GPU Implementation and Parallel Optimization for Electromagnetic Imaging

Mostafa Awwad (Undergraduate), Yuanwei Jin (Faculty Mentor), Enyue Lu (Faculty Mentor)
Dept. of Math. & Computer Science, Salisbury University, Salisbury, USA

Abstract

As the developers of the time reversal adaptive interference canceler (TRAIC) time reversal beamformer (TRBF), created a new algorithm to detect and locate targets in rich scattering environments. It utilizes time reversal in two stages: 1) Anti-focusing: TRAIC time reverses and then reshapes the clutter backscatter to mitigate the clutter response. 2) Focusing: TRBF time reverses the residual backscatter to focus the radar image on the target. Laboratory experiments with electromagnetic radar data in a highly cluttered environment confirm the superiority of TRAIC-TRBF over conventional direct subtraction (DS) beamform imaging. Searching for the optimum speed to implement this algorithm on Matlab, we implemented the algorithm utilizing a multi core and a GPU within the Matlab environment. Then we created kernel function in C++ and called them from Matlab code. While comparing the results we obtained interesting conclusions.

Objectives

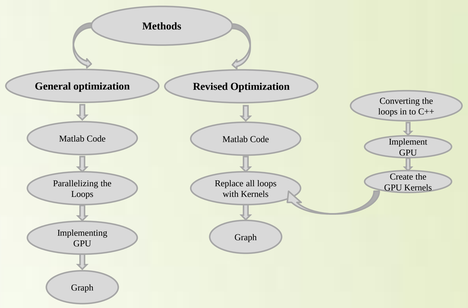
❖Finding an optimum procedure for speeding up the simulation time for sample case (Time Reversal Channel Measurement Processing and Imaging).

❖Get the most optimum speed from the Matlab code by using the following methods:

- Using parallel processing
- Using GPU with Matlab environment.

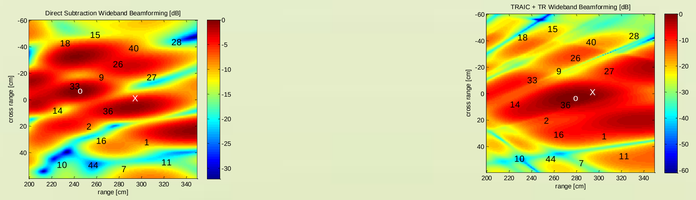
Methods

❖ General Optimization Procedure
❖ Revised Optimization Procedure

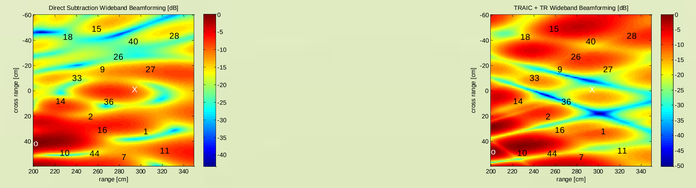


Results

- The MultiCore (4 cores) calculations is faster than the single CPU as expected and also faster than GPU for this example.
- With more than 4 cores, the speed up factor would be higher
- With the Hankel function approximation, the multicore processing is the fastest
- The time needed for the transfer of data between the system memory and the GPU memory is the determining factor for GPU speed in this simple test case.
- For much more computationally intensive cases, for example with more frequency points, it is expected that the GPU time to be less than the Multi-Core time.
- Because the results based on the approximation of Hankel function are not in match with the original results, It is recommended to use the Multi-Core or the GPU without Hankel function approximation. This is showing a minimum speed up factor of 2.



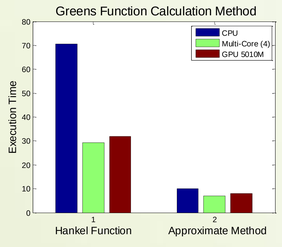
Results using Besselh



Results using Hankel Approximation In the Paper

Results using Hankel Approximation in Balanis Book

Processor Type	Time using Hankel function (Besselh) [s]	Time using Hankel Approximation instead of Besselh [s]
CPU - Single	70.501	9.964
CPU – MultiCore (4)	29.31	6.89
GPU – M5010 (348)	31.86	7.95



Conclusions

It is clear that the results based on the Hankel function approximation are not in perfect match with those based on using the Hankel function itself. Further work needs to be done to create a routine for Bessel h in C++, then continue the rest of the loops in C++, then call those functions in the Matlab code after parallelizing them.

Future Work

1. Write the Bessel h function routine in C++.
2. Convert the rest of the Matlab code loops to C++.
3. Create the call function in Matlab for the C++.
4. Parallelize the C++ code on GPU.
5. Call the functions from within Matlab.

References

José M. F. Moura, and Yuanwei Jin, “Time Reversal Imaging by Adaptive Interference Canceling,” IEEE Trans. Antennas and Propagation, vol. 56, no. 1, January 2008.

Constantine A. Balanis, *Advanced Engineering Electromagnetics*, John Wiley & Sons, Inc., Appendix IV, 1989.