SARCASTIC User Manual

Darren G. Muff

17th July 2018

REVISION HISTORY

Revision	Date	Author(s)	Description
1.0	26 Aug 2018	DGM	created - SARCASTIC V5.0+
1.1	12 Apr 2019	DGM	corrections - SARCASTIC V5.1+
1.2	16 Apr 2019	DGM	Fixed Grazing angle in Table 31 - SARCASTIC V5.1.1+

CONTENTS

INTRODUCTION

SARCASTIC™ simulates a SAR collection using a computer aided design (CAD) model and a Synthetic Aperture Radar (SAR) data set. The SAR dataset used is called CPHD (Common Phase History Data) which is a format for storing pulse position and timing and other ancillary details (called *narrowband* data) together with the received and digitised signal collected by the radar (called the *wideband* data). The CPHD format specification is in the `doc` directory as `NGA.STND.0068-1_1.0.0_CPHD_Design_Implement_2016_09_30.pdf`

When a SAR collects data the CPHD file has both narrowband and wideband components. The purpose of SARCASTIC is to use the narrowband data and the CAD model to generate a new CPHD dataset that has the same narrowband data but with simulated wideband data. In this way the same SAR image formation processor (IFP) can be used to process the actual SAR collection as well as the simulated collection thereby making the two images easier to compare.

A basic simulation run follows the following steps:

- Obtain a CPHD file or generate your own that contains just narrowband data describing the SAR imaging operation. (The tool `cphdShell` can be used to do this).

- Build a CAD model of your target using your favourite CAD drawing package. Sketchup™ is useful for this as the Sketchup™ Warehouse has lots of fan-made CAD model. Save the CAD model as either a Collada (.dae) file or a .ply file.

    – If you saved your CAD model as a .dae file then you can convert it to a .ply file using `colladaToPlyFile`.

- The .ply file needs to be prepared as it is likely to contain lots of degenerate or polygamous triangles. The .ply file is prepared using the tool `materialise`. This operation performs several steps that will be discussed in more detail in Chapter 5.

- Now that the CAD model has been prepared and a CPHD dataset is available, `SARCASTIC` is run. This reads in both and produces a CPHD file as its output.

- the CPHD file is processed into a SAR image using a SAR Image Formation Processor. Any IFP can be used that can work with CPHD files but for completeness I have included a basic one called `tdpocl` with the distribution. If a true SAR collection was used as input to `SARCASTIC` (rather than a simulated one using `cphdShell`) then the true SAR collection can also be processed using `tdpocl` with the same parameters used for the simulated CPHD file.

- The image(s) are produced in a very simple file format called .dat. These can be viewed using the python script `imdisplay.py` which can be found in the `scripts` folder.

A flow chart describing this approach and the operation of such an exploitation tool can be seen in Figure 11.
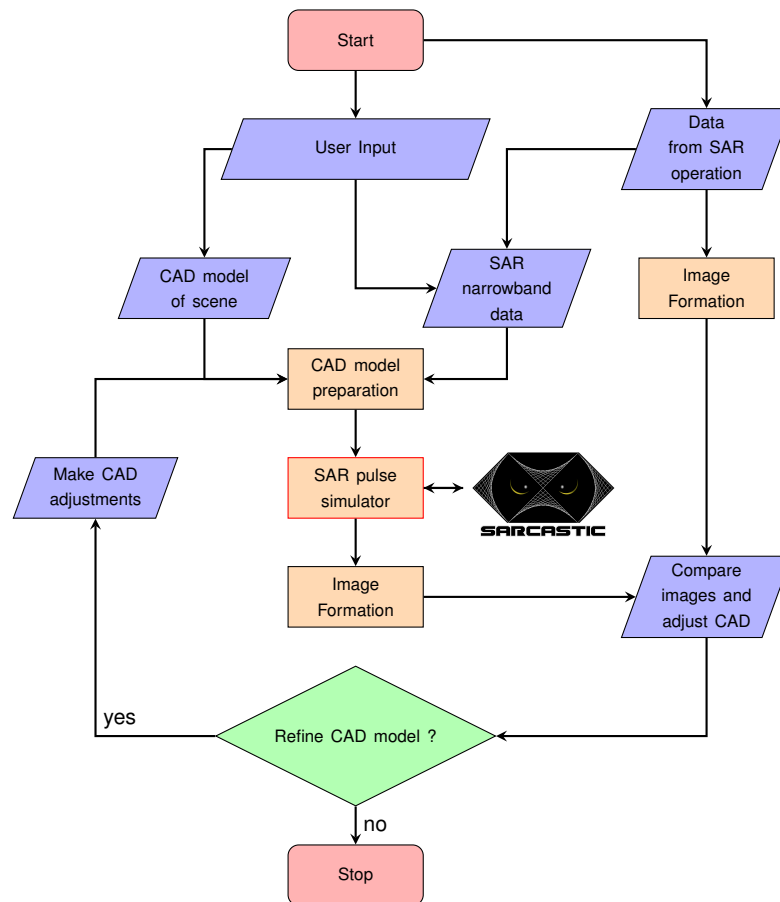

Figure 11: Process flow for an exploitation tool that will simulate a radar scene using collected raw data and a CAD model

1.1 Interacting with the Binaries

Each of the binaries in SARCASTIC has a common interface. It was decided that its quite useful to keep a record of how each dataset is produced or to have a record of the input parameters used when running an executable. This information is easily lost in the command line history and so a common interface is used that will store the user input to a `.rc` file as text. When you run a binary you may see a question like:

```
Enter cphd filename [./cphdFile.cph]:
```

The text in square brackets is the default response if you just hit `return`. You can also enter ? followed by `return` if you don't understand the question. For example:

```
Enter cphd filename [./cphdFile.cph]: ?
A CPHD filename to examine
```

Once the program has completed it will store the input information in a file that is preceded with a `.` and ends in `rc`. For example after successfully running `cphdInfo` the directory where you ran it from will contain the file `.cphdInforc`.
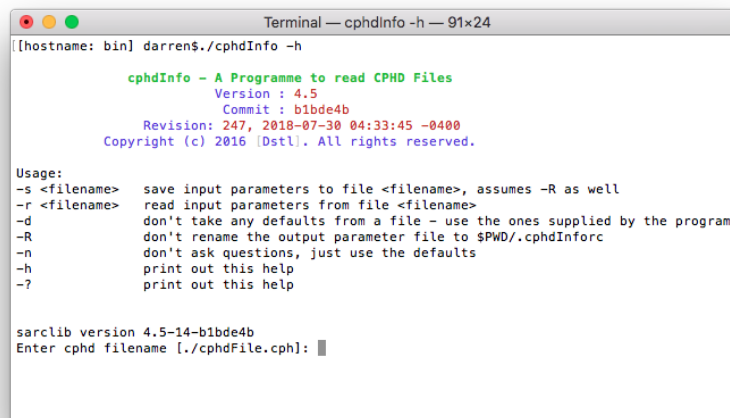
This is just a text file and you can look at its contents to see how you ran the binary using `cat`:

```
$ cat .cphdInforc
Fname:./cphdFile.cph
showNBData:N
showWBData:N
```

The next time that you run the programme, the `rc` file will be used as default input parameters so that you can see what you did last time. You can override this and use the program defaults by running the program with the `-d` option. There are also options to save the program input to a different file name or to not ask any questions at all and to just use the defaults. You can see all the options by adding `-h` to the command line. For example see Figure 12.

