Decoding Sonavision Data Files

Chris Moorhead
University of Aberdeen
Department of Computing Science, Meston 215
Sonavision *c.moorhead.18@abdn.ac.uk*

July 2019

Contents

1 Introduction

The following is a description and explanation of the files generated using the Sonavision software for interpretation and replay by the associated software. The raw binary data can be imported into Python and viewed as pairs of hexadecimal digits each of which represents one byte (8 bits) of binary data where $00000000 \equiv 00 \equiv 0$ and $11111111 \equiv FF \equiv 256 = 16^2$. Some of the content (notably the text portions) can be viewed via a hex editor but the majority of the content is not interpretable by this means. As a result, experimentation was needed to assign meaning to the remaining data. This was done using the methods described fully in section 3.

Example Hex Code

19 53 56 20 53 4f 4e 41 52 20 44 41 54 41 20 56 65 72 73 69 6f 6e 20 31 2e 30
00 00 00 00 06 00 00 00 01 00 00 00 00 00 00 00 00 80 56 40 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 00 00 02
00 00 00 01 00 00 00 06 57 47 53 20 38 34 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 01 00 00 00 19 54 72 61 6e 73 76 65 72 73 65 20 4d 65 72 63 61 74
6f 72 20 28 55 54 4d 29 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 80 84 1e 41 00 00 00 00 00 00 00 00 78 9c a2 23 b9 fc 19 53 56 20 53 4f
4e 41 52 20 44 41 54 41 20 56 65 72 73 69 6f 6e 20 31 2e 30 00 00 00 00 06 00
00 00 01 00 00 00 00 00 00 00 00 80 56 40 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 00 00 02 00 00 00 01 00 00
00 06 57 47 53 20 38 34 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00
00 00 19 54 72 61 6e 73 76 65 72 73 65 20 4d 65 72 63 61 74 6f 72 20 28 55 54
4d 29 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 84 1e 41
00 00 00 00 00 00 00 00 78 9c a2 23 b9 fc ef 3f 01 00 00 00 00 00 00 00 00 f0
3f 06 57 47 53 20 38 34 00 00 00 40 a6 54 58 41 88 6d 74 96 1d a4 72 40 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 01 00 00 58 02 01 00 00 00 00 00 00 00 00 04 01 03 00 00 0a 00 00 01 01
05 90 01 00 00 c8 05 00 00 5a 00 00 00 e8 03 64 00 b8 0b b0 04 1f 04 04 00 00
00 00 00 00 00 00 00 01 00 00 00 c9 00 a2 5c 94 03 00 00 e8 03 90 01 0a 11 19
11 11 31 58 7f 7f 67 48 21 11 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a
0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a 0a
0a 0a 0a 0a 0a 0a 0a

Example of Hex Editor Interpretation

.SV SONAR DATA V ersion 1.0..... .....V@.. ....WGS
84.... ..... .Transverse Merc ator (UTM)..... ..... .A.....x..#..
.?.....?.W GS 84...@.TXA.mt ...r@..... .....

Note that the second is a direct interpretation of the first. When viewing on Python, we translate the hex digit pairs to integers, with each pair representing a number from 0 to 255. This is simply because it is more intuitive to interpret for a human.

2 Format of Files

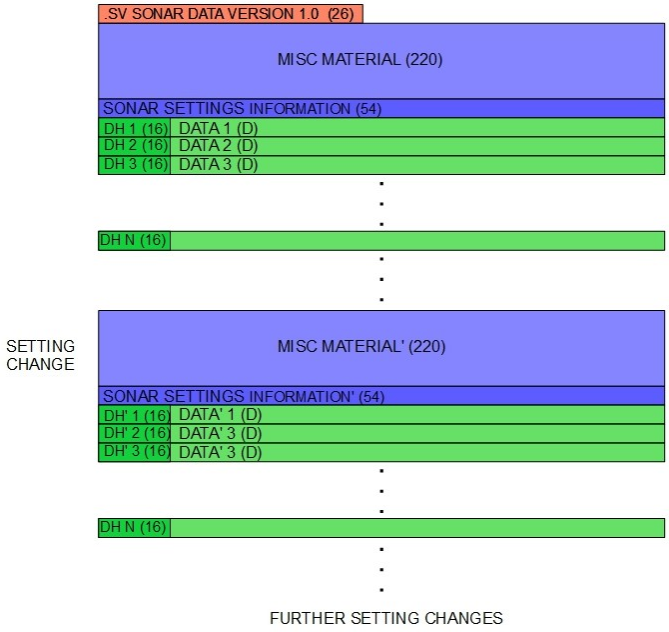

Figure 1: Layout of the Interpreted Data

Figure 1. shows how the data is stored in diagrammatic form. We will explain the format in more detail in this section. The numbers in brackets refer to the length of the data used to store that information in bytes.

Title Header (26)

Every file has a portion that begins with the binary encoding of the text ".sv sonar data version 1.0".

Non-Recording Data (274)

The next portion contains data that is independent of the scan made. It contains no information about the readings obtained by the sonar device i.e. it is defined by the uplink process only. This comes in two sections.

- **Miscellaneous Material (220)**: Some of this is text, but it does not seem to change with the settings of the sonar device. There is some evidence that it is hardware-based information. One portion of this block translates as text ".Transverse Mercator (UTM)" and twice we see ".WGS 84". Transverse Mercator is a means of mapping between coordinate systems and likely refers to some code that is necessary for replaying the files. Regardless, it does not appear to have relevance in the how the recordings

are made, or the data that is stored.

- **Setting Data (54)**: This portion contains numerical data relating to the settings of the sonar device set via the Sonavision software.

Setting Data Decryption

A complete list of the known information is as follows (positions within the Non-Recording Data given in square brackets). They correspond to variables set by the user before or during the recording. All units are given below.

- Header Sequence [1-8]: The seq [0, 0, 0, 0, 6, 0, 0, 0, 1] marks the beginning of this portion.
- Sonar Frequency, Hz [221-222]
- Resolution, bits [232]
- Speed Mode [233]: {0: "Super Fast, 1: "Fast, 2: " Normal", 3: "High Resolution"}
- Scan Type [234]: {0: "Stop", 1: "Scan", 2: FlyBack, 3: "Revolution"}
- Range, m [237]
- Percentage Gain [239]
- Number of Data Points [243-244]
- Head Offset, deg [245-255]: Sets where the zero mark of the desplay withn the software compared to the actual scanner.
- Speed of Sound, m/s [247-248]: Fixed at 1480.
- Sector Centre, deg [249-250]: In the Rotate and Stop modes we interpret this as the start position in degrees
- Sector Size, deg [251-252]: The size of the interval over which we are scanning, centred at the Sector Centre.

The significance of some other data is obtained from the Uplink Telegram provided by Sonavision.

