Washington University in St. Louis

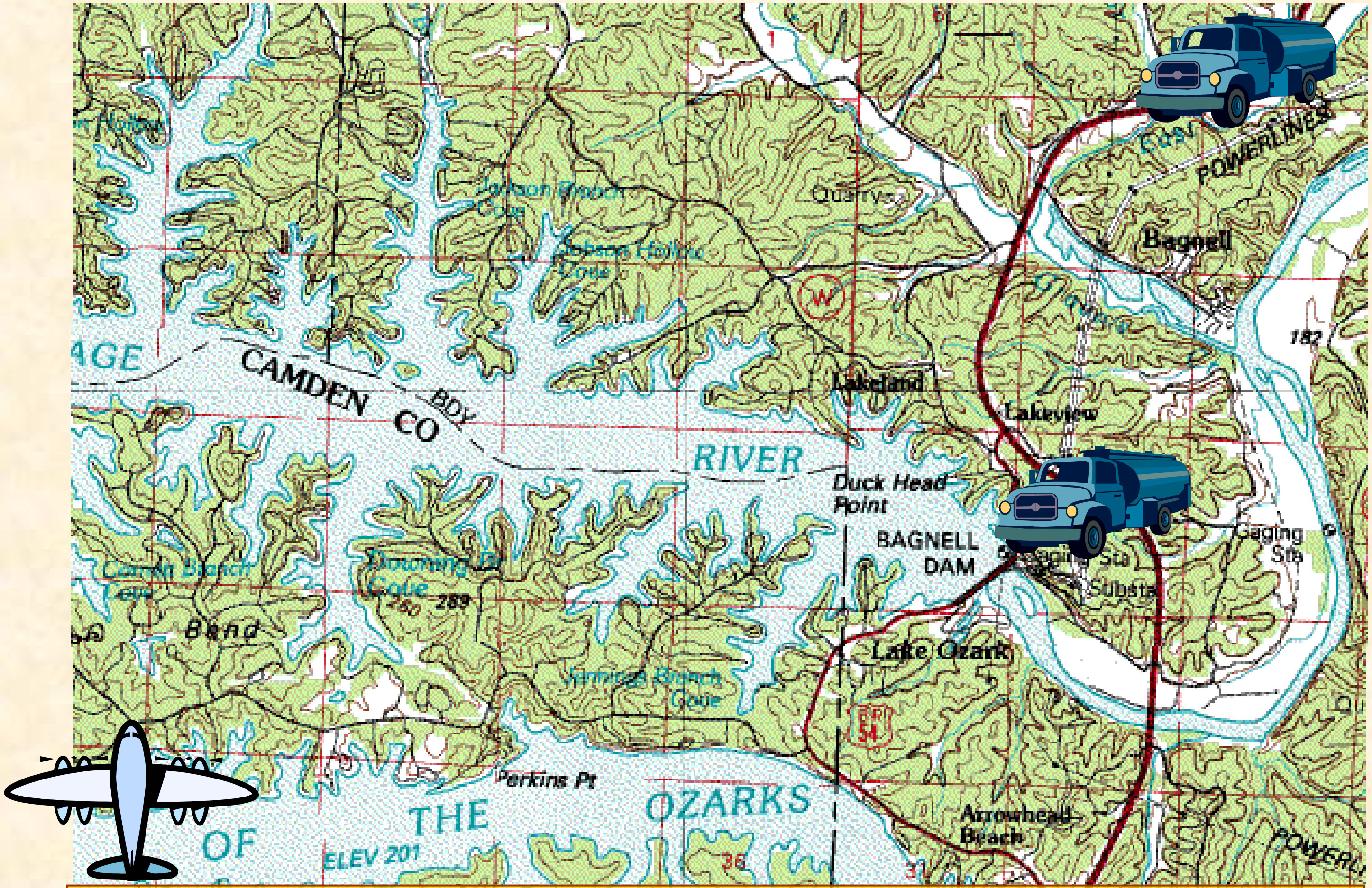
Parallel Matlab Computation for STAP Clutter Scattering Function Estimation and Moving Target Estimation

Roger Chamberlain, Daniel R. Fuhrmann,
John Maschmeyer, and Lisandro Boggio

School of Engineering and Applied Science
Washington University in St. Louis

This work was supported by the U.S. Defense Advanced Research Projects Agency through a contract with the U.S. Air Force Research Laboratory, No. F30602-03-2-0043.