For completeness the options are:

- `-s <filename>`: save input parameters to file `<filename>`. Assumes `-R` as well

- `-r <filename>`: read input parameters from file `<filename>`.

- `-d` : Don't take any defaults from a file - use the ones supplied by the program

- `-R` : Don't rename the output parameter file to `$PWD/.cphdInforc`

- `-n` : Don't ask questions, just use the defaults

- `-h` : Print out this help

- `-?` : Print out this help

Figure 12: Example help information provided when running one of the SAR-CASTIC binaries with the -h option

2

INSTALLATION

This chapter will show you how to install SARCASTIC. To start with you will need a copy of sarcastic source (for example `sarcastic-5.0-Source.tar.gz`)
    place the copy of SARCASTIC in a folder on your Linux or OSX machine. Then unpack the source code using the `tar` command:

```
$ tar zxvf sarcastic -5.0 - Source.tar.gz
```

This should create a working directory that looks a bit like:

```
sarcastic -5.0 - Source
├── CMakeLists.txt
├── bin
├── cmake
├── doc
├── examples
├── scripts
└── src
```

Figure 21: SARCASTIC directory structure

2.1   Dependencies

Before compiling check that your system has the following packages installed:

- cmake (Version > 3.13.2 - on some systems called cmake3)

- git (Version > 2.17)

- gdal (Version > 2.3.2_1)

- opencl (Version > 1.2)

- `expat` (Version > 2.2.1)

- `fftw` (Version > 3.3.8)

- `boost` (Version > 1.68.0_1)

- `CGAL` (Version > 4.13)

- `readline` (Version > 7.0.5)

If any of these are missing then a warning will be thrown when running `cmake` to tell you which package(s) are not found.

2.2 Suggested Additional Packages

`SARCASTIC` uses a pretty basic CAD model file format called .ply (Stanford Triangle Format). You can find out more about it here. There are many ways to make a .ply file. from a CAD model. My preferred way at the moment is to use Sketchup™ Make™ (Its free) and export the file as a collada™ file before conversion to a .ply file.

A useful (and pretty awesome) tool to visualise your CAD model is meshlab which can be obtained from http://www.meshlab.net.

There are some useful python scripts in the `scripts` folder. To use these you will need `python-2.7` installed.

2.3 Compilation

`SARCASTIC` is compiled using the `cmake` build system. To build sarcastic just type the following commands from the command line:

```
$ mkdir build
$ cd build
$ cmake ..
$ make
$ make install
```

2.4 Setting the Path

After compilation it is useful to be able to run the binaries from a different directory that holds all your working files. There are two ways to do this. The first is simply to add the path to the `SARCASTIC` binaries to your executable path. If you are using the `bash` shell then either type the following from the command line or add it to your ˜/`.bash_profile` (changing the '/path/to' with your own installation directory path)

```
$export PATH="$PATH:/path/to/sarcastic/bin"
```

If you are brave then you can add the executables to the system /usr/lo-cal/bin directory. The `install` command places the executables in a subdirectory called `bin` so you just need to overide the `CMAKE_INSTALL_PREFIX` in the `cmake file` with :

```
        $ cmake -DCMAKE_INSTALL_PREFIX=/usr/local/ ..
```

(running this from your SARCASTIC build directory - obvs)

2.5 Summary

If all goes well you should end up with a directory structure the same as Figure 22 that contains the executable files (highlighted in red).

```
sarcastic-4.5.14-Source
├── CMakeLists.txt
├── bin
│   ├── SARTrace
│   ├── bircs
│   ├── colladaToPlyFile
│   ├── cphdInfo
│   ├── cphdShell
│   ├── materialise
│   ├── sarcastic
│   └── tdpocl
├── build
├── cmake
├── doc
├── examples
├── scripts
└── src
```

Figure 22: SARCASTIC after a successful build with executables highlighted in red

The next section will provide some useful information about the concepts in SARCASTIC. It will describe the processing steps required to convert a CAD model into a SAR image and explain some of the fundamentals required for running all the packages within the SARCASTIC distribution.

3

CREATE A RADAR COLLECTION TO SIMULATE

SARCASTIC simulates a SAR scene using narrowband information provided by a CPHD file. Ideally it would use an existing SAR collection for its CPHD data but this may not always be available. If this is the case then don't worry; the SARCASTIC distribution contains a tool to generate your own CPHD files (without the wideband radar data from actual received radar pulses). The tool for this is called `cphdShell` and in this section we will walk through its use to create a CPHD file.

3.1 Create a 'Blank' CPHD File Using `cphdShell`
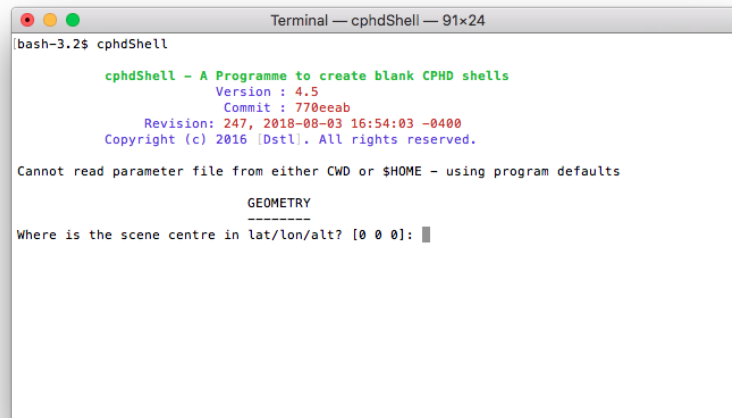
In this section we will use `cphdShell` to create a CPHD dataset. Before we do this we we need an idea of the imaging geometry that we want to simulate. This can be anything you want (within reason) but be aware that you will need to know a bit about Synthetic Aperture Radar if you want to input sensible parameters that will form an image at the end. I'll try and explain as much as possible as we proceed. To start with, create a working directory where you want to store all the files and change directory into it. Also set your path to be the `bin` directory for SARCASTIC (if you haven't already:

```
$ mkdir tutorial
$ cd tutorial
$ export PATH="$PATH:/path/to/sarcastic/bin"
```

Now run `cphdShell`. You should end up with something like in Figure 31
    The program is asking for a scene centre. Pressing `?` and hitting `return` will provide some hints one what it needs:

```
Enter the location on the Earth for the stabilisation
 reference point (SRP) in decimal degrees and metres
```

9

Figure 31: Running `cphdShell` for the first time

In this example we will pick an arbitrary location, ten degrees north of the equator, on the meridian and at 0 metres in altitude:

```
Where is the scene centre in lat/lon/alt? [0 0 0]: 10 0 0
```

The scene centre is used to reference the location of the pulses in the CPHD file. As CPHD uses a global coordinate system for all of its position information, this has to be a latitude, longitude and altitude[1].