- Start range, cm [253-254]: Inferred from the fact the the next to bytes are...
- End range, cm [255-256]
- Unit Head Transmission Frequency, Hz [261-262]: This is double the Sonar Frequency.
- Unit Head Reciever Frequency, Hz [263-264]: This is the Sonar Frequency + 455 Hz.
- Wedge Size, data points: 1,2,4,8 from slowest to fastest speed. This sets the sample rate of the data that defines all values in a wedge of that size.

Where the positions are allocated one byte, the number stored can only range between 0 and 256. If a larger range is needed, then two bytes are needed and the convention is that the least significant byte is in the first position with the most significant byte in second position. eg. The speed of sound is stored at 1480 m/s which is found in positions 247 and 248 as 200 5 $\equiv 200 + (256 \times 5)$

There is some redundancy here as the sonar range appears in two places in different units. Similarly there are calculations that are made on the head frequency that are stored elsewhere. The wedge size is also dependent on the speed of the scan via the formula $W = 2^{S-1}$. As we will also see, some of this information is also replicated in the Data Headers.

The modes are shown in Figure 2.

Recorded Data (n x (16 + D))

n is the number of time steps recorded and D is the number of data points per step set by the setting data in the previous block. Every discrete time step is recorded in the following format.

- **Data Header (16)**: Some information about when and the size of the following data.
- **Data at Current Time Step (D)**: This consists of the data recorded according to the specifications in the setting data in the Setting Data section.

Interpretation of Data Header

The data header breakdown by byte position in the header is as follows:

1 : 1 (fixed)

2 : 0 (fixed)

3 : 0 (fixed)

4 : 0 (fixed)

5-6 : "Time value" that increases as the recording progresses.

7-8 : "Date value"

9-10 : "Angle value". On a scale of 0-1023. Every increment of 1 is 360/1024 deg or approx. 1/3 of a degree.

11-12 : "Start Range", again inferred from the fact that this is usually fixed at zero and the following is...

13-14 : "Range Limit" in cm.

15-16 : "Number of Data Points", D

As mentioned previously, there is a large amount of redundancy here that cannot as yet be explained. The start and end points of the scan never change and are

entirely dependent on the speed of the scan as defined in the previous block. The first four entries also never change.

The exact nature of the "time" and "date" entries are not deciphered. Multiple methods of interpretation were tried, but nothing meaningful resulted. See Section 3 for more explanation.

The only part of this header that is of use for interpretation is the angle value and so this is all we extract when constructing our images from the data.

Interpretation of Data

For the data itself, the values recorded are dependent on previous setting. We can record the exact same area with the same objects and we will get widely different values for the data (assuming noise is negligible). These values are dependent on the percentage gain and resolution previously set by the user. It is very important to note that these settings fix the numerical values of the data and that incorrect settings can result in a loss of information between the signals received by the device and those recorded. Here are some features of the data.

- **Lower Threshold**: When a sonar device receives a signal, there is a minimum threshold above zero that establishes a point above which we can determine that signal is a real response due to an object and not just noise. For the Sonavision software, this value is 10 (and seems to be fixed as such). Any received values below this value in the range 0-9 are automatically given the value 10 and stored as this in the data.
- **Upper Threshold**: Likewise, the upper threshold sets a cap on what value the signal can obtain. This is set (and also seems to be fixed) at 127 and is the saturation level of the interpreted image.
- **Percentage Gain**: Although called percentage gain, this is not a linear increase. The percentage gain is based on decibels and is calculated on the received values before the lower threshold is applied. The upper threshold is applied afterwards. Thus the increase in the stored values is exponential for a linear increase in gain and small error measurements at zero gain are likewise exponentially increased. Appendix A shows this behaviour in more detail.
- **Permissable Values**: The data stored can only take on a finite number of values that are determined by the resolution of the recording (in bits). The formula that determines the set of permissible values is:

$$\mathtt{P} = \left\{ 10 + \left\lfloor \frac{118i}{2^b - 1} \right\rfloor \middle| 0 \leq i \leq 2^b \right\}$$

  This equates to splitting the interval 10-127 into 2^b subintervals and allocating the measured values into one of these. For 7 bits, there are only 117 possible values.

Further Setting and Data Blocks

If a setting is changed by the user using the software while the scan has already begun, it will cause the software to create a new block of non-recording data. The pattern above repeats for each time a new setting is instructed and is marked by the header sequence [0, 0, 0, 0, 6, 0, 0, 0, 1].

3 Methodology

From the outset, the significance of none of the data was known. In order to extract meaning from the hex data, the sonar device was set to stop mode in a position that it scanned empty space. Thus we should expect the data to be close to zero at every time step, barring small fluctuations due to noise in the measurements. The first observation after comparing several files was that there was a block of data followed by sections that exhibited some kind of periodicity. The first block was always 330 bytes and the period of the other data was found to be 416 bytes. Checking the data slices at this periodicity gives initial portions as follows as in Figure 3.

The initial observations are that the first four digits are constant, the fifth increases monotonically but not with any regularity, some digits appear to have a minor variance around an unknown mean and those that we expect to be 0 (empty space in the scan) are in fact recorded as 10.

```
[1, 0, 0, 0, 161, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 13, 17, 17, 15, 12, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 16, 17, 15, 11, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 15, 12, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 13, 17, 17, 15, 12, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 15, 12, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 13, 17, 17, 15, 11, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 15, 11, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 15, 11, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 16, 17, 15, 12, 10, 10]
[1, 0, 0, 0, 162, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 16, 12, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 16, 12, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 16, 17, 15, 11, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 13, 17, 17, 16, 12, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 15, 12, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 15, 12, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 15, 11, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 12, 17, 17, 15, 11, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 13, 17, 17, 16, 12, 10, 10]
[1, 0, 0, 0, 163, 99, 167, 92, 172, 3, 0, 0, 232, 3, 144, 1, 13, 17, 17, 15, 12, 10, 10]
```

Figure 2: Initial sequences of the periodic portions of the data file

To test this further, the mean and variances of the first 90 digits in the sequence were calculated over the first 1000 periods. If the digits were in fact constant over all the periods then we would expect that the variance be zero for that position in the sequence. Small variances can be attributed to the natural noise characteristics of the recorded data and large variances towards wildly differing or increasing/decreasing values in the data over the periods analysed. All this confirmed the initial observations but also revealed that the fifth position in the sequences is not in fact constant but slowly increasing.

A series of experiments were then performed in which only one of the following variables was modified:

- Sonar range - 2, 4, 6, 8, 10, 15, 20 metres
- Head frequency - 600, 800 or 1000 Hz
- Scan speed - High Resolution, Normal, Fast, Super Fast
- Resolution in bits - 1-7 bits
- Gain - 0 to 100 in steps of 10

The experiments and their conclusions can be summarised as follows.