Context



Problem: Detect ground moving targets in the presence of ground clutter

Context

- Wide-area surveillance airborne radar
- Arbitrary flight path
- Multiple sensors and Doppler pulses
- Space-time adaptive processing (STAP)
  - Better knowledge of the clutter covariance matrix gives better detection performance

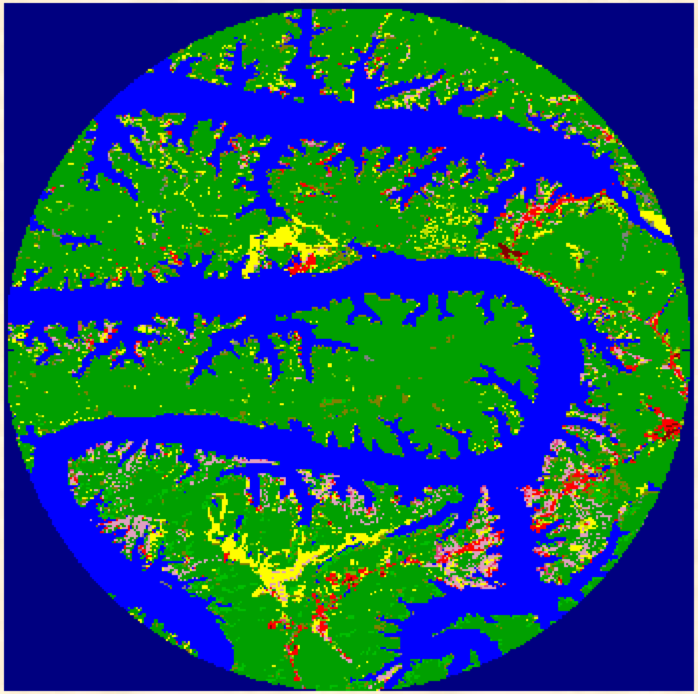
Objective: Estimate the clutter covariance matrix and detect moving targets

Approach

- Ground subdivided into pixels or ground patches
- Known range and angle of each patch with respect to airborne platform
- Known illumination pattern
- Received data: sum of returns from targets and all patches on the ground
- Prior knowledge is available:
  - Digital Terrain Elevation Maps
  - Land use information

Terrain Simulation

- **Region of Interest**
  - Lake of the Ozarks
  - 15 km diameter
  - 197,316 pixels
  - 30m resolution



Datasets

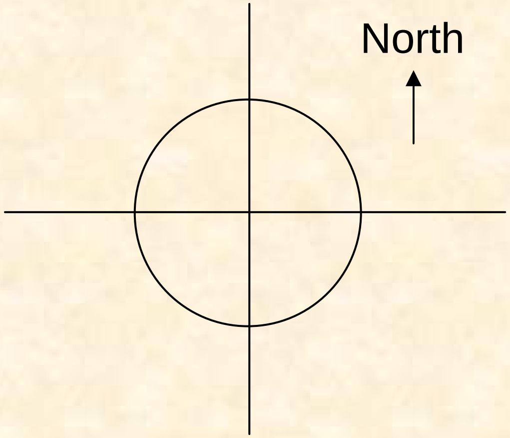
- Obtained from USGS Seamless Data Server
  - 30m resolution
- Digital Elevation Model
  - Used for modeling geometry
- Land Use
  - Scattering function based on 21 classes of land cover
    - 9 primary classes
      - Water, Developed, Barren, Forested Upland, Shrubland, Non-Natural Woody, Herbaceous Upland Natural/Semi-natural Vegetation, Herbaceous Planted/Cultivated, Wetlands
    - Each class contains one or more categories, e.g.
      - Open Water, High-Intensity Residential, Deciduous Forest, Row Crops
  - Scattering function chosen arbitrarily for simulation