For the tutorial we will use a nominal SAR system as an example sensor.

`cphdShell` will then ask for lots of different parameters. For this basic run, we will use the parameters provided in Table 31

Answer the questions using the parameters from the table. Some useful information will be printed out as the program runs. When all the input has been collected you will be presented with

```
Calculating Narrowaband data...
```

followed after a short while by

```
Writing CPHD File "outputCPHDFile.cph".......Done
```

If you see this then there is a good chance that your CPHD file has been written to the file that you specified. There is an easy way to check this though using the program `cphdInfo`.

3.2 Checking a CPHD File Using `cphdInfo`

The program to look at the contents of a CPHD file is called `cphdInfo` and sits in the `SARCASTIC bin` directory. You can run this in the usual way:

```
$ cphdInfo
```

---

[1]which is converted to Earth-centred, Earth-fixed (ECEF) internally

Parameter	.rc label	Value	Comment
Scene centre	SRPlla	10 0 0	Latitude/longitude/altitude
Slant range	TxRslant	5000.0m	
Grazing angle	TxGraz	35.26deg	Square trihedral boresight
Azimuth angle	TxAzim	270deg	
Look direction	TXlookDir	Right	right of velocity vector
Squint angle	TXSquint	0deg	Broadside. +ve is forward
Sensor mode	SARMODE	SPOTLIGHT	
Bistatic	bistatic	n	CPHD file is monostatic
Centre frequency	fcent	9.6e9Hz	
Antenna length	TxAntLen	1.0m	
Antenna height	TxAntHgt	0.3m	
Platform velocity	TxVel	50m/s	
PRF	PRF	100Hz	
Azimuth image size	imageX	100m	check for ambiguities
Range image size	imageY	100m	check for ambiguities
Azimuth resolution	azRes	0.1m	
Range resolution	raRes	0.1m	
On the ground?	groundres	Y	determines bandwidth
Az oversampling factor	oversampAz	1.2	
Rng oversampling factor	oversampRa	1.2	
Pulse len (or duty cycle)	pulselen	0.02ms	
ADC rate	ADCRate	100MHz	
Datetime string	dateTime	20180816080500	Whatever you want
Dataset ID	datasetID	Tutorial	Whatever you want
Output file name	outCPHDFilename	outputCPHDFile.cph	Whatever you want
CPHD file type	CPHDTYPE	3	'3' is simplest
Polarisation	polarisation	VV	
Sensor name	sensorName	BrightSpark	

Table 31: A set of simple input parameters to `cphdShell` to create a CPHD file with similar narrowband data as Bright Spark

When asked to enter a CPHD file just enter the one you created above. You can also print out the narrowband data for each pulse which will provide a location and time for each pulse in the CPHD file. For now we will say 'no' to this. There is also the option to save the wideband data to a file. This is useful for debugging. When 'yes' is applied to this option, `cphdInfo` will perform a 2D FFT of the wideband data and save it to a file in a DSTL `.dat` format. It can then be viewed (using `imdisplay.py` - more on this later) to gain insight into whats going on inside the wideband data of the CPHD file.

Once you run `cphdInfo` you should get something that looks a bit like Figure 32 in your terminal. It is worth pointing out (because I always forget) that the azimuth resolution in `cphdInfo` is the *finest* that can be achieved so if you specified an oversampling factor in `cphdShell` then this will be finer than the requested azimuth resolution.

```
[bash-3.2$ cpHdInfo

            cphdInfo - A Programme to read CPHD Files
                      Version : 4.5
                      Commit : 3d8dd45
             Revision: 247, 2018-08-10 17:16:53 +0100
             Copyright (c) 2016 [Dstl]. All rights reserved.

[Enter cphd filename [outputCPHDFile.cph]:
[Do you want to see the narrowband data in each pulse [N]:
[Do you want to save the wideband data to a file? [N]:

                    Collection Summary
          -------------------------------------------------------
Image ID                  : Tutorial-BSpark
Time of Collect           : 20180816080500
Dataset Classification    : OFFICIAL
Collection mode           : SPOTLIGHT
Collection Geometry       : Monostatic
Is SRP fixed              : Yes
Sensor                    : BrightSpark
CPHD file data type       : cmplxf
Number of channels        : 1
Channel 0 polarisation    : VV
  Azimuth 3dB beam width   : 0.021915
  Elevation 3dB beam Width : 0.043829
Deskew applied?           : No
Tasked SRP                : 10.000000 0.000000 0.000000 (deg, deg, m)
Centre Azimuth is         : 270.0 deg
Centre Grazing angle is   :  10.0 deg
Centre Squint angle is    : -0.000000 deg
Centre range is           : 5000.0 metres
Number of pulses          : 52606
ADC Samples per pulse     :  8266
Chirp rate                : 3.805e+14 Hz/sec
ADRate                    : 4.000e+08 Samps/sec
TX Pulse duration         : 0.020000 milliseconds
Look direction is         : 'Right'
Mean PRF                  :  1667.0 Hz
Collection duration       :    31.6 secs
Transmit bandwidth        : 7610.431 MHz
Slant Range resolution    :   0.020 m
Ground Range resolution   :   0.020 m
Transmit Centre Frequency :  34.200 GHz
Transmit Chirp start/stop : 30394.785 - 38005.215 MHz
Centre wavelength         :   0.009 m
Synthetic aperture centre : 6282728.052037,-4924.038765,1100398.337020 metres
Velocity at aperture Centre :   50.0 m/s
Synthetic aperture distance :  1577.8 m
Finest azimuth resolution :   0.014 m
Maximum range scene size  : 157.569 m
Illuminated area (az x ra) :  109.6 x  1262.0m
bash-3.2$
```

Figure 32: Terminal output when running `cphdInfo` command on the CPHD file generated using `cphdShell` in the above example

3.3 Summary

In this chapter we generated a CPHD file that we can use to simulate SAR data using `cphdShell`. We also saw how a CPHD file can be inspected using `cphdInfo`. In the next chapter we will look at how to generate a CAD model in a format that SARCASTIC can interact with.

4

BUILD YOUR CAD MODEL

The next step is to build a scene model that can be simulated using the parameters specified in the CPHD file generated in the last chapter. This might sound simple as there are many CAD software packages that are freely available. Despite this it can be quite challenging to make sure that the CAD model is converted into an internal format that supports the precision required for electro-magnetic field tracing. SARCASTIC uses a two-step approach to convert the data:

- The first step uses a CAD package to design and build the scene and then convert the output to a triangle soup in a simple-to-read format. A triangle soup is just a list of triangles, each individually specified but with no inherent structure or association to other triangles within the model.

- The second step converts the triangle soup into a well-ordered triangle mesh that can efficiently be used by SARCASTIC.

The program used to perform the first step is called `colladaToPlyFile` and will be covered in this chapter. The program used for the second step is called `materialise` (for reasons that will become apparent) and will be covered in the next chapter.

4.1   Preparing the CAD Model

The first step is to design your scene. I have found the Sketchup™ package to be useful for this. Not only is there a free version but you can export the files as Collada™ `.dae` files which is an open standard. I have also used Autodesk Maya™ which seems like a more powerful, if more complicated (and more expensive) tool. At some point I'll explore Blender (when I'm brave enough).