Experiment 1: Range and Scan Speed

The fixed variables were gain (75), resolution (7 bit), head freq (500 Hz) and angle (290 deg). Making recordings at each range and scan speed gives 28 files. The assumption is that the first 300 bytes relate to settings, so we look at this portion for each recording and measure any changes between recordings. All positions that are not constant over the recordings are presumed to be caused by changes in range and scan speed alone. This resulted in 7 positions of interest which were interpreted as follows:

- Position 291: Dependent on the scan speed with the following key:

  $$\{\texttt{highres} = 1, \texttt{normal} = 2, \texttt{fast} = 4, \texttt{superfast} = 8\}$$

  This corresponds to the size of wedge that each time step is spread over.

- Position 259: Also dependent on scan speed with the following key:

  $$\{\texttt{highres} = 3, \texttt{normal} = 2, \texttt{fast} = 1, \texttt{superfast} = 0\}$$

  This corresponds to $log_2(\texttt{SweepAngle})$.

- Position 263: This corresponds to the range of the scan. Each entry corresponds exactly to the range in metres.

- Position 281: Similarly, this corresponds to range, although the function mapping range in m to this values is at first unclear:

  $$\{\texttt{2m} = 200, \texttt{4m} = 144, \texttt{6m} = 88, \texttt{8m} = 32, \texttt{10m} = 232, \texttt{15m} = 220, \texttt{20m} = 220\}$$

- Position 282: Likewise with this line, again the function is unclear:

  $$\{\texttt{2m} = 0, \texttt{4m} = 1, \texttt{6m} = 2, \texttt{8m} = 3, \texttt{10m} = 3, \texttt{15m} = 5, \texttt{20m} = 7\}$$

  When we combine both together using the formula:

  $$r = 256 Pos_{282} + Pos_{281}$$

  Where r is the range of the scan in centimetres i.e. these two bytes together give a little endian representation of the range in centimetres.

- Position 269: Only has two possible values, 144 and 200. When at superfast speed, this is 200 and for any other speed it is 144.
- Position 270: Similarly, when at superfast speed, this is 0 and 1 for any other speed. What the function of this is unknown.

Due to the representation of range in centimetres being given over 2 bytes with the least significant byte first, the values of the fixed variables were converted into this form and these values searched for in the initial sequence of one of the recorded data files. These positions being found, they were verified as being in the same location in the remaining 27 files. The gain and resolution only require one bit to store their value but were similarly located and verified. Lastly, we also made calculations on the speed of sound and quantities described in the sonar telegram provided by Sonavision that are dependent on head frequency to convert it into the least significant bit form.

- Speed of Sound: Fixed as $1480 \equiv 200\ 5$. Note that this is a good estimation of the speed of sound in fresh water. The speed of sound in sea water is slightly higher, around 1530 m/s. This is something that can be altered by the user for the purposes of measuring distances more accurately.
- U/W Unit Head Transmit Frequency: 2 * head freq $= 1000 \equiv 232\ 3$
- U/W Unit Head Recieve Frequency: head freq + 455 $= 955 \equiv 187\ 3$

These were located and verified as above.

Periodicity of Data

After this first experiment, an investigation of the periodicity of the data portion of the file was made. It was previously noted (Figure 3) that the period was 416 bytes, but it seems highly likely that the number of data points would be divisible by 10. Note that there is no particular reason to assume that the number of data points recorded at each time step is not dependent on a number of factors. The assumption is that the first 16 data points relate to something other than the recorded data and the rest relate to the received intensities of the sonar device. On inspection, it was found that the periodicity was constant at 416 over all speed and range setting with the exception of super fast speed where it was 216. This reinforced the hypothesis that the first 16 points relate to something else. It also explains the reason why positions 269 and 270 in the initial portion of each file differs only in the super fast setting. When we translate these numbers to a single integer using the reverse of the above method, we find that they evaluate as 200 for super fast and 400 for all other speeds. We consider that the original hypothesis was therefore true and the problem solved subject to verification by later measurements.

Function of the Periodic Header

Since we have determined the number of data points relating directly to measurements made by the sonar, this reveals a further question. What is the

significance of the 16 digits at the beginning of each period? A quick investigation shows that the range in cm and the number of data points appears here (as described in the previous section).

The fifth digit increases but not regularly. Observation of further files shows that if it increases beyond 255 (the maximum value that it can attain), the sixth digit also increases by one. Since this increase is not regular it is possibly related to a time element since the number of data points collected in a given time period will experience some natural variation. Various means of trying to understand what the number stands for specifically but not successfully, including time since the start of recording, time of day measured from midnight and converted into second or minutes and time since start of midnight on 1st Jan. Converting both values in base 256 into an integer gives 20711.

Experiment 2: Offset Angle and Fixed Angle

For this experiment, we changed only two variables. The fixed angle is the sector centre in stop mode and gives the angle of the scanner. The offset angle determines which bearing is designated the zero position. The recordings were made with fixed angle from 0 to 315 degrees at 45 degree intervals for at offset 0 degrees and with fixed angles of 0 to 180 degrees for offsets of 90, 180 and 270 degrees. This gave a total of 23 recordings with the main purpose of verifying our previous hypothesis. Some code was written to extract this information with the following output. The first line is the .dat filename named according to the fixed and offset angles.

```
fwd normal g=0.dat
Sonar Freq:  500 Hz
Resolution:  7 Bit
Scan Speed:  Normal
Data Points per Scan:  400
Scan Mode:  Rotate
Range in m:  4
Range in cm:  400
Percentage Gain:  0
Start Angle:  290 deg
Head Offset:  0 deg
Speed of Sound:  1480 m/s
```

```
fwd normal g=10.dat
Sonar Freq:  500 Hz
Resolution:  7 Bit
Scan Speed:  Normal
Data Points per Scan:  400
Scan Mode:  Rotate
Range in m:  4
Range in cm:  400
Percentage Gain:  10
Start Angle:  290 deg
Head Offset:  0 deg
Speed of Sound:  1480 m/s
```

Figure 3: Reading variables from data files

Experiment 3: Gain Changes and Mode

This experiment had two parts. For the first, the sonar was used in stop mode and in the second it was used in rotate mode. For both, the gain was varied between 0 and 100 in steps of 10. In the first, we can examine how the intensity of pixels in the same location, in this case 14 consecutive pixels of a scan in stop mode with

