DolphiCam$^{\text{TM}}$ Research 1.3 User Manual

Fredrik Lingvall

April 20, 2015

Contents

1 Introduction

This document describes DolphiCam^{TM} *Research* version 1.3 of the DolphiCam^{TM} acoustic camera. DolphiCam^{TM} Research is a variant of the normal DolphiCam^{TM} with additional support for saving ultrasound sensor data and use user defined transmit pulses. DolphiCam^{TM} Research comes with a set of m-files for loading and analyzing data in Matlab or GNU Octave. DolphiCam^{TM} Research also comes with the DolphiCam^{TM} *DataAnalyzer* software which is stand-alone tool for analyzing and viewing DolphiCam^{TM} data as well as to export data to powerful 3D viewers such as ParaView [1].

2 Installation

Before installing the DolphiCam^{TM} Research Windows software one must first have installed the standard Windows software as described in the DolphiCam^{TM} user manual. Then the DolphiCam^{TM} Research can be installed by decompressing the zip-file (package) `DolphiCam_Research_Beta.zip` on a suitable location on the disk. The DolphiCam^{TM} Research package contains the following files and folders:

```
DolphiCam_Research_Beta -|
                         README.txt
                         dc_research_userman_xxxx.pdf
                         DolphiCam Research Software -|
                                                      <Windows sw and camera fw>
                         m_files -|
                                  Contents.m
                                  dc_data_viewer.m
                                  load_dc_data.m
                         example_data -|
                                       data_air_filtered.hdf5
                                       data_air_filtered.rdf
                                       data_air_unfiltered.hdf5
                                       data_air_unfiltered.rdf
                                       data_plexiglass_filtered.hdf5
                                       data_plexiglass_filtered.rdf
                                       data_plexiglass_filtered.vti
                                       data_plexiglass_unfiltered.hdf5
                                       data_plexiglass_unfiltered.rdf
                                       data_plexiglass_unfiltered.vti
                                       plexi_flower_wiener.vti
```

The Windows software and firmware files for the camera are located the in `DolphiCam Research Software` folder. There is currently no Windows installer for the DolphiCam^{TM} Research Windows software; the Windows application is simply started by running `DolphiCam Research Software/DolphiCam.exe`. After starting the `DolphiCam.exe` application one will be asked to install the Research firmware. When the firmware has been loaded into camera one should have the new Research features enabled (see Section 3).

The Matlab/Octave functions are located the in the `m_files` folder. To use these functions one needs set the Matlab or Octave path to the folder containing the m-files. The path can be set by, for example, adding

```
addpath(/home/my_user/dc_research_folder/m_files)
```

(replace the path above with the actual one where the m-files is located) to Matlab's `startup.m` file or Octave's `.octaverc` file.[1]

DolphiCam™ Research also comes with a set of example files which are located in the `example_data` folder. The files with the file extension `.rdf` can be loaded by the `load_dc_data` function by Matlab/Octave as described in Section 4. The files with extension `.hdf5` is generated by the stand-alone DolphiCam™ DataAnalyzer tool. These files can be loaded by Matlab and DolphiCam™ DataAnalyzer (or any other tool that can read HDF5 files). The DolphiCam™ DataAnalyzer is a post-procsssing/analyzing tool which comes with DolphiCam™ Research that can be downloaded from

http://www.dolphitech.com

at *Support→Software Download*. Furthermore, the DolphiCam™ DataAnalyzer can export data in the `.vti` VTK format. These files can then be opened in ParaView for advanced 3D visualization (see the DolphiCam™ DataAnalyzer user manual for more information). The included data files in the DolphiCam™ Research package is shown in Table 1.

File	**Description**
`data_air_filtered.rdf` `data_air_filtered.hdf5`	CF08, Pulse 1, transducer in air, filtered data
`data_air_unfiltered.rdf` `data_air_unfiltered.hdf5`	CF08, Pulse 1, transducer in air, unfiltered data

`data_plexiglass_filtered.hdf5`
`data_plexiglass_filtered.hdf5`
`data_plexiglass_filtered.vti`
	CF08, Pulse 1, plexiglass flower, filtered data

`data_plexiglass_unfiltered.hdf5`
`data_plexiglass_unfiltered.hdf5`
`data_plexiglass_unfiltered.vti`
	CF08, Pulse 1, plexiglass flower, unfiltered data

