Radar memory compression model

1. Summary

Compression/Decompression Engine is a new feature of TI mmWave radar devices, starting with the IWR6843 ES2.0 device. This engine is designed to provide memory compression and decompression to increase the maximum size of radar cube that is saved in L3 memory. For more information about memory compression, see the Memory Compression and Decompression Engine for TI mmWave radar application notes.

All of the compression methods that are available can be lossy. When the 'desired compression ratio' cannot be achieved without loss, the engine drop the minimum number of LSB bits from every sample in a block so that the compression ratio is achieved. Consequently, weaker objects may not get detected because of the increased noise floor due to quantization.

In this delivery, we provide a MATLAB script to enable evaluation of this engine. As shown in **Figure 1**, this MATLAB script takes input data from either captured data or simulated data. After range FFT, the range FFT output is passed through the bit-matching c-model of the compression engine before saved to memory. Later, Doppler FFT is applied to the decompressed range FFT Output. At the end of the script, the Doppler FFT output with and without compression is compared to demonstrate the impact of compression engine.

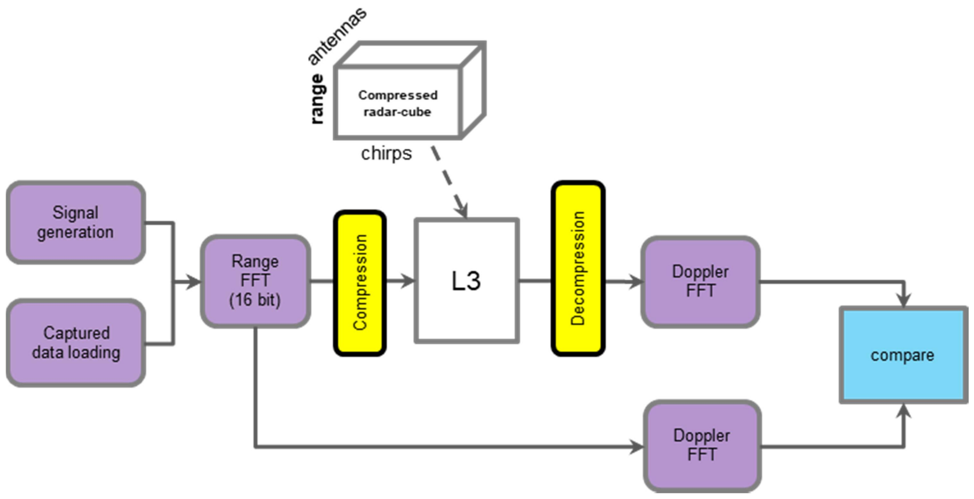

Figure 1 Block diagram of radar memory compression model

Note: (1) To match with our range FFT module in the mmWave SDK, the range FFT output is rounded to 16 bits in this MATLAB script. (2) For the data processing, only one frame of data is used. (3) The bit-matching c-model of the compression/decompression engine is wrapped as mex function and provided as a black box only.

2. Running the script

To start with, the user can run the main script under .\matlab\
>>*test_main.m*

Note that all the programmable parameters are located in *setSystemParam.m*. By default, the script uses a pre-captured data, provided with this package.

First, the range-Doppler heat-map is generated to provide an overall look of the captured data (or simulated data). From this range-Doppler heat-map, users can find the range bin and Doppler Bin of the peak location. Note that the data set (included in the package) is captured in an Anechoic Chamber with a strong corner reflector, which results a distinct peak in the range-Doppler heat-map plot at range Bin of 38 and zero Doppler bin as shown in **Figure 2**.