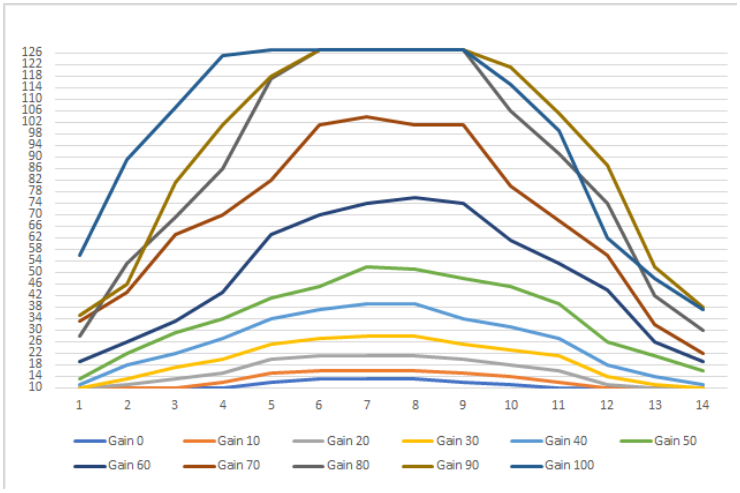

Figure 4: Increase in intensity with gain for a short sequence of consecutive pixels

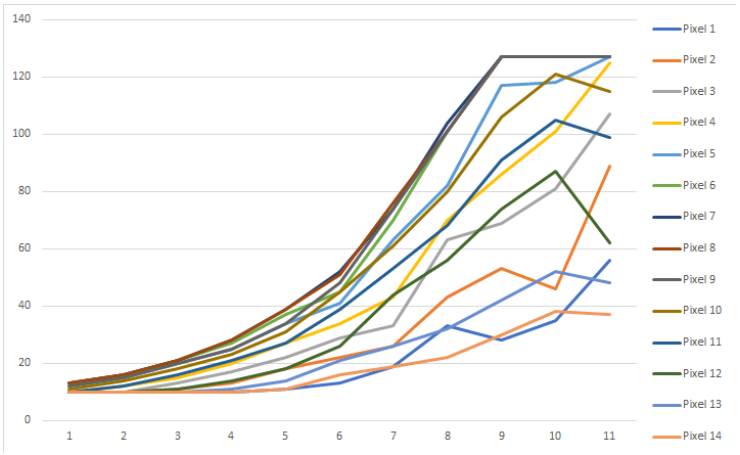

Figure 5: Intensity increase per pixel as gain increases

Figure 4 demonstrates the upper and lower thresholds for the intensity data as well as how the increase in intensity is non-linear. Figures 5 and 6 show more clearly the exponential nature of the increase - the log of the intensity increases roughly linearly with gain (ignoring the effects of the upper threshold). Note that the software is doing calculation based on non-integer values above the lower threshold as, for larger portions of the data, it is clear that some pixels recorded as 10 increase as the gain is increases and others do not.

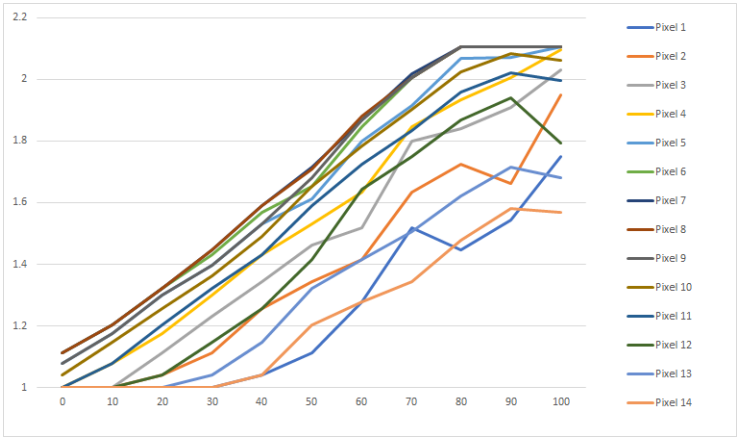

Figure 6: Log plot of intensity increase per pixel as gain increases

The second part of the experiment looks at the effect of changing the mode to one that is not of a fixed position in the tank. Previously, without any knowledge about the structure of the data, it would have been impossible to make interpretation of data that is constantly changing. Since the data portion has been identified, we examine how it changes as the sonar direction changes.

The change in mode from stop to rotate mode is reflected in position 233 and detected using the previous method.

To look at how the clockwise rotation is recorded, the behaviour of the periodic headers can be examined as it is highly likely that any angle information is contained in the portion preceding the scan data at each time step (Figure 7.)

Positions 9 and 10 in these headers increase as time progresses and the scan moves clockwise. When position 9 goes past 255, position 10 increases by 1, and this pattern continues until position 10 reaches 3 and then it reduced to 0 rather than increasing to 4. The recordings were made such that they begin close to the angle marked zero and last for two full revolutions. During the two revolutions, the above cycle occurs twice. Thus the conclusion was made that each increase of 256 on digit 9 corresponds to a clockwise rotation of 90 degrees and one full revolution corresponds to a difference of 1024. The relative position to the zero point. A quick calculation shows that $360/1080 \approx 0.35 degrees$ and Figure 7 shows an approximate agreement with the wedge size for normal speed being 2 as the above number shows increments of around 2. This also occurs for other speeds (1 for high resolution and 4 for fast). To double-check this interpretation, the angle recorded in the data was checked for one of the recordings made in stop mode. The expected two digits were 213 3 but what was actually in the file was 214 3. The error is due to rounding as a non-integer calculation of the first digit is 213.333. Note that in other places in the data file the variables that

relate to angles are stored in degrees.

```
fwd normal g=0.dat
First  5 Periodic Headers:
[1, 0, 0, 0, 32, 88, 184, 92, 241, 3, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 32, 88, 184, 92, 243, 3, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 33, 88, 184, 92, 245, 3, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 33, 88, 184, 92, 247, 3, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 33, 88, 184, 92, 249, 3, 0, 0, 144, 1, 144, 1]

Last  5 Periodic Headers:
[1, 0, 0, 0, 134, 88, 184, 92, 12, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 134, 88, 184, 92, 14, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 134, 88, 184, 92, 16, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 134, 88, 184, 92, 18, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 134, 88, 184, 92, 20, 0, 0, 0, 144, 1, 144, 1]

fwd normal g=10.dat
First  5 Periodic Headers:
[1, 0, 0, 0, 184, 88, 184, 92, 253, 3, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 184, 88, 184, 92, 255, 3, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 184, 88, 184, 92, 0, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 184, 88, 184, 92, 2, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 184, 88, 184, 92, 4, 0, 0, 0, 144, 1, 144, 1]

Last  5 Periodic Headers:
[1, 0, 0, 0, 29, 89, 184, 92, 14, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 29, 89, 184, 92, 16, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 29, 89, 184, 92, 18, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 30, 89, 184, 92, 20, 0, 0, 0, 144, 1, 144, 1]
[1, 0, 0, 0, 30, 89, 184, 92, 22, 0, 0, 0, 144, 1, 144, 1]
```

Figure 7: First and Last 5 Periodic Headers for Two Sample Scans

Checking the periodicity of the data worked with all other files except for one, so an investigation was made into that by playing to file to see why it was different. The reason was that one of the settings, the gain setting, was altered during the recording. It was discovered by looking at the data, and later verified, that every time a setting change is made there is a new block of setting data similar to the opening block of every file (see Figure 1). Each new setting is located by searching for the header sequence [0,0,0,0,6,0,0,0,1]. Looking at the periodic data recorded between setting changes, it can be seen that the time position in the periodic headers remains continuous.