There is a Sketchup™ file containing a test scene in the `examples` directory called `trihedrals.skp`. If you load it into Sketchup™ it should look something like Figure 41. The figure shows a small 'farm' of 9 trihedrals. By convention

SARCASTIC assumes that the Y-axis (green in Sketchup™) points to the direction of north and the origin of the CAD model is the scene centre position that is stored in the CPHD file. In this case the trihedrals are all facing the direction of the negative X-axis which is the same as pointing to the west. As the CPHD file defined the azimuth angle of the collection to be 270° then this is perfect for the tutorial.

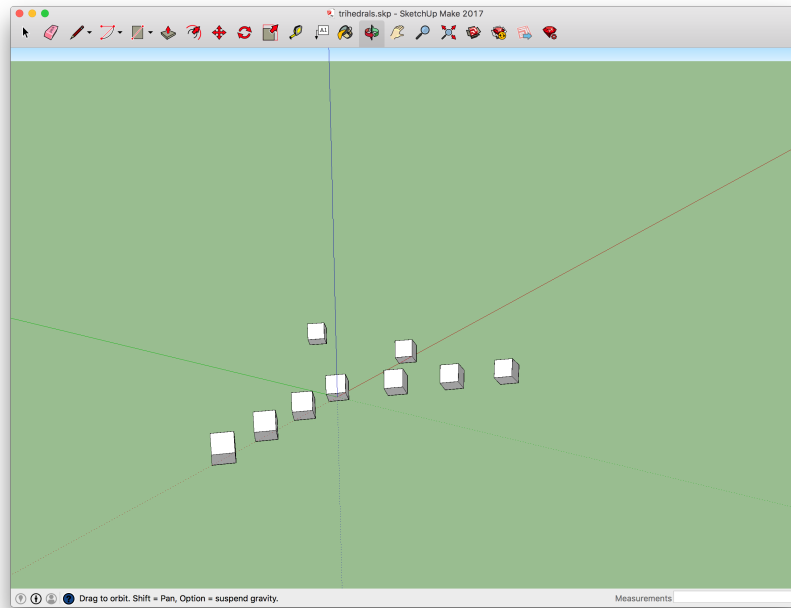Figure 41: `trihedrals.skp` file displayed using Sketchup™

The next step is to convert the CAD model to a Collada file. To do this:

- In Sketchup™ go to `File->export->3D Model`

- In the dialogue box make sure that the format is `COLLADA (*.dae)` and navigate to the location where the tutorial files are being kept.

- Under `options` make sure that `Export Two-Sided Faces` and `Triangulate All Faces` are ticked:
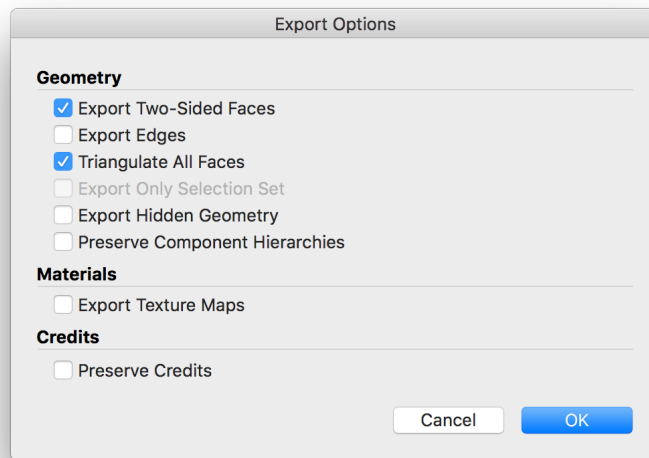
- Hit `OK` and then `Export`

Your tutorial directory should now have a file in it called `trihedrals.dae`.

4.2 Using `colladaToPlyFile`

The Collada file now needs to be converted to a .ply file. This is a simple file format developed by Stanford University. Details of the format and some useful tools can be found here. To convert the file we will use the program `colladaToPlyFile`. Rune the file as usual:

```
$ colladaToPlyFile
```

Then enter the name of the .dae file and teh name of the desired output file:

```
Collada .dae file to read [scene.dae]: trihedrals.dae
Name of output file [triangles.ply]: triangles.ply
```

The program will then ask you to :

```
Input materialfile filename [/.../MaterialProperties.txt]:
```

Its ok to hit return here and accept the default material file as this path was set up when SARCASTIC was compiled. For information the materialfile is a text file that contains some basic information about different material types and their effects on radar imagery. It will be discussed more in the next chapter.

Once all the input has been set you will be presented with a lot of text information about the triangles present in the .dae file. If the last line says `Done` then everything worked OK and the collada file has been converted to a .ply file.

4.3   Looking at the Results

To check that the file was produced OK load it into `meshlab`. It should look something like Figure 42
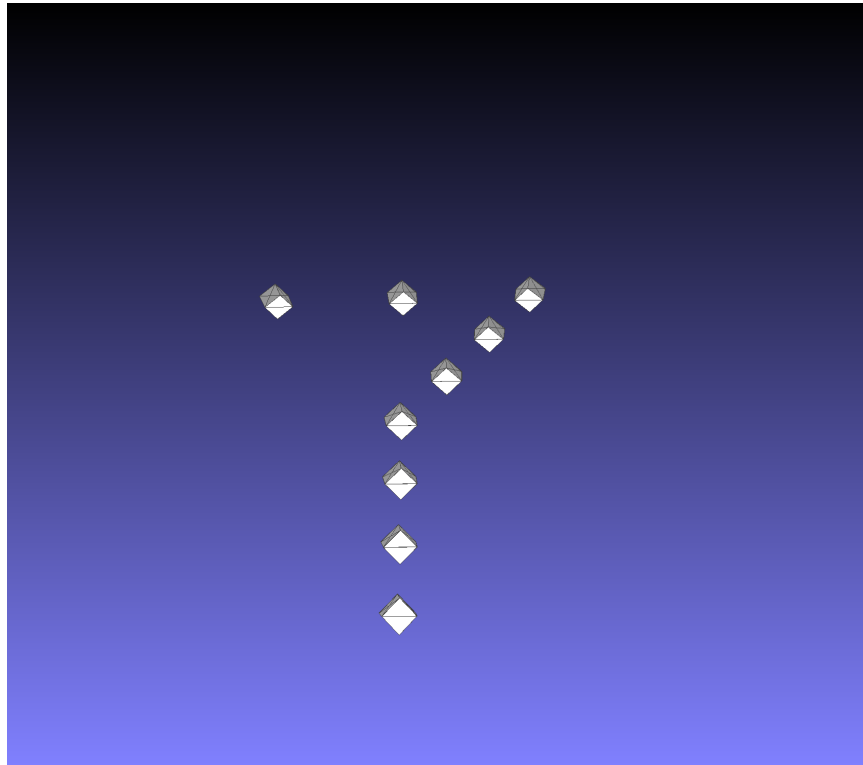

Figure 42: Output from `colladaToPlyFile` using the example `trihe-drals.dae` as input and display using `meshlab`

PREPARE FOR RADAR RAY TRACING

A triangle mesh is comprised of a set of unique vertex points with triangles being defined by linking three points together. The original triangle soup can contain degenerate[1] or skinny triangles that can cause errors when calculating the surface field integral of the incident EM field. To address this the triangle mesh is Delaunay triangulated into smaller, nearly-equal-angled, triangles. Real-life materials also have a coarser surface roughness than a perfectly planar surface produced by a CAD package. Rather than manipulating the initial mesh in the CAD package (a process called 'bump-mapping'), the approach adopted by SARCASTIC is to distort the mesh by a random amount which is determined by the material properties. To perform both the Delaunay triangulation and the surface roughening we use a program called `materialise`. This will be discussed in this chapter.

