RITSAR

Douglas Macdonald

Rochester Institute of Technology

March 29, 2015

Contents

1 Introduction

Below is a description of each of the functions contained in the ritsar package. For installation instructions, reference the README file.

Each of the sections below represent a module contained within the ritsar package and each subsection details how to use each function. The format for the title of each subsection is "function(parameters)." This is followed by a short description of what the function does. Next, each parameter is defined. Finally, a description of what the function returns is given.

As an example, lets say you wanted to simulate your own phase history. The function required for this is simulate_phs located in the phsTools module. Before this function can be used, the required parameters need to be defined. The descriptions of each required parameter can be found in subsection 2.1. The first parameter that needs to be defined is "platform", which is a Python dictionary. A description of each key that goes into platform is also given in subsection 2.1. Once platform is defined, the user then creates an array specifying the locations of each point reflector in the scene as well as the corresponding amplitudes. After all this is done the user can then create their own phase history, stored in the phs variable, by typing the following into the Python console:

phs = ritsar.phsTools.simulate_phs(platform, points, amplitudes)

For a full end-to-end demonstration, "main.py" in the ./examples folder can be run. The default setup processes an X-band collect using the polar format algorithm. The final lines of main.py can be commented/un-commented to use the backprojection algorithm. For a demonstration of the omega-k algorithm, OpenCV will be required. You will most likely also want to use a UHF setup. Under the "#include dictionaries" block near the top of main.py, change SARplatform to SARplatformUHF and comment/un-comment the bottom lines of the file as necessary. The platform dictionary keys defined in SARplatform and SARplatformUHF were taken from Carrera [1].

2 phsTools

2.1 simulate_phs(platform, points, amplitudes)

This function is a basic phase history simulator. It takes a list of target locations and amplitudes and saves the demodulated signal to './phase_history.npy'. It also outputs the signal to the function caller. The amplitude of the received signal for a perfect point reflector is assumed to be one arbitrary unit. A uniform beam pattern is assumed and $1/r^4$ attenuation is not modeled.

Parameters:

- **platform** : dictionary
  dictionary that has the following keys defined:
  - **nsamples** : int
    number of fast time samples
  - **npulses** : int
    number of pulses
  - **chirprate** : float
    chirprate in Hz/s
  - **f_0** : float
    carrier frequency in Hz
  - **t** : float array [nsamples]
    demodulated fast-time centered at t=0. The array should go from $(-nsamples/2, nsamples/2) \times \Delta t$ where, Δt = (A/D sampling rate)$^{-1}$
  - **pos** : float array [npulses x 3]
    (x,y,z) coordinates of platform, referenced to scene center, which is located at (0,0,0)
- **points** : float array [npoints x 3]
  locations of each point referenced to scene center
- **amplitudes** : float array [npoints]
  reflectance of each point. May be a complex number.

Returns:

- **phase_history** : complex array [npulses x nsamples]
  resultant phase history

2.2 RVP_correct(phs, platform)

Corrects Residual Video Phase using the formulation in Carrera Appendix C.

Parameters:

- **phs** : complex array [npulses x nsamples]
  complex phase history to be corrected for residual video phase
- **platform** : dictionary
  dictionary that has the following keys defined:
  - **nsamples** : int
    number of fast time samples
  - **npulses** : int
    number of pulses
  - **chirprate** : float
    chirprate in Hz/s
  - **delta_r** : float
    spacing between fast time samples. Defined here as $\frac{c}{2B_{IF}}$ where, B_{IF} is the bandwidth of the intermediate frequency (de-chirped) signal. B_{IF} is simply $(t_{max} - t_{min}) \times chirprate$ where t is the demodulated fast time, as defined in subsection 2.1.

Returns:

- **phase_history** : complex array [npulses x nsamples]
  resultant phase history, corrected for residual video phase

2.3 phs_to_const_ref(phs, platform, upchirp)

This program converts a phase history that was demodulated using a pulse dependant range to scene center to a phase history that is demodulated using a fixed reference. The fixed reference is defined as the minimum range to scene center.

Parameters:

- **phs** : complex array [npulses x nsamples]
  complex phase history that was compensated to a point
- **platform** : dictionary
  dictionary that has the following keys defined:
  - **npulses** : int
    number of pulses
  - **chirprate** : float
    chirprate in Hz/s
  - **f_0** : float
    carrier frequency in Hz
  - **t** : float array [nsamples]
    demodulated fast-time centered at t=0. The array should go from $(-nsamples/2, nsamples/2)\times\Delta t$ where, Δt = (A/D sampling rate)$^{-1}$
  - **pos** : float array [npulses x 3]
    (x,y,z) coordinates of platform, referenced to scene center, which is located at (0,0,0)

Returns:

- **phase_history** : complex array [npulses x nsamples]
  resultant phase history, compensated to a line

3 imgTools

In this section the img_plane dictionary is introduced. A sample img_plane dictionary generator is provided in sarit/examples/dictionaries.

3.1 polar_format(phs, platform, img_plane, taylor)

This is the Polar Format algorithm. The phase history data as well as platform and image plane dictionaries are taken as inputs.

The phase history data is collected on a two-dimensional surface in k-space. For each pulse, a strip of this surface is collected. The first step in this program is to project each strip onto the (ku,kv) plane defined by the normal vector contained in the image plane dictionary. This will result in data that is unevenly spaced in (ku,kv). This unevenly spaced data is interpolated onto an evenly spaced (ku,kv) grid defined in the image plane dictionary. The interpolation is done along the radial direction first, then along the along-track direction. Further details of this method are given in both the Jakowitz and Carrera texts.