Coordinate Systems

- Datasets referenced in spherical coordinates
  - Latitude, Longitude, Elevation
- Convert to Cartesian Coordinates
  - Simpler to use over small region
  - Computations can be made independent of Earth model

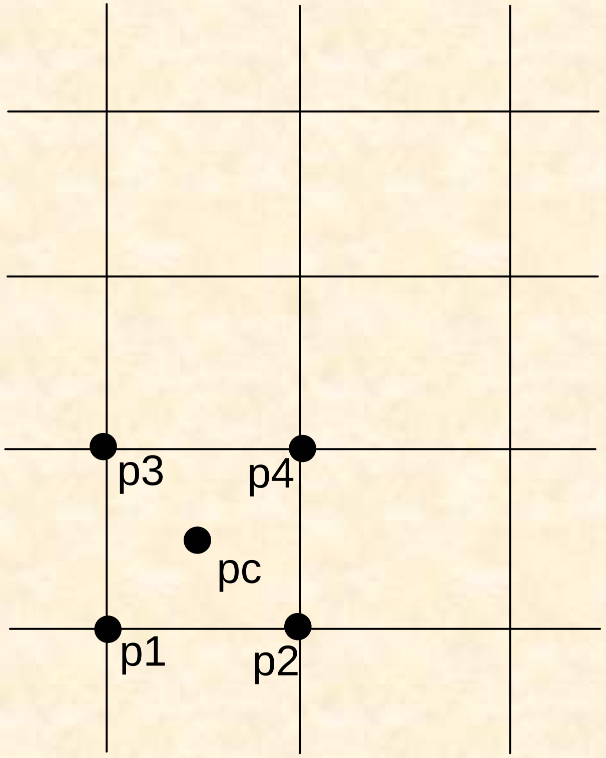
Coordinate Conversion

- First Stage
  - Origin at Earth's center
  - Use Geodetic Reference System 1980 (GRS80)
- Second Stage
  - Move origin to center of region of interest
  - Elevation along Z-axis
  - North along positive Y-axis