5.1 Materialise

To run `materialise` type:

```
$ materialise
```

You will then be asked to enter the name of a `ply` file for input, the name of an output `ply` file and the name of a material properties file (see Figure 51)

The first two parameters are pretty obvious but the material file needs a bit of explanation as it is used throughout the process.

5.2 Material Properties File

The path to the `materialProperties.txt` file is set up during compilation so you can see where the default file lives in the source tree. This file describes

---

[1]degenerate triangles are those triangles that have two of its vertices at locations that are closer together than the machine's precision
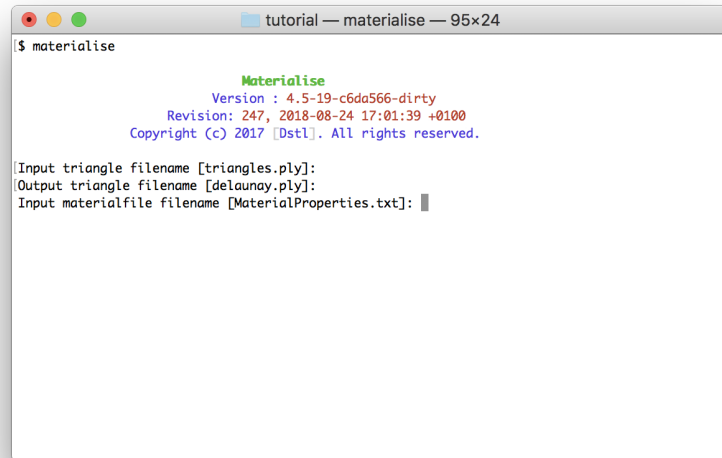
Figure 51: Running `materialise`

id	Name	corrLen	Roughness	Rs	Rm	Specular	R,G,B
00	MATERIAL	0.5	0.0	0.0	9.0e9	0.7	255 255 255
01	ASPHALT	0.5	0.05	1.0e18	9.0e9	0.8	128 128 128
02	BRICK	0.1	0.001	1.0e18	9.0e9	0.7	224 224 224
03	CONCRETE	0.2	0.01	120.0	9.0e9	0.3	176 19 35
04	METAL	0.6	0.0	1.0e-8	9.0e9	1.0	176 94 41
05	ROOFING	0.1	0.1	1.0e18	9.0e9	0.6	214 186 062
06	VEGETATION	0.1	0.1	2000.0	9.0e9	0.2	166 214 054
07	WATER	0.0	10.1	2.0e1	9.0e9	1.0	056 125 214
08	WOOD	0.1	0.001	1.0e14	9.0e9	0.6	137 046 014
09	GRASS	0.05	0.1	2000.0	9.0e9	0.2	100 214 050
10	PLASTIC	0.05	0.0	1.0e20	9.0e9	0.7	200 0 0

Table 51: Default contents of the `materialProperties.txt` file

the names of key materials and associates various scattering properties with the material. When the CAD model is built, the facets can be 'painted' with a swatch in Sketchup™. During the `colladaToPlyFile` process the material names in the CAD file are searched to see if any of the materials match those in the `materialProperties.txt` file. If they do then the parameters within the `materialProperties.txt` file are used to deform the mesh and for EM scattering calculations within SARCASTIC.

The default material properties can be seen in Table 51

If a facet within the CAD file does not have a material swatch applied to it then it takes on the identity of the item in the table called MATERIAL. This is set up to be a perfect electrical conductor (PEC). The definition of the parameters

in Table 51 are :

- **id** : This is the index of the material number in the table and is used to cross reference parameters throughout the execution of sarcastic tools.

- **Name** : This is the name of the material. If `colladaToPlyFile` is run then these names are searched in the collada™ file to work out which material each facet is made of. This is useful if you use Sketchup™ to make your CAD model as you can 'paint' the facets in the CAD model with the paint tool and the color description of the paint will be associated with one of the names in the table below if the paint name matches.

- **corrLen** : This is the correlation length of the facet. It is used when decomposing the triangle mesh into Delaunay triangles and sets the maximum area for each Delaunay triangle.

- **Roughness** : After Delaunay triangulation the mesh triangles are distorted in the direction of the triangle normal by a random amount. The standard deviation of the distortions are this number.

- **Rs** : Electrical resistivity of the material

- **Rm** : Magnetic conductivity of the surface material

- **Specular** : The amount of ray energy passed forward in the specular direction after each bounce.

- **Diffuse** : The amount of energy scattered diffusely after each bounce (not used)

- **Shinyness** : Not used

- **R,G,B** : Used for colouring each material when saving the triangles to a `.ply` file

5.3   The `materialise` **Algorithm**

Now that we know what the `materialProperties` file does we can run `materialise`. When the program is executed it performs the following steps:

- The ply file is read in and validated. This includes checking for and removing degenerate triangles and duplicate points.

- The program finds the first triangle in the mesh and attempts to grow the region around it by comparing all the other triangles and checking them to see if they are made of the same material, have the same surface normal *and* share a vertex point with any of the vertices already in the region. If a triangle is found it is added to the region and removed from the list of remaining triangles.

- Once all the triangles within a region are found then the vertices are Delaunay triangulated using the `corrLen` parameter to specify the maximum size of each Delaunay triangle

(a) Before   (b) After
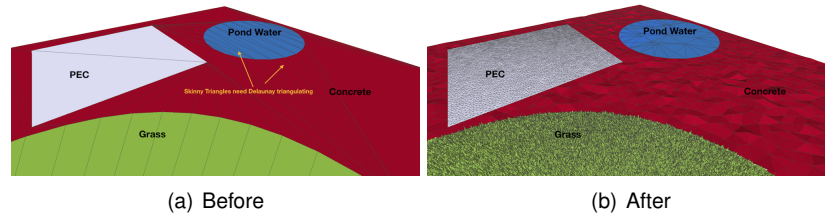
Figure 52: Comparison of CAD models before and after `materialise`

(a) Before   (b) After
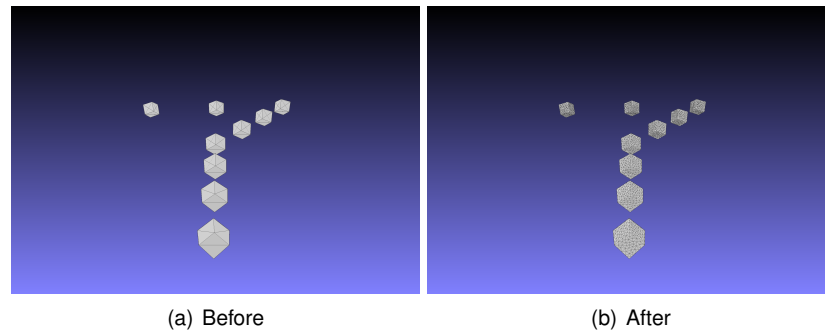
Figure 53: Trihedral array before and after `materialise`

- Following Delaunay triangulation, the vertices that are on the perimeter of the new mesh are identified

- For each item not on the perimeter of the mesh, the vertex is adjusted by a random amount in the direction of the facet normal. The amount of displacement is set so that the standard deviation of all displacements is equal to the `Roughness` parameter.