Experiment 4: Resolution

In the section on interpretability of data, we mentioned the notion of "permissable values". To verify rudimentary observations on there only being a finite number of possible values for each setting, statistical counts of each unique value were made. The formula:

$$\mathtt{P} = \left\{ 10 + \left\lfloor \frac{118i}{2^b - 1} \right\rfloor \middle| 0 \leq i \leq 2^b \right\}$$

intuitively makes sense as it is how we would naturally split continuous measurements into 2^b discrete integer values within a given interval. The number of possible values determined by the above formula and the integers observed in a

frequency count of the data files matched exactly. We will only use recordings at 7 bit resolution as 8 bits are reserved for every data point anyway, so there is no saving in space or speed to record at anything less. Employing histogram distribution we could, if desired, create data/images at a lower resolution from a higher one, but not vice versa.

Final Notes for this Section

There is now sufficient information to be able to extract the necessary details from an arbitrary .dat file. Two levels of detail are possible, but the lesser detail is sufficient for most purposes (Figure 8.).

```
FILE NAME: tyre 500Hz g=100 hires.dat
FILE SETTINGS
-------------
Sonar Freq:  500 Hz
Resolution:  7 Bit
Scan Speed:  High Resolution
Scan Mode:  Scan
Range:  4 m
Percentage Gain:  100
Sector Centre:  220 deg
Sector Width:  90 deg
```

```
fast 2bit.dat
Sonar Freq:  500 Hz
Resolution:  2 Bit
Scan Speed:  Fast
Data Points per Scan:  400
Scan Mode:  Rotate
Range in m:  4
Range in cm:  400
Percentage Gain:  75
Start Angle:  345 deg
Head Offset:  0 deg
Speed of Sound:  1480 m/s

fast 3bit.dat
Sonar Freq:  500 Hz
Resolution:  3 Bit
Scan Speed:  Fast
Data Points per Scan:  400
Scan Mode:  Rotate
Range in m:  4
Range in cm:  400
Percentage Gain:  75
Start Angle:  345 deg
Head Offset:  0 deg
Speed of Sound:  1480 m/s
```

```
FILE NAME: tyre r=4 f=1000 hires.dat
INITIAL SETTINGS
----------------
Sonar Freq:  1000 Hz
Resolution:  7 Bit
Scan Speed:  High Resolution
Scan Mode:  Scan
Range:  4 m
Percentage Gain:  60
Sector Centre:  5 deg
Sector Width:  100 deg

SETTING CHANGE 1
--------------
Percentage Gain:  61

SETTING CHANGE 2
--------------
Percentage Gain:  62
```

Figure 8: Examples of Reading Details from File: (*Left*) Basic setting extraction for file without changes made during recording (*centre*) Full setting extraction from file. Note that the superfluous fields are usually constants or affect only the display when using the software (*Right*) Basic setting extraction when the user changes setting during recording. In this case, the gain was increased from an initial 60 to 85. The step increases recorded in the data and the relevant field is printed to screen.

There remains some information that we have not obtained locations for in the data, mostly because it has not been relevant for what we will need. They are:

- **Baud Rate, kbaud**: The rate at which the sonar device communicates with the software.
- **Autogain Settings, on/off**: Rather than set the gain, it can be controlled automatically. The gain itself would likely be recorded but the on/off status of the Autogain will be noted somewhere in the data.
- **Scan Direction**: The user sets this to clockwise or anticlockwise. Since

the angle at any given moment is recorded in the data, it would be superfluous to extract this information.

Finally, there are a couple of avenues that we have not investigated. Firstly, we have not looked at setting mode 2, which corresponds to Flyback Mode. Secondly, we have not looked at what happens to the data if the mode changes during recording. For both of these, we would expect the data stored to be extracted in the same way. For the former, once the scan reaches the end of one sector, it is sent immediately back to the starting angle and so the data consists only of all clockwise or anticlockwise sweeps. Our extraction algorithm in the next section should be able to cope with this without any additional considerations. For the latter, it would likely just be the same as any other setting change with the caveat that the data slices for the new mode must be extracted differently from the previous one.

4 Extracting Images from the Data

The task now remains to create images from the data files in a consistent manner. We have the following requirements:

- We should load a file and be able to extract images from it, regardless of mode.
- All images extracted from a single file should be of uniform dimensions as we would want to use this in a classifier.
- All images should have the same orientation. This is necessary since data recorded in the reverse direction will result in a mirrored image.

The eventual path to extracting images is shown in Figure 9.

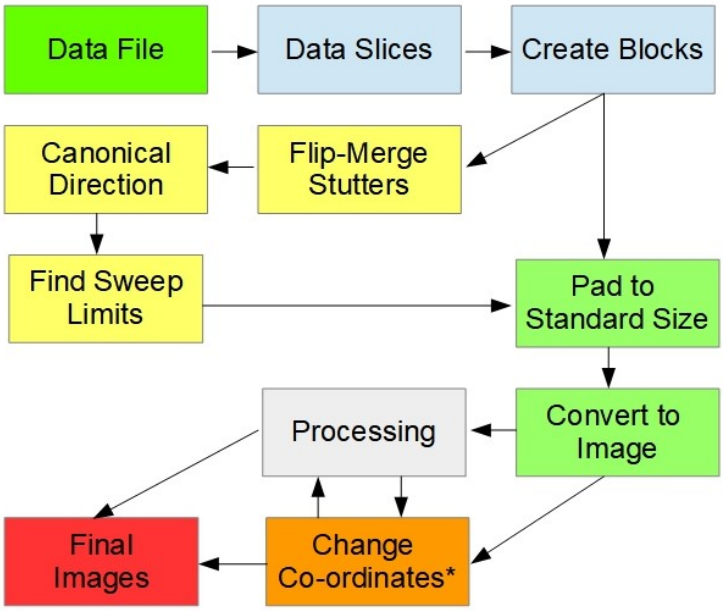

Figure 9: Process of Image Extraction

This section will explain and justify each step of the process as well as show some of the less obvious difficulties.

Data File

Our data file, as interpreted by the previous two sections, is our starting point. Incorrect interpretation of this file could lead to trouble further down the line, so there is motivation to understand it as fully as possible.