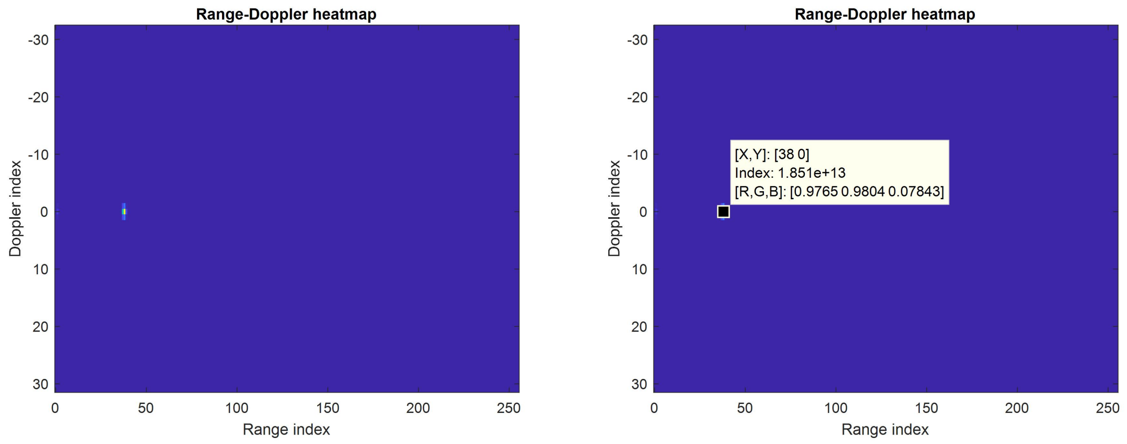

Figure 2 Found the range and Doppler peak information from range-Doppler heat-map

Using the known (rangeBin, DopplerBin) coordinate location, users can program the following parameter in the parameter file so that the script can extract the Doppler output at this range to show the compression effect.

```
param.resultDemo.rangeBin = 38;
```

Users can easily check the performance with different compression ratios by changing the parameter below,

```
param.cmpOptions.cmpRatio = 0.5;
```

Figure 3 shows the results with a 50% compression ratio; **Figure 4** shows the results with a 62.5% compression ratio; **Figure 5** shows the results with a 37.5% compression ratio; A 37.5% compression ratio means that the compressed radar cube is only 37.5% of original radar cube size. A smaller compression ratio results in a higher noise floor.

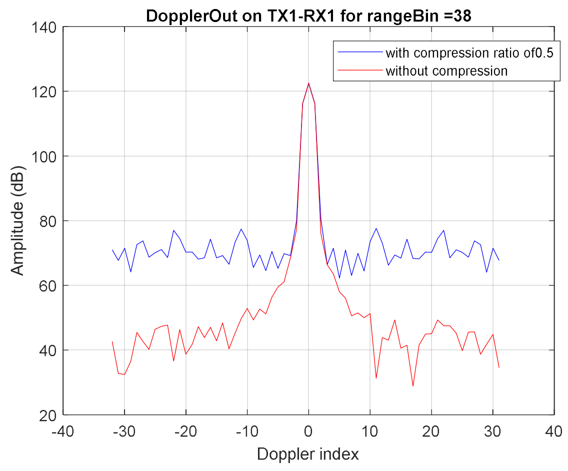

Figure 3 Performance with compression ratio of 0.5

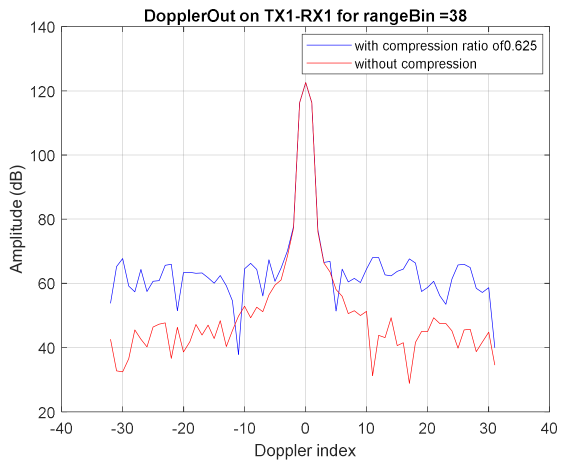

Figure 4 Performance with compression ratio of 0.625

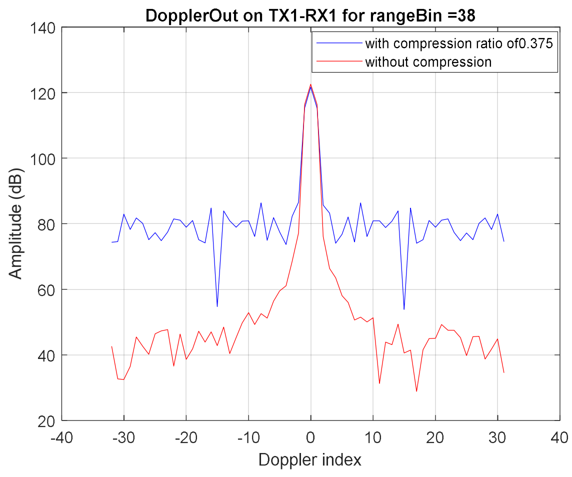

Figure 5 Performance with compression ratio of 0.375

3. Programmable Parameters

All the programmable parameters are located in *setSystemParam.m*. The script provides two options for ADC raw data input:

- param.capturedDataFlag = 1: ADC data is loaded from a pre-captured data file;
- param.capturedDataFlag = 0: ADC data is generated through simulation;

Note that this simulated ADC data does not include any analog impairment such as thermal noise or ADC quantization noise.