5.4   Comparing Results

Depending on the number of new triangles being generated, `materialise` can take a while to run. This is one of the reasons that the program is separate to the `SARCASTIC` main executable. The progress meter will provide an indication of run time during execution.

Once complete the new triangle mesh is saved to the .ply file specified in the program input. For comparison, Figure 52 shows an example scene before and after running `materialise`.

If `materialise` is performed on the triangle array from the examples folder then it should look similar to Figure 53

RUN SARCASTIC

If you've made it this far then you should have a CPHD file and a CAD model in .ply file format. The next step is to run SARCASTIC to simulate the radar responses from the scene as if it had been illuminated b a SAR sensor that has the same narrowband parameters as those in the CPHD file.

SARCASTIC is run from the command line and should look something like Figure 61.

```
$ sarcastic
```

You should then see something like:

Enter the name of the PLY file that we prepared in the previous section using MATERIALISE. SARCASTIC will then read in the file and performs some basic integrity checks on it. Next enter the name of the CPHD file that contains the narrowband information to be used. The program will then ask for a start pulse within the CPHD file and the number of pulses to simulate. A few basic scene parameters will then be provided including the scene size and the *'Max undersample for image'*. This last term is worth a short discussion on. SAR-CASTIC assumes that the scene is collected in a spotlight mode of operation. In this mode the pulse repetition frequency within the CPHD file determines the azimuth extent of the SAR image. As the simulated scene is often much smaller than that of a true SAR collection, it is likely that the PRF does not need to be so high for the simulated scene. The reduction in required PRF is called the *'Max undersample for image'*. The next question question will ask for a *max undersample factor*. Enter a number between 1 and the value provided as *'Max undersample for image'*. The main reason for not using a value of '1' is to speed up the processing time (and as the processing time is proportional to the number of pulses being simulated a undersample value of N will provide a reduction in processing time by N).

The next question asks the user for instructions on how to generate the rays that will be cast through the scene to simulate the EM fields. The options are :

Figure 61: Running SARCASTIC

- 'TRIANGLECENTRE'. This uses the centre of each triangle within the CAD model as the aim point for each ray. As a result there will be the same number of rays as there are triangles. This is the most efficient ray generation technique but if the scene contains larger triangles then the ray density can be sparse after the first bounce meaning that not all triangles are intersected on multiple bounces. This is best when just looking at surface features with fewer multipath artifacts.

- 'RANDOMRAYS'. This process generates a Gaussian distribution of random rays at the scene. If this option is chosen then the program will next ask how many rays in elevation and azimuth to generate. This provides a level of control over run time with more rays taking longer but more precisely intersecting with multipath facets.

- 'FIRSTTIMERANDOM'. This option generates a field of random rays on the first pulse and then reuses the aim points for each subsequent pulse. It guarantees that the same position on each triangle is hit on the first bounce for all pulses. This option is mainly used for debugging.

- 'PARALLELRANDOM'. This is another debugging option. It generates a Gaussian distribution of random rays but then adjusts the origin of each ray to make all the rays parallel. In this way it simulates the rays originating from a point at infinity and is useful when debugging scenes where the simulated sensor is very close to the target area.

The next question asks for the polarisation to use. Options are 'VV','VH','HV','HH','V_' and 'H_'. If one of the last two are used then the received 'H' and 'V' fields will be combined.

     Following this the name of an output CPHD file is asked for. This should be different to the input file. This file will contain the correct narrowband information but also have the wideband data filled in from the simulated returned signal.

     Finally the name of a material properties file will be asked for. This should be the same one as used in MATERIALISE.