Data Slices

The data slices are the basis of any image construction and contain the bare minimum of information that we need to construct images from the data files. Each slice is a 1D array of length D+1 - the angle as recorded for that data slice followed by the D data points subject to the constraints of the setting parameters. The number of slices is entirely dependent on the length of the scan. We extract the above at the expense of losing any information about how the settings may have changed during the recording, so at this point it is best practice to try to keep the settings unchanged during recording. On the other hand, we do have some example files where there was a change in setting which nonetheless resulted in successfully extracted images. The experiments we carried out on some sample objects at a variety of frequencies, gains, ranges

and speeds did not include Super Fast speed, so D=400 for all of the data slices for which we extracted images.

Create Blocks

Each image must consist of some subset of the recorded data. The three different modes demand three different treatments, as shown in Figure 10. Note that, for each data file, we would expect to retrieve a number of different images. For our test experiments, these were approximately 10-32 for each file, with the slower scans producing 10-12 and the faster ones 28-32.

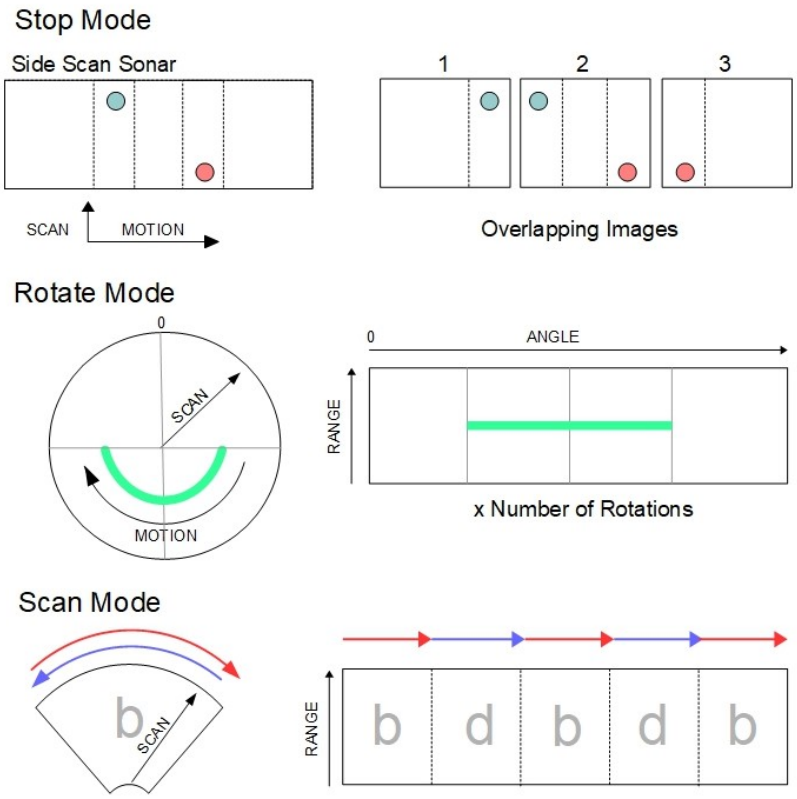

Figure 10: Data Block Extraction for Each of the Three Modes

Stop mode is used for side-scan sonar where we move the device laterally and record the backscatter from each pulse. Extracting images from this data is relatively simple and simply require splitting the array representing the data slices into equal portion, perhaps with some specified level of overlap similar to the sliding window approach used in object detection.

For rotate mode, we separate the array after each full rotation, measuring from the angle at the beginning of the recording.

Lastly, for scan mode, it would be natural to separate images based on clockwise/anticlockwise sweeps of the area of interest. Designating a change of direction as the separation point between image extractions seems a natural way to approach this. As will be shown in the next section, however, this approach needs some processing.

For the last two modes, it can often be the case that two consecutive data slices are assigned the same angle. The data therefore goes through a preprocessing stage where we merge the two (or more) consecutive data slices into one by averaging the intensities over equivalent data points in each slice. Not only does this give a more smooth movement, it also can be used to remove some level of noise. The results of examining the proportion of data slices remaining after the merging process give the following mean, standard deviation and skewness obtained from 17 Hi Res, 14 Normal and 17 Fast data files.

$$\begin{aligned} \text{Hi Res: } \mu = 0.638 \quad \sigma = 0.0008 \quad \text{skew} \approx 0 \\ \text{Normal: } \mu = 0.989 \quad \sigma = 0.0019 \quad \text{skew} = 0.662 \\ \text{Fast: } \mu = 0.975 \quad \sigma = 0.0205 \quad skew = -3.49 \end{aligned}$$

While there is some variation in the contraction factor for the Fast and Normal speed scans, the Hi-Res scans have an almost constant contraction factor. The reason for this is not known. It could be that the angles we are merging correspond to a finer gradation than is recorded, but since the repeated consecutive angles have no obvious pattern it is not likely that this is due to rounding of the recorded angle. Furthermore, a difference in angle of 1 between each consecutive data slice corresponds to measurements of $1/3^o$. Even at this level, the precision is doubtful, so it seems unlikely that the device would be more accurate. If we were recording at a finer gradation, it would be simple to record the angles on a scale of 0-2047 or similar so each discrete step corresponded to a unique integer without taking up any extra memory (for bytes allows us to have a scale of up to 0-65536).

Finally, it should be noted that the angles are stored in a base 1024 number system (much as we have a base 360 system for degrees) so it is not sufficient to determine direction solely on whether the angles for consecutive data slices are increasing or decreasing. If the sweep crosses the angle designated 0 in the anticlockwise direction it can go from 2^o to 358^o. This will be interpreted as an increase of 356^o in the clockwise direction rather than a decrease of 4^o in the anticlockwise direction. A general way of dealing with such exceptional cases is implemented.

Flip-Merge

Ideally, the results of the block extractor should be of roughly the same size and easily obtained by identifying the points where we have a change in direction In reality, recordings experience a "stutter" where there is a short change in

direction counter to the expected one lasting two or three data slices (Figure 11). This results in the extraction of very short data blocks that break up what should be one continuous sweep.

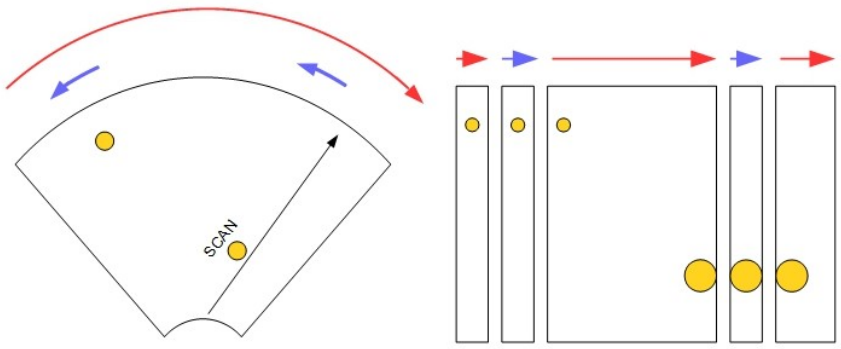

Figure 11: Erroneous Block Extraction due to Intermittent Counter-clockwise Portions

Note that this also results in the same area being scanned in triplicate. The correction for this is to reverse the direction of all the erroneous segments and then merge all of the data slices that are triplicated using the same method as above. This involves joining together all of the flipped sections, so we called the algorithm Flip-Merge. Since the erroneous segments are small, we can identify them by size and infer the correct direction from that of the adjacent blocks. The threshold we used was a block size of less than 10 data slices. There are some subtleties involved in degenerate cases where the erroneous segment occurs at the start of a sequence of extracted data blocks or when we have two adjacent erroneous segments. Some care also has to be taken with determining sorting in a base 1024 system.