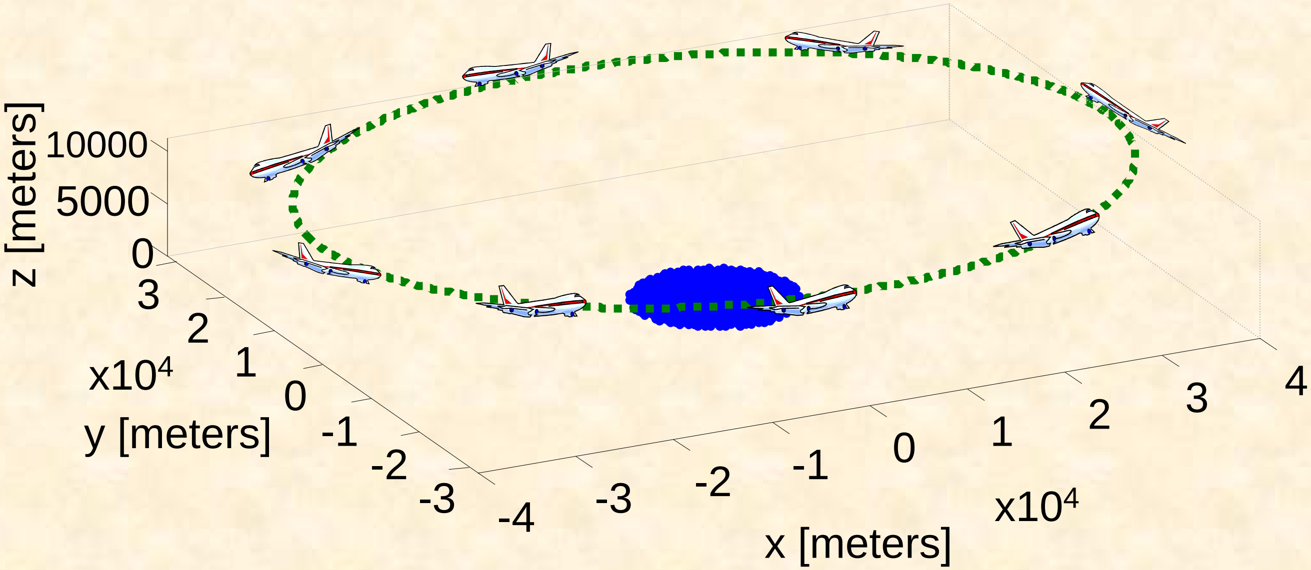


Coordinate Systems

- Adjacent data samples grouped into patches
  - Each patch, or pixel, contains:
    - Location for each corner
    - Location of center
    - Scattering function
    - Normal vectors
- 197,316 pixels in all



Simulation Setup



- Platform moves around region of interest
  - Actual flight path is arbitrary
- Eight looks

SIMULATION PARAMETERS

- Platform
  - Flies in circular path around region
  - Radius 25 km
  - Altitude 7 km
  - 8 different viewpoints
- Radar
  - f_c : 10 GHz
  - BW: 10 MHz
  - PRF: 2 KHz
  - Pulses per CPI: 38
  - ULA elements: 12
  - Range gates: 990

Geometry Parameters

- Geometry dependent parameters required for simulation
  - Range to each pixel
  - Projected area of each pixel
    - Incident energy incorporates range and projected area of patch
  - Occluded pixels
    - Patches hidden from radar are removed using Z-buffer algorithm
      - Patches sorted by distance from radar
      - Any patch facing backwards or directly behind another is removed
  - Angle between platform's velocity vector and line of sight to each pixel

Datacube Generation

- Received data from a single patch

Return from n^{th} path is a random variable

$$u_k(n) \sim \mathcal{CN}\left(0, \lambda_k(n)\sigma_n\right)$$

Radar response vector

$$\boldsymbol{z}_k = \boldsymbol{s}_k + \boldsymbol{n}_k + \sum_{n=1}^{N} u_k(n)\, \boldsymbol{a}_k(n)$$

receiver noise

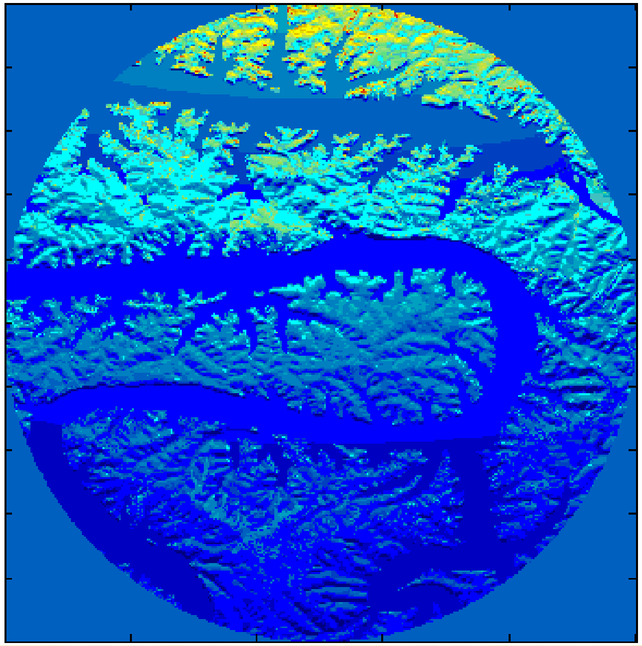
$$\boldsymbol{n}_k(n) \sim \mathcal{CN}\left(0, \epsilon \boldsymbol{I}_M\right)$$

