Maximum Chirplet Transform Code

These notes complement the Maximum Chirplet Transform Matlab code written by Fabien Millioz and Mike Davies, last updated 2016. This is a software implementation of the Maximum Chirplet Transform and its application to the detection of FMCW signals (piecewise linear chirps). Theory and details of the technique can be found in:

F. Millioz and M. E. Davies, "Sparse detection in the chirplet transform: application to FMCW radar signals," IEEE Transactions on Sig. Proc. vol 60(6), pp 2800 – 2813, 2012.

Please reference this paper in any publication that makes use of this software. Copyright, University of Edinburgh 2016. Details of the code functions are given below along with a description of the important variables and recommended settings for key parameters. Appendix A contains additional notes on the code. The README file is reproduced for convenience in appendix B.

Main Function: chirpgathering

`chirpgathering.m` – This function takes the input signal and applies the chirplet transform to the data generating a frequency-time-chirprate (f-t-c) representation. Such a representation is redundant and the next step is to determine a sparse chirplet decomposition of the signal using minimal computation. This is done through the `detectionWinmax` function.

The detection phase is a `light' version of Matching Pursuit. We detect one chirplet at a time (the largest) and then subtract a conservative estimate of its influence on this detection of subsequent chirplets. Subtraction of the chirplet from the original signal would be costly so, instead, we subtract an upper bound (`winmax`) from the Maximum Chirplet Transform (MCT) of the signal which in turn is the maximum (over the chirprate) of the absolute value of the chirplet transform. The `winmax` represents a conservative approximation of the influence of chirplets at one T-F point on neighbouring points. Sequential detection is performed until there are no significant MCT values above the noise floor. The noise level is determined through a spectral kurtosis based method.

The function is composed of a small number of subfunctions detailed below:

Chirplet Transform

The chirplet transform is computed in `chirpletc.m` (or for real signals `chirplet.m` can be used). The output is a Frequency-Time-Chirprate Tensor. This indexing order is chosen so that when vectorizing a Frequency-Time representation in Matlab the frequencies remain together in frames.

Maxwindow

The algorithm creates the `maxwindow` function used in `detectionWinmax.m`. This is used to remove the energy leakage (correlation) into neighbouring TF bins from any detected chirplet. See [Millioz and Davies 12] for details.

DetectionWinmax

The function estimates the noise level using the spectral kurtosis [Millioz and Martin, ICASSP, 2010] of the signal's spectrogram (the chirplet transform section with zero chirprate). It then sequentially

detects significant chirplets and masks out the leaked energy. The algorithm terminates once there are no more significant chirplets to be detected. The algorithm uses a constant false alarm detection strategy with a constant false alarm rate, `pfa`.

Gathering

The `gathering.m` function collects together detected chirplets to form individual linear chirps. Chirplets are associated with existing chirps if they are in the proximity of the existing chirp (in time, frequency and chirprate – see recommended parameters below). If there are multiple chirps that match in proximity then the closest in frequency is selected. It is possible for the chirplet to still be equally likely to be assigned to multiple chirps (particularly at crossing chirps). In this scenario the code generates the following warning:

```
'Warning: ambiguity in chirplet assignation'
```

The code then assigns the chirplet arbitrarily to the first chirp in the list.

The final step in `gathering.m` is to post process the chirp list and to remove chirps that are likely to be false positives. These are chirps with only a small number of window lengths in duration. In the post processing step any chirp of length `< 4*dt` is removed where `dt` is the time between frames (measured in number of samples). This is consistent with the recommended gathering parameter choice: `deltat=3*dt` below. Thus it will not detect chirps that do not extend for more than 4 chirplet windows.

Sig_gathering

In the final stage we implement a simple greedy signal gathering algorithm in `sig_gathering.m`. The algorithm assumes that all signals are piecewise continuous in TF and run from the start of the data to the end, i.e. that they take the form of piecewise linear chirps. Since we do not look for extreme chirprates we allow discontinuities in frequency.

The goal is to associate the end of each chirp with the start of another with the constraint that the beginning of each chirp may only be associated with one other chirp. Initial association is done through TF proximity with Time and Frequency given equal weighting (other weightings would be possible):

$$d_{i,j} = \sqrt{\frac{1}{N^2}\left(n_{e_j} - n_{b_i}\right)^2 + \frac{1}{K^2}\left(k_{e_j} + k_{b_i}\right)^2} \;, \forall i \neq j$$

with N and K the number of time and frequency bins and n_{b_j}, n_{e_j}, k_{b_j} and k_{e_j} the time and frequency bin positions of the start and end of chirp j. A secondary association is performed if chirps are not close in TF using a time based criterion:

$$d'_{i,j} = \left|n_{e_j} - n_{b_i}\right| \;, \forall i \neq j$$

This allows the connection of chirps with time proximity and discontinuities in frequency.

The set of starting chirps are identified and define the start of the output signals. Then each chirp is sequentially assigned to one of these signals or left unassigned (`signal number =0`).

Note: no information about the periodicity or similarity of chirprates has been used. Clearly more sophisticated chirp gathering algorithms would be possible.

Recommended Parameter Settings

There are a number of parameters that need to be set for this algorithm. Below we give our recommendations on how to select these values:

`wlen` This variable controls the length of the TF analysis window, $\phi_M[m]$, whose type is defined by `typef`. Ideally this should be of the order of the smallest linear chirp component in the FMCW signals that we are interested in detecting. The longer the window the more coherent gain that we will have. This will improve the SNR at which detection is possible. However longer window lengths require a finer sampling of the chirprate (and hence larger `nbcr` – see below). This can significantly increase the computation of the chirplet transform.