     If the default example was used then the output should look something like:

```
bash-3.2$ sarcastic

       SAR Ray Caster for the Intelligence Community [SARCASTIC]
                 Version : 4.5-28-dc7a03f
            Revision: 247, 2018-10-05 17:56:11 +0100
         Copyright (c) 2017 [Dstl]. All rights reserved.


Name of Base scene [trihedrals.del.ply]:
Reading in file trihedrals.del.ply...
Done. Checking file Integrity...
Done
CPHD Filename [outputCPHDFile.cph]:
Start Pulse (0-2247) [0]:
Number of pulses (1-2249) [2249]:
Scene Doppler bandwidth      :     0.01 Hz
Mean PRF is                  :   100.04 Hz
Scene extent in azimuth      :    26.51 m
Finest azimuth resolution    :     0.07 m
Min Az image pixels          :      381 pix
Max undersample for image    :        5x
Pulse undersampling factor [1]:
Effective PRF is             : 100.044484 Hz
There are four ways to cast rays in SARCASTIC :
        1 : TRIANGLECENTRE
        2 : RANDOMRAYS
        3 : FIRSTTIMERANDOM
        4 : PARALLELRANDOM
Enter number for ray generation method or '?' for help [2]:
Azimuth rays in radar beam? [300]:
Elevation rays in radar beam? [300]:
Input file has polarisation set to 'VV'
Enter polarisation to simulate [VV]:
Output CPHD Filename [outputCPHDFile.sarc.cph]:
Input materialfile filename [MaterialProperties.txt]:
Material properties used in this file :
 id          Name CorrLen Roughness       Rs        Rm
  0      Material   0.500     0.000 0.0e+00  9.0e+09
  1       ASPHALT   0.500     0.050 1.0e+18  9.0e+09
  2         BRICK   0.100     0.001 1.0e+18  9.0e+09
  3      CONCRETE   0.200     0.010 1.2e+02  9.0e+09
  4         METAL   0.600     0.000 1.0e-08  9.0e+09
  5       ROOFING   0.100     0.100 1.0e+18  9.0e+09
```

```
  6      VEGETATION    0.100      0.100 2.0e+03 9.0e+09
  7           WATER    0.000     10.100 2.0e+01 9.0e+09
  8            WOOD    0.100      0.001 1.0e+14 9.0e+09
  9           GRASS    0.050      0.100 2.0e+03 9.0e+09
 10         PLASTIC    0.050      0.000 1.0e+20 9.0e+09
```

```
                        Collection Summary
------------------------------------------------------------
Image ID                    :   20180830153222
Time of Collect             : 20180830153222
Dataset Classification      : OFFICIAL
Collection mode             : SPOTLIGHT
Collection Geometry         : Monostatic
Is SRP fixed                : Yes
Sensor                      : SARCSENSOR
CPHD file data type         : cmplxf
Number of channels          : 1
Channel 0 polarisation      : VV
   Azimuth 3dB beam width   : 0.031228
   Elevation 3dB beam Width  : 0.104095
Deskew applied?             : No
Tasked SRP                  : 10. 0. 0. (deg, deg, m)
Centre Azimuth is           : 270.0 deg
Centre Grazing angle is     :  35.3 deg
Centre Squint angle is      : 0.000000 deg
Centre range is             : 5000.0 metres
Number of pulses            :  2249
ADC Samples per pulse       :  2066
Chirp rate                  : 9.179e+13 Hz/sec
ADRate                      : 1.000e+08 Samps/sec
TX Pulse duration           : 0.020000 milliseconds
Look direction is           : 'Right'
Mean PRF                    :   100.0 Hz
Collection duration         :    22.5 secs
Transmit bandwidth          : 1835.747 MHz
Slant Range resolution      :   0.082 m
Ground Range resolution     :   0.100 m
Transmit Centre Frequency   :   9.600 GHz
Transmit Chirp start/stop   : 8682.127 - 10517.873 MHz
Centre wavelength           :   0.031 m
Synthetic aperture centre   : 6284715.,-4082.,1100746. metres
Velocity at aperture Centre :    50.0 m/s
Synthetic aperture distance : 1124.0 m
Finest azimuth resolution   :   0.069 m
Maximum range scene size    : 163.308 m
Illuminated area (az x ra)  :   156.1 x   901.6m
Wavelength                  : 0.031228 m
Slant range resolution      : 0.081654 m
Input Mesh has 13890 triangles
Processing Large Nodes...
```

```
[  0] active list size:  1 Smalllist size   0 nextlist size: 2
[  1] active list size:  2 Smalllist size   0 nextlist size: 4
[  2] active list size:  4 Smalllist size   0 nextlist size: 8
[  3] active list size:  8 Smalllist size   1 nextlist size:15
[  4] active list size: 15 Smalllist size   3 nextlist size:28
[  5] active list size: 28 Smalllist size   4 nextlist size:55
[  6] active list size: 55 Smalllist size  34 nextlist size:80
[  7] active list size: 80 Smalllist size  98 nextlist size:96
[  8] active list size: 96 Smalllist size 205 nextlist size:85
[  9] active list size: 85 Smalllist size 310 nextlist size:65
[ 10] active list size: 65 Smalllist size 394 nextlist size:46
[ 11] active list size: 46 Smalllist size 458 nextlist size:28
[ 12] active list size: 28 Smalllist size 505 nextlist size: 9
[ 13] active list size:  9 Smalllist size 522 nextlist size: 1
[ 14] active list size:  1 Smalllist size 524 nextlist size: 0
Done!
PreProcessing Small Nodes ...
Done!
Processing Small Nodes...
[  0] active list size:  524    nextlist size: 1032
[  1] active list size: 1032    nextlist size:  868
[  2] active list size:  868    nextlist size:  280
[  3] active list size:  280    nextlist size:   20
[  4] active list size:   20    nextlist size:    2
[  5] active list size:    2    nextlist size:    0
```

              KDTree Summary
==
  Triangles   : 13890
  Smallnode size   : 64
  Number of Nodes   : 3249
  Number of leaves (NL)   : 1625
  Number of non-empty leaves   : 1625
  Ave triangles / non-empty leaf : 16.8
  Expected Traversals   : 4.7
  Expected leaves visited   : 0.4
  Expected triangle intersections : 7.0

  Total tree traversal cost   : 209.8

RayTracing beam (Az,El): 2.38e-04degx6.49e-05deg (26.51x12.51 metres (ground))
Ray density   : 11.3 x 41.5 [11.3 x 24.0 ground plane] per metre
Ray separation   : 0.088 x 0.024 [0.088 x 0.042 ground plane] metres
EIRP   : 2.706015e+06 Watts (64.323302 dBW)
 [0%]>>>.. ETC 26/Oct/18 13:27
           Timing Summary
==
Time spent building rays   : 60974.06 ms
Time spent shooting rays   : 820146.20 ms
Time spent reflecting rays   : 2600.90 ms

```
Time spent building shadowRays    :   642.33 ms
Time spent packing the pulse      : 35195.50 ms
Time spent calculating PO Fields  :  9068.36 ms
Time data processing in loop      : 17493.54 ms
Number of loops                   :    2232 pulses
Total time spent on pulses        : 946120.90 ms
------------------------------------------
  Total time per pulse            :    17.66 ms
Done in  44.129047 seconds
Writing CPHD File "outputCPHDFile.sarc.cph".......Done
```

The folder should now contain a file called *outputCPHDFile.sarc.cph*. This is a CPHD file with all the wideband data content being completed by SAR-CASTIC. The next step is to convert this CPHD file into a SAR image using an image formation processor. This will be covered in the next section.

CHECKING THE RESULTS

To form a SAR image from the CPHD file requires an image formation processor (IFP). There are many variants of these and the relative merits are beyond the scope of this document. Fortunately the SARCASTIC distribution contains a very simple (but relatively efficient) SAR processor called TDPOCL. This is the subject of this section.

To run the IFP type :

```
$ tdpocl
```

You should then see something similar to Figure 71

The first question you will be presented with is to enter the name of the CPHD file that you want to form a SAR image from. Its worth noting here that TDPOCL forms a SAR image from any CPHD file, not just ones produced by SARCASTIC. Enter the name of the CPHD file you want to process here (or hit return to use the default).

Next you will be asked for the number of pulses that you wish to process. The input request will provide you with the number of pulses in the CPHD file that you entered. FOr the finest resolution use them all.

You will then be asked for the start pulse to process. Again for the finest resolution this should be '0'.

The next input requires you to decide whether to use a 'surface file' or not. The way that TDPOCL works is it first creates an array of ECEF coordinates that represent each pixel in the output image. This array is called a 'surface file'. The processor then handcrafts each pixel using the pulses specified and apply the appropriate phase corrections to each pulse for that pixel. If you answer 'N' or 'no' then a default focal plane surface will be generated that is a flat plane tangential to the Earth's surface at the stabilisation reference point in the CPHD file. In most cases you will just hit return or 'N'. It is worth recognising that this option allows for a focal surface that is not a flat plane to be used to form a SAR imagery, a useful feature when forming imagery over mountain ranges.

Figure 71: Running tdpocl

Next you will be asked for the scene centre of the image in latitude/longitude (decimal degrees) and altitude. By default the SRP of the CPHD file is used.

If a default focal plane has been asked for then the next questions relate to the pixel size of the output image. First the image pixel spacing in the along-track direction, then in the range direction. This is followed by the number of pixels in the along track direction, then in the range direction.

After this, the program asked for the name of the surface file to save to disk. This is so that subsequent runs can use the exact same surface file and therefore guarantee that the pixels in the images are co-located.

Finally the name of the output image file is requested. This should end with .DAT which is a very simple image file format.