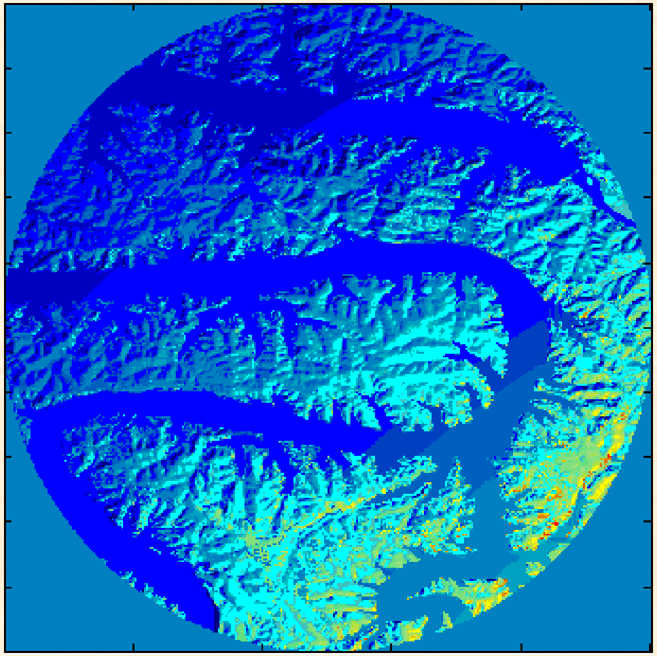
return from targets

- Response at a single range gate
  - Sum over all patches in range gate

ILLUMINATION

Illumination from different looks





Scattering Function Estimation

- Prototype designed and tested first
  - Implements EM algorithm
  - Uses a Small-Scale Dataset with 554 pixels
- EM algorithm requires response vector for each pixel, in each look
  - For Small-Scale Simulation
    - 2,020,992 complex doubles
    - 30.8 MB of data
- Large-Scale Simulation contains 197,316 pixels
  - 719,808,768 complex doubles
  - 10.73 GB

Memory Reducing Techniques

- Maintain only the Doppler and spatial vectors
  - Compute Kronecker product as needed
  - Reduces requirements to 1.17 GB

- Lookup Table
  - Finely sampled table containing Doppler and spatial vectors
  - Indexed by a single value
  - Further reduces memory requirements
    - 10.64 MB when using a 10,000 entry table