3.1 Parameter related to the captured data

It is important to match the chirp/frame configuration parameters between data capture procedure and data loading procedure. These parameters include

```
param.capturedData.fileName = '..\data\adc_data_Raw_0.bin';
param.capturedData.numAdcSamples  = 256;
param.capturedData.numChirps = 64;
param.capturedData.numVirtualAnt  = 12; % total virtual antenna in
TDM-MIMO case

param.capturedData.targetLoc.range_bin =  38;
param.capturedData.targetLoc.doppler_bin =  0;
```

Note that the information on targetLoc is only used for results comparison plotting. Users need to determine these values from the range-Doppler heat-map as shown in **Figure 2**.

3.2 Parameter related to the simulated data

In simulated data, we need to define the basic chirp configuration parameter as well as the information for target location. Currently, only one target is simulated.

```
param.simulatedData.numAdcSamples  = 256;
param.simulatedData.numChirps = 64;
param.simulatedData.numVirtualAnt  = 8;  % assume linear antenna
with lambda/2 spacing
param.simulatedData.targetLoc.range_bin=50;
param.simulatedData.targetLoc.doppler_bin=10;
param.simulatedData.targetLoc.angle_bin=5;
```

For the simulated data,

- The users can define target location and speed, but only one target is defined and included.
- The antennas are assumed to be linear with lambda/2 spacing
- There is no thermal noise or other analog impairment being simulated.
- Only one frame of data is generated

3.3 Parameter related to the compression/decompression engine

The key parameter is compression ratio. Parameter 'dimOrder' tells the order of the dimensions for compression. Users can try to compress on antenna, chirp or range domain. Parameter 'samplePerBlock' tells how many samples to compress at one time. This parameter has to be a power of 2 and a larger number will improve performance.

```
      param.cmpOptions.method = 'EGE'; %'EGE' is the only option;
      param.cmpOptions.cmpRatio = 0.5;
      param.cmpOptions.dimOrder = 'chirp_antenna_range';
      'antenna_range_chirp', 'range_antenna_chirp', 'chirp_antenna_range'
      param.cmpOptions.samplesPerBlock = 8; % has to be power of 2, and
      larger number will improve performance.
```

There is a memory limitation in compression/decompression engine, which is not captured in this MATLAB model. For different 'dimOrder', the required memory is listed in the table below:

dimOrder	Memory needed during compression	Memory needed during decompression
range_antenna_chirp	$N_{Rx} \times N_{range} \times 4$	$N_{Rx} \times N_{chirp} \times \mathrm{samplesPerBlock} \times 4$
chirp_antenna_range	$N_{Rx} \times N_{range} \times \mathrm{samplesPerBlock} \times 4$	$N_{Rx} \times N_{chirps} \times 4$
antenna_range_chirp	$N_{Rx} \times N_{range} \times 4$	$max(N_{Rx}, samplesPerBlock) \times N_{chirp} \times 4$

The total hardware memory is 16KB. The memory usage needs to be within this 16Kbyte for both compression and decompression stage.

4. Data capture tool

The data capture system includes

1) DCA1000 data capture card
2) mmwaveICBoost card
3) mmwave IWR6843 ES2.0 board

The newest released radar studio can be downloaded at http://software-dl.ti.com/ra-processors/esd/MMWAVE-STUDIO/latest/index_FDS.html

A training video can be found at https://training.ti.com/dca1000-training-video to help understand the capture procedure.

After the capture, make sure to update following parameter to match your captured data:

```
      param.capturedData.fileName
      param.capturedData.numAdcSamples
      param.capturedData.numChirps
      param.capturedData.numVirtualAnt
```

Number of chirps per frame is also called num_loop in frame configuration. The number of virtual antenna is the total of virtual antenna in the TDM-MIMO mode.