University Defence Research Collaboration (UDRC) Signal Processing in a Networked Battlespace

Parallel Processing of the Fast Decimation-in-image Back-projection Algorithm for SAR

Shaun I. Kelly, Mike E. Davies and John Thompson, University of Edinburgh

Abstract: Fast back-projection algorithms provide substantial speedup when compared with the standard back-projection algorithm. However in many potential applications of synthetic aperture radar, further speedup is still required in order to make the application operationally feasible. Multi-core central processing units and graphic processing units are considered to speed up a very recently proposed fast back-projection algorithm (the fast decimation-in-image back-projection algorithm).

Standard Back-projection Algorithm

The back-projection (BP) algorithm is a very accurate and versatile image formation algorithm for synthetic aperture radar (SAR). Therefore, the BP algorithm can be used when image quality is of critical importance.

The benefits of the BP algorithm:

- Compensation for wavefront curvature.
- Compensation for antenna beam patterns.
- Compensation for non-flat scene terrain.

In many applications of SAR, the BP algorithm is prohibitively slow. These include:

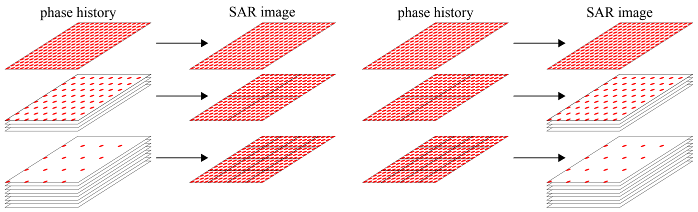
- VideoSAR
- Large scene SAR.
- 3D SAR.
- Iterative image formation SAR.

Fast Back-projection Algorithms

In the last two decades there has been many fast BP algorithms proposed. Fast BP algorithms achieve the same order of complexity as far-field algorithms, $\mathcal{O}(N^2 \log(N))$ operations, without making the far-field approximation. These algorithms include the recently proposed **fast decimation-in-image (DII) BP algorithm (Kelly, 2014)**.

Two components of fast algorithms:

1. Sub-image standard BP.
2. Decimation/upsampling.



decimation in phase history　　decimation in image

We wish to exploit parallelism in these two components to further reduce the image formation times of Fast BP algorithms.

Parallel Processing

Multi-core Central Processors Units

Pros:

- Very little overhead because no memory transfers are required.
- Close to linear speed up with the number of cores.

Cons:

- Currently their are only a modest number of cores in a multi-core CPU (typically less than 10).

Multi-core Graphics Processors Units

Pros:

- Large number of cores (up to multiple thousands).

Cons:

- Large overhead due to memory transfers to and from the GPU.
- Single precision floating point currently has more support than double precision.

Implementation

Exploiting Parallelism

Sub-image standard BP

The standard BP algorithm is an *embarrassingly parallel* problem, therefore, each pixel was computed as a parallel thread.

Upsampling

Each stage of the image upsampling is dependent on the previous stage so no multi-stage parallelism was possible. However, we were able to use each pixel in the single stage upsampling as a parallel thread.

Implementation Details

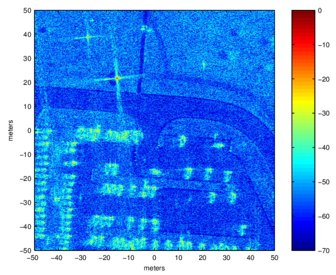
- The DII BP algorithm was implemented using the C programming language.
- The Open Multi-Processing (OpenMP) API was used to utilise a multi-core CPU.
- The Compute Unified Device Architecture (CUDA) API (version 6.0) was used to utilise a Nvidia GPU.
- Range FFT in standard BP was implemented using non-uniform FFT with 24 sample interpolation kernel as suggested in (Greengard, 2004).
- Compiled using GCC (version 4.6.3) with maximum compiler optimisation ("O3").

Results

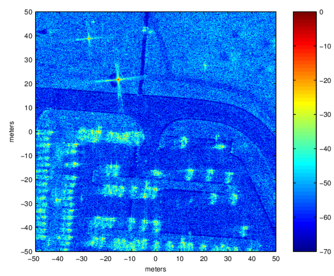
The results were generated on the following hardware:

- Intel Core i3-2100 CPU (two processors)
- Nvidia GTX 550 Ti GPU (192 single and 16 double precision cores)

Single versus Double Precision Arithmetic



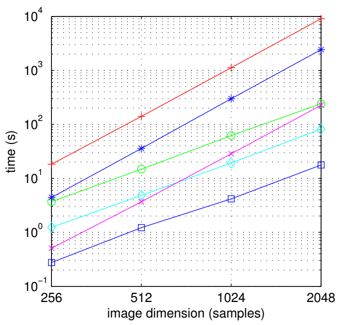
Single Precision



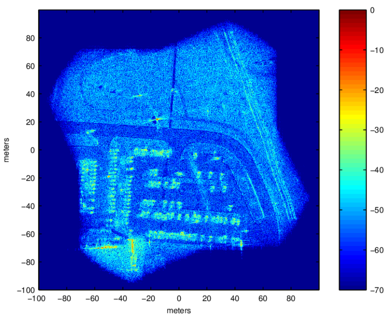
Double Precision

Single precision arithmetic has visible sidelobes from the bright targets.

Computational Performance



'+' BP algorithm (single-core CPU)
'∘' Fast BP algorithm (single-core CPU)
'∗' BP algorithm (multi-core CPU)
'⋄' Fast BP algorithm (multi-core CPU)
'×' BP algorithm (GPU)
'□' Fast BP algorithm (GPU)



AFRL Gotcha dataset
first pass and HH polarization
$1 - 25^\circ$ (2931 aperture samples)
6000×6000 pixels
antenna beamwidths: 1.15° azimuth and 0.57° polar
Image formed in 143.45 seconds

Parallel processors can be used in addition to Fast BP algorithms to significantly speed up the standard BP algorithm.

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) and Dstl.