The Need For Parallelism

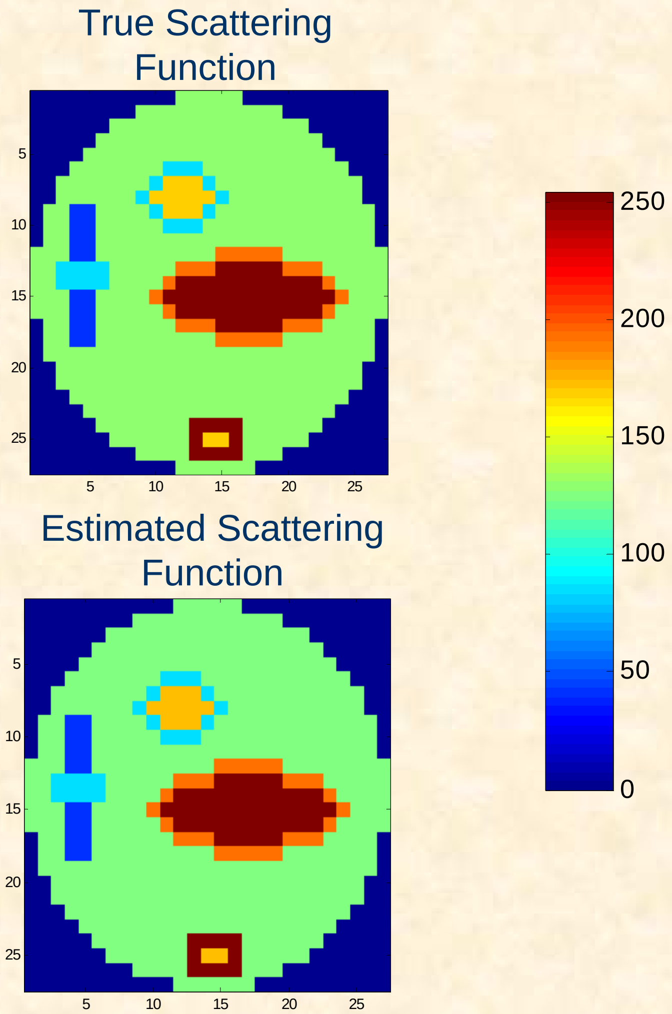
- The EM Algorithm can be parallelized in multiple ways
  - Across looks
  - Across range gates
- Parallelism improves the algorithm
  - Significant speedup in processing time
  - Additional physical memory available
    - Only 150 MB needed per look for the response vectors (250 MB when all other necessary data are included)
  - More effective cache
    - Possible gain when using a Lookup Table

Parallelism Using MatlabMPI

- MatlabMPI provides parallel interface
  - Allows passing of messages between multiple systems that share a file system
- Use 9 parallel threads (1 master, 8 slaves)
  - Slaves perform iterations of the EM algorithm on a single look
  - Master provides slaves with data and collects results from each iteration
- Messages only sent at beginning and end of each iteration

Results

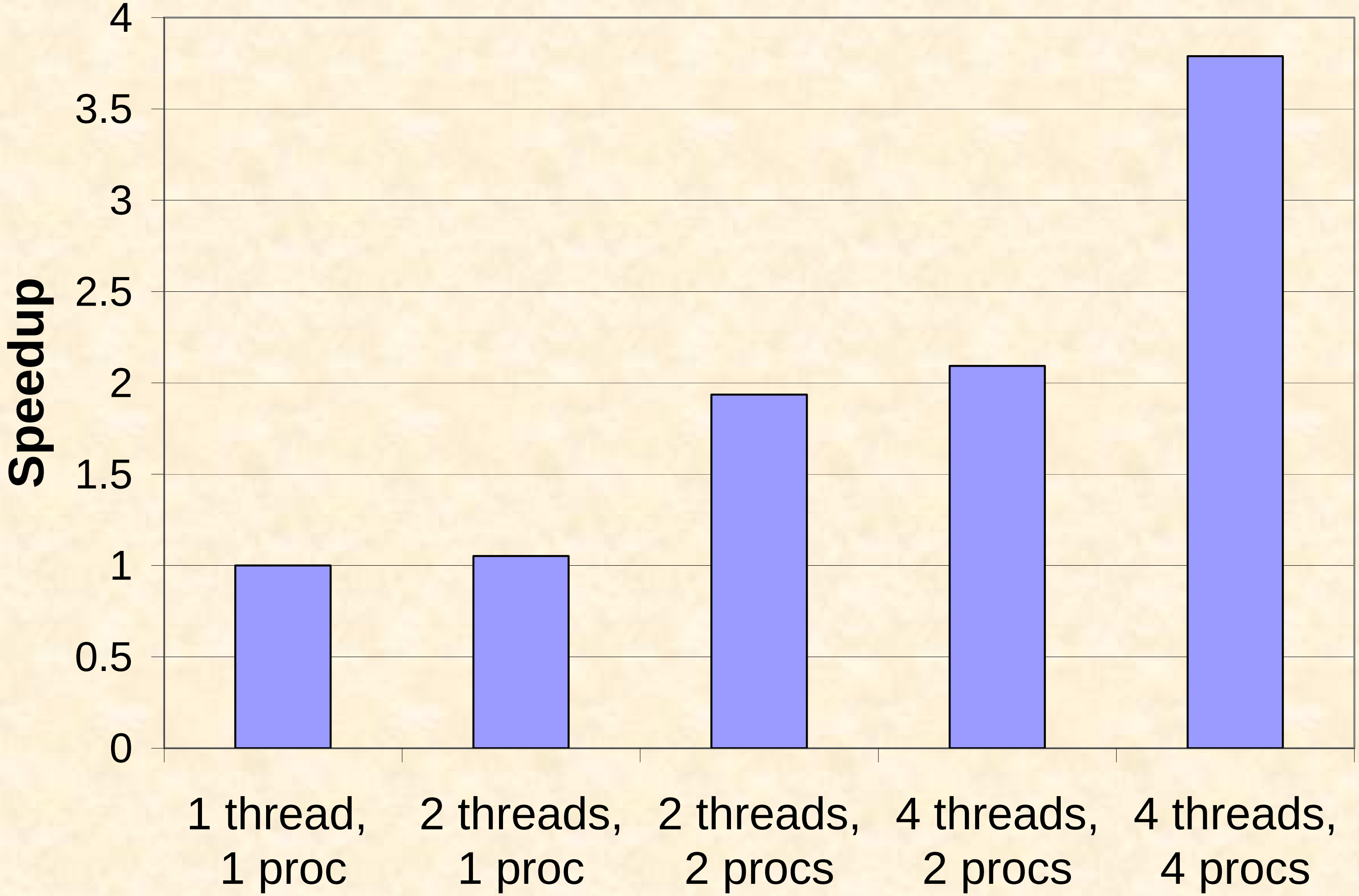
- Small-Scale Simulation
  - Provides results identical to prototype version
  - Runs 4% slower than non-parallel version
    - Computation for a single look is too fast to gain from parallelism
    - Message passing overhead too large
    - Not a problem for full-scale simulation



