented Brainweb digital phantom. The IR-BSSFP fingerprint dictionary with a set of pseudo-random excitations. A Multi-shot Echo Planar Imaging (EPI) Cartesian k-space acquisition.

Note (1) that before running this demo you need to compile our Cover tree MATLAB wrapper located in the folder covertree\_matlab\_wrapper (please follow the instructions in the next section).

Note (2) that if you run this demo for the first time, the dictionary and MRF phantom construction steps might take a little while.

Note (3) that the current demo allows for using low-rank subspace reconstruction based on dominant SVD components of the MRF dictionary, for all tested algorithms [see e.g. McGivney et al. MRM’14 for the TM algorithm]. The subspace dimension is chosen by the variable input.svd\_k which for the IR-BSSFP dictionary used in this experiment is recommended to be set 200 (dictionaries encoding more complex/non-linear dynamics may require higher values!).

generate\_dict\_phantom.m

This function generates: i) the IR-BSSFP dictionary (TR=10 msec, pseudo-random flip angles in FAnTR\_bSSFP.mat) calling brain\_dict\_true.m, and ii) the synthetic MRF data from the segmented Brainweb phantom (calling brain\_phantom.m) with ground truth T1, T2, off-resonance frequency and proton density parameter maps. After running the demo for the first time, data will be stored in the subfolder .\data\Bloch for future experiments.

sampling\_mask\_FASTsingle.m

Creates the forward and adjoint (back-projection) operators used for the gradient updates in e.g. the iterative recovery methods. Since we use the Multi-shot EPI Cartesian acquisition, these operators are based on FFT/iFFT transforms. In real-data applications one has to modify this operator to additionally account for multi-coil setups (or other sampling schemes such as spiral read-outs which requires using non-uniform Fourier transform).

algo\_IPA.m

Iterative projection algorithm for MRF reconstruction. Based on the chosen type of dictionary search (exact/inexact), it translates to BLIP, TM (one iteration of BLIP) or the CoverBLIP algorithm.

find\_nearest\_MRF\_prox\_ct/ find\_nearest\_MRF\_prox\_mat

Dictionary matching using cover tree’s ANN or the exact brute-force searches, correspondingly. The level of approximation in cover tree’s ANN is set by the variable param.epsilon.

**MATLAB wrapper for Cover trees Approximate Nearest Neighbor (ANN) search**

We present a MATLAB wrapper for the (1+ ε)-ANN search with Cover tree data structure.

**Compatibility Notes**

The tool was developed in its current form on MATLAB R2016b. You need a C++ compiler that MATLAB can use with its mex command. Tested on these systems:

-- LINUX KDE/Ubuntu, 64-bit

-- MAC OS X, 64-bit

**How to install**

1. Add the directory to the MATLAB path.
2. You need a C++ compiler. You can configure MATLAB to use a compatible C++ compiler by typing mex -setup at the MATLAB prompt.
3. Run the compile\_covertree\_code.m file.

Note: If the MATLAB error about static TLS happens, please use the command in the terminal before opening the MATLAB.

export LD\_PRELOAD=/usr/lib/gcc/x86\_64-linux-gun/4.8/libgomp.so

**How to use**

* Generate the cover tree structure based on the dictionary variable dict:

QT = mexCovertree((single(dict)));

Where QT represents the object of the cover tree structure.

* Output the cover tree structure:

tree\_structure = QT.eNN\_tree\_structure();

Where tree\_structure is a matrix representing the tree structure.

* Parallel ANN searches for multiple queries with approximation level of epsilon (independent search for each query, but many queries in parallel):

[Label, D\_atom, dist, counter] = QT.eNN\_Loop\_Current(queries, epsilon, upperbound);

Where the inputs queries, epsilon, upperbound represent the queries, the approximation level and the upper bound of the distances between queries and atoms in the dictionary respectively. When initialization is used, this upperbound corresponds to the distance between a query and the corresponding initialized ANN solution. If this distance happen to be small, the tree search benefits (accelerates) due to eliminating many top level nodes and their branches.

The outputs Label, D\_atom, dist, counter represent the label (index) of the found ANN atom from the dictionary, the content of the atom, the corresponding distance between the query and the atom, and the number of pair-wise distance calculations during the search (computational complexity), respectively.

(Note that this method is currently used in our CoverBLIP demo)

* Batch ANN searches for multiple queries with approximation level of epsilon (this approach constructs a dual-tree on a query batch to exploit correlations between queries):

[Label, D\_atom, dist, counter] = QT. eNN\_BSearch(queries, epsilon, batch\_size);

Where the inputs include queries, epsilon, batch\_size represent the queries, the approximation error bound and the size of the batch respectively. The outputs include Label, D\_atom, dist, counter, index represent the label of the found ANN atom, the content of the atom, distance between the query and the atom, the number of pair-wise distance calculations (search complexity) during the search and the original index of the queries as an input batch respectively.

* Multiple queries ANN search with stopping condition at a certain level of the tree:

[Label, D\_atom, dist, counter] = QT.eNN\_BSearch(queries, epsilon, stop\_level);

Where the inputs include queries, epsilon, stop\_level represent the queries, the approximation error bound and the level that stops searching respectively. The outputs include Label, D\_atom, dist, counter represent the label of the found ANN atom, the content of the atom, the corresponding distance between the query and the atom, and the number of distance calculations during the search respectively.

The search will stop at stop\_level of the cover tree structure even if the best ANN is still not found.

* Delete the cover tree object

delete(QT);

**Acknowledgements**

Our MATLAB cover tree wrapper/implementation is based on the following papers:

1. *Alina Beygelzimer, Sham Kakade, and John Langford. "Cover trees for nearest neighbor." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.*
2. *Mike Izbicki and Christian Shelton. "Faster cover trees." Proceedings of the 32nd International Conference on Machine Learning (ICML-15). 2015.*

and the code from the authors: [*http://hunch.net/~jl/projects/cover\_tree/cover\_tree.html*](http://hunch.net/~jl/projects/cover_tree/cover_tree.html)*.*

We also acknowledge accessing the following files on MATLAB Central File Exchange: [*http://uk.mathworks.com/matlabcentral/fileexchange/45020-matlab-class-wrapper-for-a-c++-implementation-of-a-quadtree*](http://uk.mathworks.com/matlabcentral/fileexchange/45020-matlab-class-wrapper-for-a-c++-implementation-of-a-quadtree)*.*