Figure 11 also demonstrates the phenomenon we face with how the sonar records the data and what the objects look like in real life. On the left hand side, we can see a scene containing two yellow circles. The data is recorded in polar coordinates (angle, distance from centre). The right hand side shows the shape of the received data. Objects that are closer take up more of the field of view and so appear larger in the data, while ones that are distant appear smaller. This is how our own eyes operate too. In addition, we have some distortion of shape due to the change in coordinate system.

The contraction factor would be calculated using only a very small number of data slices, so is negligible compared to the previous contraction - around two orders of magnitude smaller.

Canonical Direction

For the purposes of consistence, it makes sense to have all of the data blocks representing the same image. Our default direction is clockwise, so we flip all the anticlockwise segments. The reason is shown in Figure 10 where we can see the image is reversed for anticlockwise passes of the letter 'b'.

Find Sweep Limits

It is possible that there is some natural variation on the most clockwise and anticlockwise points in the data. This could because of some change in the centre or width of the scan made during the recording. The exact start of the recorded data is also dependent on when the user clicks on record and instructs which file to save the incoming data to. As this is not fixed, the recording could start anywhere and so the beginning will only show part of a full scan. Extracting this data make sure we know what the bounds of our scanning domain is.

Pad to Standard Size

There are two levels of padding applied here. Firstly, we want all images extracted from the same file to be a standard size. For this we simply pad out the each block so it matches the sweep limits found in the previous step. We chose to pad it with empty space, but one could also pad it with data copied from the nearest edge. Empty space is suitable because much of our background data is just that. For other applications, the other method might be better as it would avoid the appearance of sharp gradients.

For the second type of padding, we must look at the data for some of the faster scans (Figure 12.)

```
BLOCK 1
[903, 21, 48, 127, 112, 10, 10, 10, 10, 10, 10, 42, 124, 10, 10, 10, 10, 10, 91, 10, 111, 125, 10, 10, 100, 10, 10, 10, 10]
[905, 10, 10, 100, 10, 10, 10, 10, 10, 100, 10, 21, 10, 10, 64, 10, 10, 10, 10, 10, 10, 10, 10, 29, 10, 10, 10, 10, 54, 10]
[907, 10, 10, 80, 10, 10, 10, 93, 10, 64, 10, 40, 10, 10, 10, 10, 10, 10, 37, 10, 10, 10, 31, 103, 39, 10, 10, 10, 10, 10]
[909, 10, 75, 10, 78, 64, 10, 10, 10, 10, 10, 10, 37, 10, 10, 10, 41, 10, 10, 27, 120, 25, 10, 10, 10, 10, 10, 10, 10]
[911, 10, 10, 42, 26, 10, 10, 10, 10, 54, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 92, 10, 10, 10, 10, 10, 10, 94, 10, 23]
[913, 10, 10, 10, 10, 10, 10, 10, 10, 43, 10, 10, 10, 10, 10, 73, 118, 10, 39, 10, 10, 49, 65, 11, 100, 10, 44, 10, 10]
[915, 10, 29, 10, 10, 10, 10, 10, 15, 10, 33, 10, 10, 10, 10, 10, 10, 10, 82, 10, 10, 10, 10, 95, 108, 64, 32, 10, 10, 10]
[917, 10, 10, 10, 10, 10, 10, 10, 11, 10, 10, 10, 122, 60, 10, 10, 10, 10, 67, 96, 40, 10, 55, 31, 86, 10, 10, 39, 10, 10]
[919, 10, 54, 115, 10, 10, 48, 10, 10, 93, 10, 94, 10, 10, 10, 10, 105, 10, 10, 10, 10, 10, 10, 101, 69, 10, 10, 10, 10]
[921, 10, 10, 10, 10, 10, 10, 10, 10, 71, 10, 10, 10, 14, 10, 17, 10, 10, 10, 55, 10, 71, 10, 10, 10, 10, 10, 48, 10, 10]
[923, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 64, 100, 99, 10, 45, 93, 10, 32, 10]
[925, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 55, 10, 10, 10, 10, 10, 43, 10, 10, 10, 11, 10, 10, 10, 10, 10, 10]
[927, 127, 10, 10, 10, 10, 42, 10, 10, 10, 41, 17, 25, 10, 10, 81, 123, 10, 10, 10, 126, 10, 10, 10, 10, 10, 10, 10, 10]
[929, 81, 10, 91, 10, 10, 10, 10, 10, 10, 10, 37, 10, 10, 10, 10, 10, 10, 10, 10, 37, 10, 120, 10, 10, 10, 10, 10, 10, 106]
[931, 10, 108, 69, 10, 10, 10, 10, 10, 10, 10, 10, 59, 10, 10, 10, 10, 53, 10, 58, 86, 10, 10, 110, 10, 10, 10, 11, 10]
[933, 14, 10, 10, 61, 10, 10, 25, 10, 10, 10, 10, 10, 10, 115, 10, 10, 10, 10, 10, 98, 10, 10, 10, 10, 10, 10, 125, 27]
```

Figure 12: Discretely Sampled Data: The test case shown shows the angular position in the first entry of each row. Data corresponding to measurements sampled every angular increment of 2 corresponds to Normal speed. Fast and Super Fast sample at increments of 4 and 8 respectively.

The data for the angles not represented must be generated if our extracted images are to be of the same size for the same sectors but using a different speed setting - The number of data slices obtained from a scan at normal speed

is only half the number of those scanned at high resolution but is double those obtained using fast speed. The missing data is generated by interpolation. Different methods can be chosen.

- **Nearest Neighbour**: We simply copy all data from the adjacent data slice.
- **Linear Interpolation**: We fill in each data point according to the formula:

$$D_t(x) = (t - i) \cdot \frac{(D_f(x) - D_i(x))}{f - i}$$

  Where i,t and f are the indices of the initial, target and final data slices within the data block. The initial and final are data slices for which we have data and the target slices are the ones we must interpolate. x is the position of the data point within the data slice and $D_k(x)$ is the value of a data point for slice k at point x.