`typef` window type used in analysis:

'r' = 'Rect'
'k' = 'Kaiser',4
'h' = 'Hanning'
'hm' = 'Hamming'
'b' = 'Blackman'
'g' = 'Gauss' [DEFAULT if variable not assigned]

The window should be symmetric and normalized such that `max(win) = 1.0.`

`pfa` The `pfa` determines the target probability of false alarm for the individual T-F chirplet detections (this will be a slight underestimate for chirplets detected later in the process due to the correlation in the coefficients – see paper). We have found that setting this at `1e-3` gives a good compromise between too many false detections and too few. As this is only the low level PFA we are able to remove many of these false alarms through the chirp and signal gathering stages where isolated chirplets are rejected.

`nbcr` One of the main contributions of the paper is to identify the natural discrete sampling for the chirprate that is dependent on the TF analysis window. Specifically we recommend the chirprate discretization to be:

$$\Delta_d = \frac{2 * \mathrm{CRmax}}{\mathrm{nbcr} - 1} \approx \frac{2}{(\sum_m \phi_M[m])^2}$$

Rearranging in terms of code variables we get:

$$\mathrm{nbcr} \approx 1 + \mathrm{CRmax} * \left(\sum_m \phi_M[m]\right)^2$$

Finally note that for Hann window we have $(\sum_m \phi_M[m])^2 \approx 0.25 * \mathrm{wlen}^2$. This chirprate discretization ensures that the `winmax` function is a good approximation of the chirplet leakage into neighbouring frequency bins.

`CRmax` This is the maximum (positive or negative) normalized chirprate used in the chirplet transform. This is normalized to a sampling rate of unity. Hence a chirp signal with ramp time of $500\mu s$ and $\Delta f = 150\text{MHz}$ sampled at $Fs = 500\text{MHz}$ would have a normalized chirprate of:

$$CR = \frac{0.3}{2.5\times10^5} = 1.2\text{e-}6.$$

We should therefore select `CRmax` to be larger than the maximum anticipated chirprate,

$$\text{CRmax} \geq \frac{|\Delta f|}{T_{ramp} \times F_s^2}$$

It is possible with slightly reduced performance to search for chirps with a rate higher than `CRmax` (by setting `flag=1` in `gathering.m`). This allows us to reduce the computational load by reducing the number of chirprates that need to be calculated in the Chirplet transform. See the discussion in [Millioz and Davies 2012].

`deltat` The time proximity threshold for matching detected chirplets to existing chirps. this value is fixed (in `gathering.m`) to `deltat=3*dt` where `dt` is the time between frames (measured in number of samples). **WARNING**: `deltat` is also used in `sig_gathering.m` and separately defined.

`deltac` The chirprate proximity threshold for matching detected chirplets to existing chirps. In the code this value is fixed (in `gathering.m`) to `deltat=1.5*`Δ_d`.`

`deltaf` The frequency proximity threshold for matching detected chirplets to existing chirps. In the code this value is fixed (in `gathering.m`) to:

```
deltaf=max(df, deltac*deltat/2);
```

where `df` is the frequency resolution of the chirplet transform.

Appendix A

Notes

1. In the current implementation the FMCW signals are assumed to span the full duration of the data. Therefore it is not suitable for the detection of signals such as pulsed radar signals that only span part of the signal. However it would be possible to easily adapt the algorithm to deal with such a scenario.
2. The noise in the signal is assumed to be white and Gaussian. Noise level is not assumed known and is estimated using the spectral kurtosis of the minimal statistics [Millioz and Martin, ICASSP, 2010]. A constant false alarm rate (CFAR) detection strategy is then adopted based on this noise estimate (this is within detectionWinmax.m). Other forms of noise level estimation and CFAR detection could be implemented, including local variance estimation as in classical cell-averaging CFAR detectors.
3. The calculation of the upper bound window in `chirpgathering.m` (using function `maxwindow.m`) is data independent and so for a real time implementation could be performed off-line and stored in memory.

Appendix B

Main code functions and scripts as given in the readme file:

Main program:	chirpgathering.m
Example use is in:	miniscript.m

Each script has a help showing how to use the function

In Matlab : >> help <function>

chirpgathering.m	Main program, calls a chirplet transform, a computation of the maximum window, a detection, a gathering into chirps. Note: doesn't gather into signal Note2: a flag may be set in the script to gather chirps over maximal chirprate Note3: the detection used is based on the maximum window (a pseudo-Matching Pursuit approach)
chirpletc.m	Chirplet transform for complex signal (all frequencies)
chirplet.m	Chirplet transform for real signals (half frequencies)
chirpprofilec.m	Creation of a complex signal made of piecewise linear chirplets
detectionWinmax.m	Detection using a pseudo-MP approach (cf [Millioz and Davies 2012])
gathering.m	Gathering of the chirplets into chirps
sig_gathering.m	Gathering of the chirps into signals
id2ids.m	Translation of Matlab index into set of coordinates
maxwindow.m	Computation of the maximal window
tftb_window.m	Generates the Time-frequency analysis window used in the chirplet transform. This function is written by F. Auger, June 1994 - November 1995, copyright (c) 1996 by CNRS (France), and is freely available under the terms of the GNU General Public License

Auxiliary display functions:

dispchirp.m	Display of the selected chirps or chirplets
dispsig.m	Adaptation of dispchirp to plot the signal

Example code:

miniscript.m	Example script