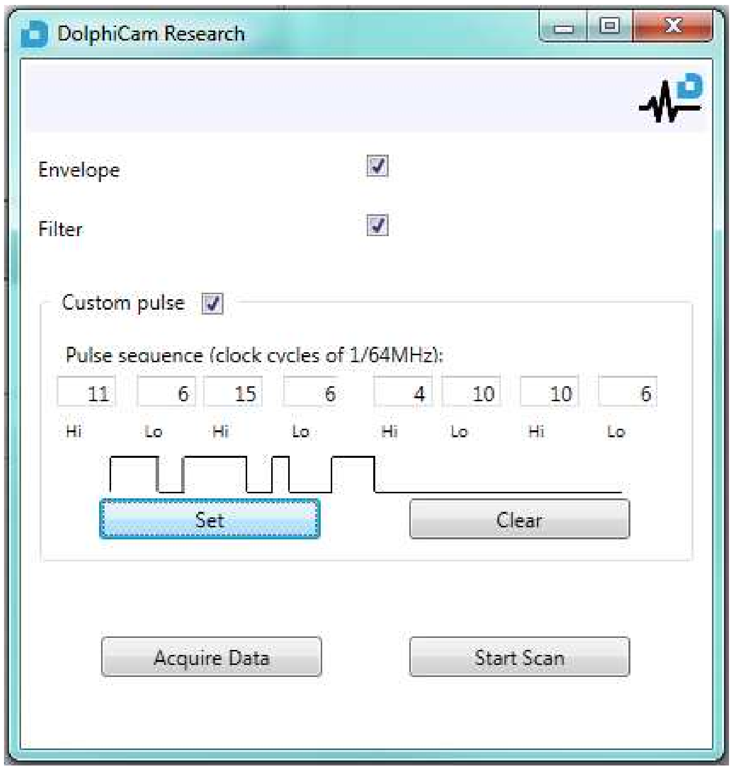
`plexi_flower_wiener.vti`	CF08, Pulse 1, processed with the Wiener filter

Table 1: Example data files included in DolphiCam™ Research.

[1]For Matlab one can also use the corresponding menu to add the path.

3 New GUI Elements for the DolphiCam^{TM} Research

The DolphiCam^{TM} Research Windows application has a new window for acquiring data in its graphical user interface (GUI) which is shown in Figure 1.



Envelope ✓

Filter ✓

Custom pulse ✓

Pulse sequence (clock cycles of 1/64MHz):

11	6	15	6	4	10	10	6
Hi	Lo	Hi	Lo	Hi	Lo	Hi	Lo

Figure 1: DolphiCam^{TM} Research GUI.

The Research window is activated from the *Tools* menu by selecting the *Research features* item and it has the following elements:

Envelope The *Envelope* check box switches the envelop on/and off in the A-scan and the two B-scans.

Filter The *Filter* check box switches filtering on and off when acquiring data. Note that the filters can only be one of the 8 (CF08) or 4 (CF16) pre-defined ones.

Custom pulse The *Custom pulse* editor has 8 fields where one can specify the number of clock cycles a 15.625 ns (1/64 MHz) for each high and low of the pulse sequence. One can define a maximum of four pulses in the pulse sequence where the allowed range for high parts are 4–15 clock cycles and 6–15 clock cycles for the low parts, respectively. The *Set* button programs the camera with the new pulse sequence and the *Clear* button clears all the fields in the pulse editor (NB. only the editor fields are cleared, not the pulse sequence loaded in the camera).

Acquire Data The *Acquire Data* button switches off scanning and starts the data acquisition. A pop-up window will be shown during the data acquisition and when the data is acquired the new file is stored in the

```
C:\Programdata\DolphiTech\DolphiCam\RawData
```

folder.

Start Scan The *Start Scan/Stop Scan* button toggles live scanning.

Note: It takes $\approx$ 12 seconds to acquire a CF08 data set and $\approx$ 24 seconds to acquire a CF16 data set, respectively. During this time the camera should not be moved otherwise there will be distortions in the acquired data.

4 Importing and Viewing DolphiCam Data in Matlab/Octave

The DolphiCam^{TM} Research comes with a set of m-files for reading and viewing ultrasound data in Matlab or Octave.