- **Cubic Interpolation**: We use the two nearest values and their gradients. The formula is given by:

$$\begin{aligned} D_s(x) = {} & h_{00}(s)D_i(x) + h_{10}(s)(f - i)m_i(x) \\ & + h_{01}(s)D_f(x) + h_{11}(s)(f - i)m_f(x) \end{aligned}$$

  Where $D_s(x)$ is the value of data point x at position s. $s = (t-i)/(f-i)$ with t,i and f as above. The functions in s are polynomials as follow:

$$\begin{aligned} h_0 0(s) &= (1 + 2s)(1 - s)^2 \\ h_1 0(s) &= s(1 - s)^2 \\ h_0 1(s) &= s^2(3 - 2s \\ h_1 1(s) &= s^2(s - 1) \end{aligned}$$

  $m_i(x)$ and $m_f(x)$ are the gradients at the initial and final known data slices at data point x between which we wish to interpolate. These gradients are calculated using the values $D_{i-1}(x)$ and $D_{f+1}(x)$ respectively to give an estimate using the standard gradient formula.

For Super Fast speed, there are only 200 data points in each data slice, so the data is first interpolated between data slices and then interpolated in the scan direction to create 400 data points per slice. There are other methods of interpolating this data too, but it should be borne in mind that when we are using any of these methods, we are making a hypothesis about how the unknown data might behave. More complex interpolation will make the image look less pixelated/smoother, but it is still fabricated data that is sensitive to noise. The No Free Lunch Theorem says that there will always be some circumstances where the algorithm does not give optimal performance, but we can still conclude that some methods will be suitably robust.

Convert to Image

All of the data we have worked with so far has been purely numbers. We do not, in fact, strictly need to work with anything else. Machine learning algorithms need only arrays of numbers. Nonetheless, it is helpful to view an image to check what is going on if only for sanity's sake. For the immediate purpose, we want to see that our interpreted data is actually similar to the replay file as viewed in the Sonavision software. There is a one-to-one relationship between the array and the image, but converting to and from the image means extra computation so we want to do this as little as possible, and not at all if it can be avoided. Figure 13 shows the result of some of successful image extractions.

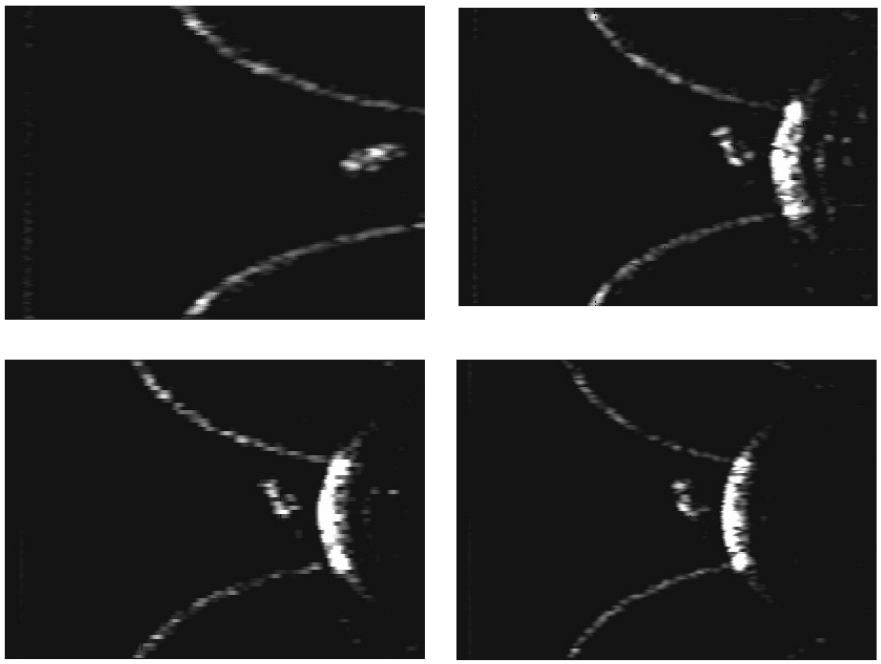

Figure 13: Extracted Images from Data Blocks: Clockwise from top left. Bag with settings range=3m freq=600Hz at normal speed. Tyre with settings range=3m freq=600Hz at normal speed. Tyre with settings range=4m freq=800Hz at fast speed. Tyre with settings range=5m freq=1000Hz at hi res speed.

Note that width of each image is the same as each data slice has length 400. The heights are different as the sectors have different widths.

Processing

For machine learning purposes, we may need to preprocess the data. Since

the images and the data blocks are equivalent, it does not matter which the preprocessing is applied to. The variety and efficacy of preprocessing techniques requires further investigation, but they may include:

- Resizing to common dimensions
- Image Fusion (where we combine more than one image to correct for noise or movement)
- Image thresholding
- Noise reduction techniques
- Histogram equalisation

Change Coordinates

Figure 13 shows the distortion caused by the polar nature of the data blocks when translated directly to image form. The objects themselves are positioned so that they do not change shape significantly, but the straight walls of the tank appear as curves. To get a truer representation of the original scene being scanned, we would need to convert this to Cartesian coordinates. This does, however, require some discussion.

- **Information loss**: While performing a transformation on the data may help improve on the distortion and make the shapes more true to life, there is a trade-off. The data points at close range contain the highest information density per area, but in performing the transformation to Cartesian coordinates, it is precisely this information that we lose. At the other extreme, the objects that are distant have low information density, and the transformation must map them to a larger area than in the original, resulting in blocky images. This can be ameliorated by using interpolation techniques but, as we noted above, this generates a hypothesis about the data and forces us to create data that may not be valid. For a machine learning algorithm using these images as a dataset, we would have a large proportion of the area of the image composed of data that has been interpolated while at the same time shrinking the area of the image containing real measurements. For this reason, it would seem to make sense that we do not perform such a transformation as it dilutes the ground truth.
- **Source and Target Training Sets**: If we use any form of pre-trained network to classify these images, we are making an assumption about the similarity between these sets. Large pre-trained networks learn a very huge number of feature detectors. The question is whether or not these will be suitable for a new task. On the one hand, the change in coordinates could mean that the objects are not easily recognisable by a human in locations where the distortion is most extreme, but on the other hand the features in the distorted objects may not be so extreme as to be not well described by the feature extractor generated by these networks. It is a question of whether the latent space is sufficiently expressive to perform classification.

Given the machinery used here, we would expect that this is not only the case but that it is over-engineered for this purpose.

- **Preprocessing**: Before we reach the stage where we have images/arrays that are suitable as inputs to a machine learning classifier, we may need to do preprocessing of some kind. Note that, in the case we are using the data *after* transformation, there is a question of where this take place in order to preserve the maximum amount of information. ie. does applying preprocessing before transformation have any benefit compared to applying it after? Do we end up with the same image?

Final Images

We are now finally ready to do something with the extracted images.