Parameters:

- **phs** : complex array [npulses x nsamples]
  complex phase history
- **platform** : dictionary
  dictionary that has the following keys defined:
  - **npulses** : int
    number of pulses
  - **f_0** : float
    carrier frequency in Hz
  - **pos** : float array [npulses x 3]
    (x,y,z) coordinates of platform, referenced to scene center, which is located at (0,0,0)
  - **k_r** : float array [nsamples]
    spatial frequencies corresponding to each fast time return. For a linear FM signal, this is given by $\frac{4\pi}{c}(f_0 + chirprate \times t)$
  - **R_c** : float
    range to scene center at aperture center
- **img_plane** : dictionary
  dictionary that has the following keys defined:
  - **n_hat** : float array [3]
    unit normal vector to image plane
  - **k_u** : float array [size(u)]
    image plane spatial frequencies along u axis. u is defined to lie along the slant range vector, projected onto the image plane.

  - **k_v** : float array [size(v)]
    image plane spatial frequencies along v axis. v is defined to lie perpendicular to the slant range vector in the image plane.

- **taylor** : int
  attenuation factor for taylor window in dB. defaults to 43 dB.

Returns:

- **img** : complex array [npulses x nsamples]
  processed image

3.2 backprojection(phs, platform, img_plane, taylor, upsample)

This is the Backprojection algorithm. The phase history data as well as platform and image plane dictionaries are taken as inputs. The (x,y,z) locations of each pixel are required, as well as the size of the final image (interpreted as [size(v) x size(u)]).

Parameters:

- **phs** : complex array [npulses x nsamples]
  complex phase history

- **platform** : dictionary
  dictionary that has the following keys defined:

  - **nsamples** : int
    number of fast time samples
  - **npulses** : int
    number of pulses
  - **pos** : float array [npulses x 3]
    (x,y,z) coordinates of platform, referenced to scene center, which is located at (0,0,0)
  - **k_r** : float array [nsamples]
    spatial frequencies corresponding to each fast time return. For a linear FM signal, this is given by $\frac{4\pi}{c}(f_0 + chirprate \times t)$
  - **delta_r** : float
    spacing between fast time samples. Defined here as $\frac{c}{2B_{IF}}$ where, B_{IF} is the bandwidth of the intermediate frequency (de-chirped) signal. B_{IF} is simply $(t_{max} - t_{min}) \times chirprate$ where t is the demodulated fast time, as defined in subsection 2.1.

- **img_plane** : dictionary
  dictionary that has the following keys defined:

  - **u** : float array [len(u)]
    for the backprojection algorithm, u is essentially a placeholder that defines one of the dimmensions of the output image. Only the length of u matters, not its contents.
  - **v** : float array [len(v)]
    for the backprojection algorithm, v is essentially a placeholder that defines one of the dimmensions of the output image. Only the length of v matters, not its contents.
  - **pixel_locs** : float array [3 x n_pixels]
    flattened array specifying the 3-dimensional location of each pixel in the rectangular output array whose shape is [len(u) x len(v)]. Consequently, $n\_pixels = len(u) \times len(v)$. The locations do not have to lie in a plane and can be specified to an arbitrary surface that matches the local terrain. A rectangular output array with the location of each pixel specified is the only constraint.

- **taylor** : int
  attenuation factor for taylor window in dB. defaults to 43 dB.

- **upsample** : int
  upsampling factor, default is 6.

Returns:

- **img** : complex array [npulses x nsamples]
  processed image

3.3 omega-k(phs, platform, taylor, upsample)

This is an omega-k algorithm based off of the algorithm prescribed in the Carrera text. Only the phase history and platform files are taken as inputs, an img_plane dictionary is not required.

The input phase history needs to have been demodulated to a fixed reference. If demodulated to scene center, the included phs_const_ref file can do the conversion for you. A straight line flight path is also assumed.

The first step in the algorithm is to perform a 1D FT along azimuth. A matched filter is applied to the resultant data to perfectly compensate the range curvature of all scatterers having minimum range R_s. The default setting for R_s is the minimum range of scene center. To correct the range curvature for other scatterers, the data is mapped onto a new grid.

Parameters:

- **phs** : complex array [npulses x nsamples]
  complex phase history

- **platform** : dictionary
  dictionary that has the following keys defined:

- **nsamples** : int
  number of fast time samples
- **npulses** : int
  number of pulses
- **pos** : float array [npulses x 3]
  (x,y,z) coordinates of platform, referenced to scene center, which is located at (0,0,0)
- **k_r** : float array [nsamples]
  spatial frequencies of demodulated signal corresponding to each fast time return. For a linear FM signal, this is given by $\frac{4\pi}{c}(f_0 + chirprate \times t)$
- **k_y** : float
  spatial frequencies along flight path. A straight line flight path (or phase history motion compensated to a straight line flight path) are required.

- **taylor** : int
  attenuation factor for taylor window in dB. defaults to 43 dB.
- **upsample** : int
  upsampling factor, default is 6.

Returns:

- **img** : complex array [npulses x nsamples]
  processed image

4 phsRead

Coming soon

4.1 DIRSIG

4.2 AFRL

4.3 Sandia

References

[1] W. G. Carrera, R. S. Goodman, and R. M. Majewski, *Spotlight Synthetic Aperture Radar*. Artech House, Inc., 1995.