4.1 load_dc_data

The `load_dc_data` function reads ultrasound data in the `.rdf` (text) format. The function takes a string with the `.rdf` file name as input and returns the ultrasound data in a matrix

```
>> Y = load_dc_data('DolphiCam 13430025 2015-01-27 13-04-45.rdf');
```

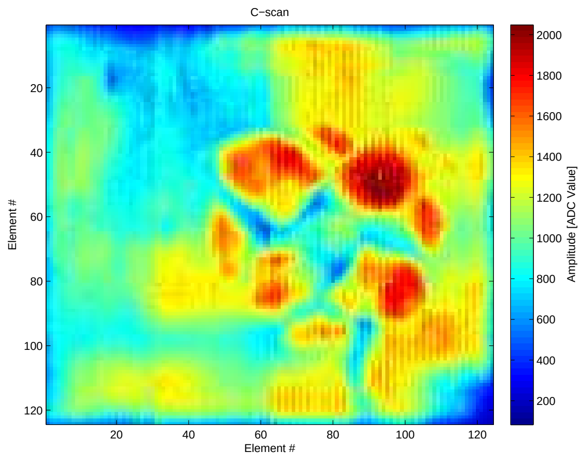
The output matrix as an $L \times 124^2 + 1$ matrix where L is the A-scan length and 124^2 is the total number of transducer elements on the 2D array. The first column in the output matrix contains a sample index vector and remaining columns contain ultrasound data for the corresponding array elements.

4.2 dc_data_viewer

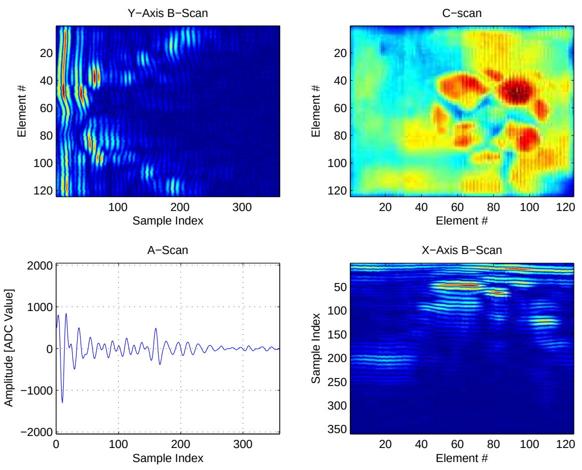
The `dc_data_viewer` function can be used for quick viewing of the data. It takes a matrix in the format returned by the `load_dc_data` function,

```
>> dc_data_viewer(Y);
```

and presents two windows of the data. In *Window 1* one first selects the position in a C-scan with the mouse (clicks) and then *Window 2* shows a vertical and a horizontal B-scan as well as an A-scan according to the position that was selected in *Window 1*. Figure 2 shows an example (an impact damage in a 6mm CFRP plate) of using the `dc_data_viewer` function.



(a) Window 1.



(b) Window 2.

Figure 2: Illustration of the `dc_data_viewer` function.

A File Header Description

A.1 `DataDimension`

`DataDimension` is a 6 element vector which describes the dimension of the data. The first four numbers describe element layout of the 2D array the (it's a 124×124 element sensor) and the last two numbers describe the A-scan length of the data (*i.e.*, start and stop indexes of the A-scan).

Example:

```
0 0 123 123 0 359
```

A.2 `TimeStamp`

`TimeStamp` is a string with the time stamp when the file was created by the DolphiCam^{TM} Windows application.

Example:

```
2015-01-21_15-12-09
```

A.3 `CameraSerialNumber`

is a string with the serial number of the camera.

`CameraSerialNumber`

Example:

```
13060013
```

A.4 `SoftwareVersion`

`SoftwareVersion` is a string with the software version used.

Example:

```
1.3.3450
```

A.5 `IsAssumedToBeInitialized`

`IsAssumedToBeInitialized` is a Boolean for internal use only.

