GrasP: A Matlab Toolbox for Graph Signal Processing

Benjamin Girault\textsuperscript{*}, Shrikanth S. Narayanan\textsuperscript{\textdagger}, Antonio Ortega\textsuperscript{\textdagger}, Paulo Gonçalves\textsuperscript{\textdagger}, Éric Fleury\textsuperscript{*}

\textsuperscript{*} Université de Lyon, Inria, ENS de Lyon, CNRS, UCB Lyon 1, 69342, Lyon, FRANCE
\textsuperscript{\textdagger} Signal and Image Processing Institute, University of Southern California, CA 90089 Los Angeles, USA

benjamin.girault@usc.edu, {shri, ortega}@sipi.usc.edu, {paulo.gonalves, eric.fleury}@ens-lyon.fr

Objectives

- Matlab toolbox to create, process and analyze graph signals.
- Framework for the community to easily share and experiment.

Initialization

\begin{itemize}
\item Install the toolbox (one time operation):
\begin{verbatim}
grasp_install;
\end{verbatim}
\end{itemize}

Figure 1: GraSP initialization.

Plotting Graphs and Graph Signals

\begin{itemize}
\item Build a graph and a graph signal:
\begin{verbatim}
\texttt{g = grasp_planRnd(100);
\texttt{A = grasp_adjacency_threshold(g, 0.01);
\end{verbatim}
\end{itemize}

Figure 2: Matlab code to generate Figure 3.

Figure 3: A random sensor network depicted using Matlab (left) and LaTeX(right).

Interfacing with \LaTeX

\begin{itemize}
\item Export a graph and two graph signals:
\begin{verbatim}
grasp_exportcsv(g, 'nodes.csv', 'edges.csv');
grasp_exportcsv_signal(g, x_2, 'heat.csv');
grasp_exportcsv_signal(g, y_hat, 'wn.csv');
\end{verbatim}
\end{itemize}

Figure 6: Matlab code to export a graph and two graph signals.

Figure 7: \LaTeX code for Figure 3.

Remark 1. The package \texttt{tikzgraph} also implements the command \texttt{backgroundflatgraph} allowing to have a background image similarly to the Matlab output of Figure 5, and \texttt{backgroundimage} to plot the signal using stems. In addition, the package \texttt{tikzmatrix} implements the command \texttt{tikzmatrix} to plot a matrix.

Interfacing with External Toolboxes

\begin{itemize}
\item Interfacing with Matlab
\begin{verbatim}
list(cur_dep).ref_bib = 'https://arxiv.org/abs/1408.5781';
list(cur_dep).optional = 1;
list(cur_dep).start_script = 'gsp_start';
list(cur_dep).root_dir = 'gspbox/';
list(cur_dep).name = 'gspbox/';
list(cur_dep).url = ['https://github.com/epfl-lts2/gspbox/','releases/download/0.6.0/gspbox-0.6.0.zip'];
grasp_gft_gui(g, x);
grasp_gft_gui(g, y);
grasp_gft_gui(g, x_hat);
grasp_gft_gui(g, y_hat);
\end{verbatim}
\end{itemize}

Figure 8: Excerpt of \texttt{grasp_dependencies.list} showing how the GSPbox \cite{GSPbox} is included in GraSP.

Figure 9: A Graphical User Interface using Matlab.

Raw list of functionalities

- Graph Structure
- Classical graphs
- Adjacency from distance matrix
- Laplacian matrices
- GFT matrix computation
- Perform the GFT on signals and operators

Main differences with the GSPbox \cite{GSPbox}

- GSPPbox: Graph cluster plotting
- GraSP: interface with GraphVIZ
- Optimized for GUIs

Figure 10: Example code to use the GUIs of Figures 11 and 12.

Figure 11: A Graphical User Interface optimized to iterate an operator (here a low pass filter) on a signal (here a WN) and animate this iteration.

Figure 12: Another GUI displaying a graph signal on a graph (top left), on the vertex axis (bottom left), and the associated GFT (top right). The last two figures can be used to interactively change the graph signal, or its GFT. Ideal low-pass and high-pass filters are available (bottom right).

Conclusion and Future Work

- Matlab toolbox implementing many useful functions to generate, manipulate and display graphs and graph signals.
- Efficient implementation of external (optional) dependencies without including the code in the archive.
- Future work: Use the Matlab graph object instead of a custom graph structure, and backport this object definition for older version of Matlab and GNU Octave (WIP).
- Future work: Merge the GSPPbox and GraSPbox.

References