The program output should look something like the following:

```
bash -3.2$ tdpocl

      Time Domain Processor OpenCL Variant (TDPOCL)
                   Version : 4.5 (Rev 247)
   Commit: 4.5-28-dc7a03f , 2018-10-05 17:56:11 +0100
       Copyright (c) 2014 [Dstl]. All rights reserved.

CPHD Filename [outputCPHDFile.sarc.cph]:
Number of pulses to process (1-2232) [2232]:
Start pulse [0]:
Use a surface file (or default focal plane for image) [N]:
Scene centre lat/long/alt [10 0 0]:
Image spacing width (m) [0.05]:
```

```
Image spacing height (m) [0.05]:
Image width (pixels) [1024]:
Image height (pixels) [1024]:
Surface filename [surface.dat]:
Sensor position       : 6284716.678815 -4082.704059 1100742.577312
Scene centre position : 6281872.829603 0.000000 1100248.547735
Azimuth angle 269.944, Grazing angle 35.26
Image Definition (lat deg, lon deg, alt m):
   9.99977,-0.00023,0.0  --------  10.00023,-0.00023,0.0
            |                       |
            |                       |
   9.99977,0.00023,0.0  --------  10.00023,0.00023,0.0
   Done 93.8% of the surface
Filename for image [image.dat]:
-----------------------------------------------------------------------
    OpenCL Device information
CL_PLATFORM_PROFILE         : FULL_PROFILE
CL_PLATFORM_VERSION         : OpenCL 1.2 (May 24 2018 20:07:03)
CL_PLATFORM_NAME            : Apple
CL_PLATFORM_VENDOR          : Apple
CPU DEVICES                 : 1
DEVICE                      : 0
  DEVICE NAME               : Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz
  DEVICE VENDOR             : Intel
DEVICES USED                : 1
LocalWorkSize set to : 0, 0
-----------------------------------------------------------------------
                    Device Memory Summary
Memory per Device                 : 65168 MByte
CPU memory for pulse compression : 65536 MByte
Number of Devices                 : 1
Number of threads                 : 24
Number of pulse blocks            : 1
Number of Pulses per block        : 2232
X surface elements                : 1024
X surface elements per device     : 1024
Y surface chunks                  : 1
Y surface rows per chunk          : 1024
Last chunk rows                   : 0
-----------------------------------------------------------------------
TX/RX to SRP ranges calculated in 0.000891 seconds
Processing pulses      0 -  2231 out of   2232 [0%]
  Device max local work size        : 128
  Device prefered warp size         : 1
  Calculated device work size       : 8,1
  Calculated device global data size : 1024,1024
  Performing RCS Correction...Done.
  Max RCS in scene is 301486.56 m^2 (   55 dBm^2)
Thinking completed in 15.029670 secs
  pixels                    : 1048576
```

```
   pixels per second            : 69767.07 Pixels/Sec
   Seconds per pixel            : 1.43e-05 Sec/Pixel
   Seconds per pixel per pulse : 6.42e-09 Sec/pixel/pulse
   Pulse pixels per second     : 155.72 MPPpS
The following images are still allocated:

There are 6 unused nodes in the memory linked list
```

The next step is to view the image. Again there are many different tools to view SAR images with and the discussion is beyond the scope of this document. The SARCASTIC distribution does include a very simple image viewer written in Python and using the OpenCV library. The image viewer can be found in the SCRIPTS folder and is called IMDISPLAY.PY.

To run the viewer just call

```
$ python imdisplay.py
```

If you run this on its own you will be presented with some useful usage information:

```
usage: imdisplay.py
    [-h] [-x XCENTRE] [-y YCENTRE] [-nx XSIZE] [-ny YSIZE]
    [-xs XSHRINK] [-ys YSHRINK] [-a ALPHA] [-b BETA] [-p]
    [-xo XOVERSAMPLE] [-yo YOVERSAMPLE]
      filename
imdisplay.py: error: too few arguments
```

The parameters should be pretty obvious but the ALPHA and BETA values require a few words of explanation.

When you run the programme with a .DAT file as an argument you will be presented with the magnitude of the values within the image and their corresponding display values, shown in the terminal. The values of ALPHA and BETA allow the contrast/brightness of the image to be adjusted. The output image values are given by $O = \alpha I + \beta$ where I is the input amplitude in the IMAGE.DAT file and O is the display value.

To see the image just type:

```
python imdisplay.py image.dat
```

The values in the output file are approximately those of the radar cross section of the target in metres squared. (The main source of error being that on physical optics calculations are used in SARCASTIC).

If everything has worked you should be presented with the image shown in Figure 72.

The SAR processor by default has increasing range direction down the image with the top of the image having the nearest range. If you click on the image and press the 'a' key then the magnitude of the pixel (before display mapping) will be printed out. To quite the display, click in the window and press the 'q' key.
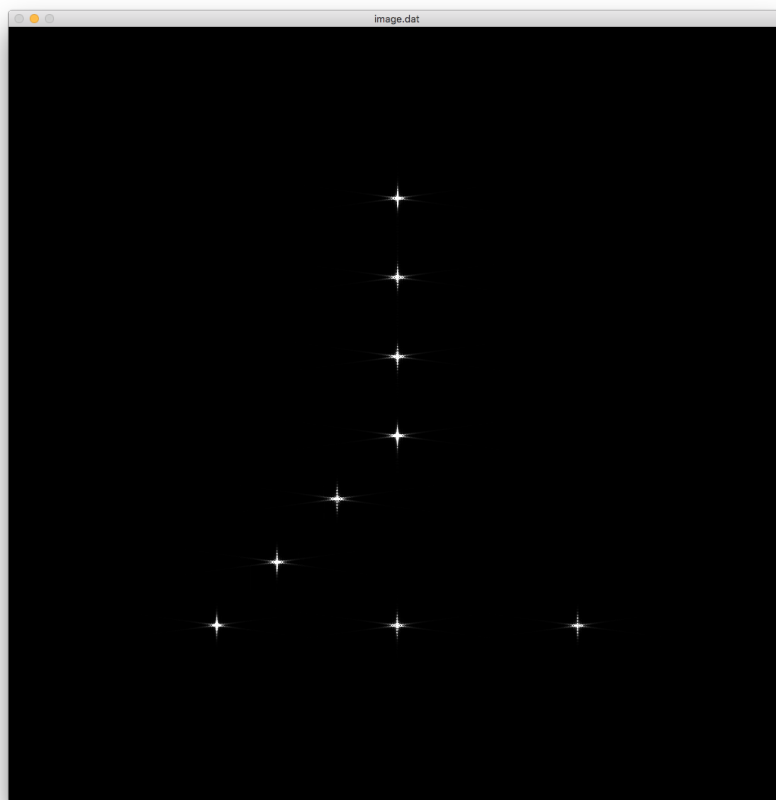
Figure 72: Simulated SAR image of the trihedral farm

8

SUMMARY

This document has provided a step-by-step tutorial to use the SARCASTIC distribution of tools to simulate a SAR image. The guide is not complete and there are many other tools that are available to interrogate the scattering mechanisms within a scene. Instructions to use these tools will be provided in a follow-on manual.