Example:

```
False
```

A.6 `CurrentSettingsFileName`

`CurrentSettingsFileName` is a string with the name of the current settings file if any.
Example:

```
Default
```

A.7 `FpgaVersion`

`FpgaVersion` is an integer with the major version of the FPGA firmware.
Example:

```
135
```

A.8 `FpgaSubVersion`

`FpgaSubVersion` is an integer with the minor version of the FPGA firmware.
Example:

```
13
```

A.9 `HardwareVersion`

`HardwareVersion` is an integer with model type of the camera. A `HardwareVersion` < 10 is a CF08 camera and a `HardwareVersion` $>= 10$ is a CF16 camera.
Example:

```
10
```

A.10 `HoldOffTime`

`HoldOffTime` is an integer with the number of samples from the start of the transmit pulse to the start of the data acquisition.
Example:

```
360
```

A.11 `AmplitudeGatingStart`

`AmplitudeGatingStart` is an integer with the start sample index of the 1st amplitude gate (for future use).
Example:

```
50
```

A.12 `AmplitudeGatingEnd`

is an integer with the end sample index of the 1st amplitude gate (for future use).

`AmplitudeGatingEnd`

Example:

```
350
```

A.13 `TimeOfFlightGatingStart`

`TimeOfFlightGatingStart` is an integer with the start sample index of the 1st time-of-flight gate (for future use).

Example:

```
50
```

A.14 `TimeOfFlightGatingEnd`

is an integer with the end sample index of the 1st time-of-flight gate (for future use).

`TimeOfFlightGatingEnd`

Example:

```
350
```

A.15 `AmplitudeGatingStart2`

`AmplitudeGatingStart2` is an integer with the start sample index of the 2nd amplitude gate (for future use).

Example:

```
0
```

A.16 `AmplitudeGatingEnd2`

`AmplitudeGatingEnd2` is an integer with the end sample index of the 2nd amplitude gate (for future use).

Example:

```
0
```

A.17 `TimeOfFlightGatingStart2`

`TimeOfFlightGatingStart2` is an integer with the start sample index of the 2nd time-of-flight gate (for future use).

Example:

```
0
```

A.18 `TimeOfFlightGatingEnd2`

`TimeOfFlightGatingEnd2` is an integer with the end sample index of the 2nd time-of-flight gate (for future use).

Example:

```
0
```

A.19 `AmplitudeGatingStart3`

`AmplitudeGatingStart3` is an integer with the start sample index of the 3rd amplitude gate (for future use).

Example:

```
0
```

A.20 `AmplitudeGatingEnd3`

`AmplitudeGatingEnd3` is an integer with the end sample index of the 3rd amplitude gate (for future use).

Example:

```
0
```

A.21 `TimeOfFlightGatingStart3`

`TimeOfFlightGatingStart3` is an integer with the start sample index of the 3rd time-of-flight gate (for future use).

Example:

```
0
```

A.22 `TimeOfFlightGatingEnd3`

`TimeOfFlightGatingEnd3` is an integer with the end sample index of the 3rd time-of-flight gate (for future use).
Example:

```
0
```

A.23 `TxLine`

`TxLine` is an integer describing the current horizontal cross hair position (for future use).
Example:

```
31
```

A.24 `RxLine`

`RxLine` is an integer describing the current vertical cross hair position (for future use).
Example:

```
31
```

A.25 `CameraModeSwitch`

`CameraModeSwitch` is a string with the position of the of the mode switch on the camera.
Example:

```
Normal
```

A.26 `CameraType`

`CameraType` a string with the type of the DolphiCam™.
Example:

```
Expert
```

A.27 `IsTimeCalibrationActive`

`IsTimeCalibrationActive` is a Boolean indicating whether time calibration is used or not (for internal use only).
Example:

```
True
```

A.28 IsScanButtonRegisterSet

`IsScanButtonRegisterSet` is a Boolean indicating the status of an internal register (for internal use only).

Example:

```
False
```

A.29 IsScanButtonPushed

`IsScanButtonPushed` is a Boolean indicating if the scan button has been pressed on the camera (for internal use only).

Example:

```
False
```

A.30 IsLowBattery

`IsLowBattery` is a Boolean which indicates if the battery was low when the data was acquired.

Example:

```
False
```

A.31 NumberOfTransmittingElements

`NumberOfTransmittingElements` is an integer with the number of transmit elements used when acquiring the data.

Example:

```
4
```

A.32 AveragesPerTransducerElement

`AveragesPerTransducerElement` is an integer with the number of averages used when acquiring the data.

Example:

```
2
```

A.33 `CaptureMethod`

`CaptureMethod` is one of the following strings

```
CaptureGreatestAbsoluteValue
CaptureHighestValue
CaptureLowestValue
```

which indicates the method used to create the C-scan.

A.34 `IsEnteringSleepMode`

`IsEnteringSleepMode` is a Boolean which indicates that the camera will sleep in the number seconds defined by `SleepModeWarningTime` which normally is 15 seconds (see also Section A.52).

Example:

```
False
```

A.35 `MatchFilterDivideFactor`

`MatchFilterDivideFactor` is an integer used internally when filtering the data.

Example:

```
512
```

A.36 `TxPulseShapeAndMatchFilter`

`TxPulseShapeAndMatchFilter` is an integer defining the pulse (and the corresponding filter) used when acquiring the data (1–8 for a CF08 camera and 1–4 for a CF16 camera).

Example:

```
1
```

A.37 `BScanXCoord`

`BScanXCoord` is an integer describing the current horizontal cross hair position (for future use).

Example:

```
62
```

A.38 `BScanYCoord`

`BScanYCoord` is an integer describing the current horizontal cross hair position (for future use).

Example:

```
61
```

A.39 `IsTimeCorrectedGainActive`

`IsTimeCorrectedGainActive` is a Boolean which indicates if time-corrected gain is active.

Example:

```
False
```

A.40 `IsTimeCorrectedGainSubpointsActive`

`IsTimeCorrectedGainSubpointsActive` is a Boolean which indicates if sub points of time-corrected gain is active.

Example:

```
False
```

A.41 `TimeCorrectedGainValues`

`TimeCorrectedGainValues` is an integer vector with the values of the internal digital-to-analog converter (DAC) used for time-corrected gain.

Example:

```
30 31 32 33 34 35 36 37 38 39 40 41 42 43
```

A.42 `TimeCorrectedGainValuesSubPoints`

`TimeCorrectedGainValuesSubPoints` is an integer vector with the values of the DAC used for time-corrected gain sub-points.

Example:

```
30 31 32 33 34 35 36 37 38 39 40 41 42
```

A.43 `DacValue`

`DacValue` is an integer used when setting the internal DAC for adjusting the sensor gain. The gain is set using two integers `DacValue` and `DacOffsetValue` (see Section A.44). The DAC value = `DacValue` + `DacOffsetValue`.

Example:

```
30
```

A.44 `DacOffsetValue`

`DacOffsetValue` is an integer used when setting the internal DAC for adjusting the sensor gain (see also Section A.43).

Example:

```
6
```

A.45 `RecordingWindowSamples`

`RecordingWindowSamples` is an integer describing the A-scan length in samples.

Example:

```
360
```

A.46 `IsRawDataRecording`

`IsRawDataRecording` a Boolean that indicates whether we record data or not (it should always be true).

Example:

```
True
```

A.47 `FilterEnable`

`FilterEnable` a Boolean that indicates whether the data is filtered or not.

Example:

```
True
```

A.48 EnablePulseSequence

`EnablePulseSequence` a Boolean that indicates whether a custom pulse is enabled or not.

Example:

```
False
```

A.49 TxPulseSeqA

`TxPulseSeqA` is the 2 byte integer which defines the first two pulses in of the pulse sequence definition.

Example:

```
17990
```

which is

```
0x4646
```

in hexadecimal form. The first nibble (from left) is the number of clock cycles the pulse is high and the second nibble is the number of cycles the pulse is low. The format of the nibbles are (`H` - high, `L` - low):

```
0xHLHL
```

The valid range of `H` is `0x00`,`0x04`–`0x0F` (0,4–15) and the valid range of a range of `L` is `0x00`,`0x06`–`0x0F` (0,6–15). If all `H` are > 0, also in `TxPulseSeqB` (see Section A.50), then four pulses are used.

A.50 TxPulseSeqB

`TxPulseSeqB` is the 2 byte integer which defines the last two pulses in of the pulse sequence definition. (see Section A.49 for a description of the format).

A.51 EnableSleepMode

`EnableSleepMode` a Boolean that shows if the sleep mode feature is enabled.

Example:

```
True
```

A.52 SleepModeWarningTime

`SleepModeWarningTime` is the number of seconds that the sleep mode warning is shown before the camera enters sleep mode.

Example:

```
15
```

A.53 CameraScanTime

`CameraScanTime` is the number of minutes before the camera enters sleep mode.

Example:

```
15
```

A.54 DcCorrection

`DcCorrection` is a Boolean that indicates whether DC correction (bias correction) is on or not. It should always be enabled.

Example:

```
True
```

References

[1] Paraview. http://www.paraview.org.