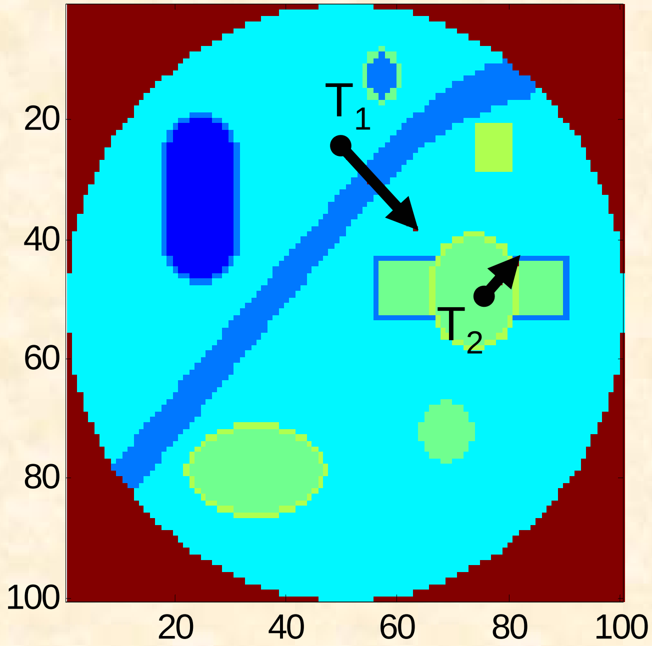
Results

- Full-Scale Simulation
  - Does not use Lookup Table
  - To avoid large messages, some inputs are read from disk
- Execution Environment
  - 2.4 GHz Pentium IV processors w/ hyperthreading
  - 1 GB RAM each
  - 4 nodes

Results


Speedup
0
0.5
1
1.5
2
2.5
3
3.5
4
1 thread, 1 proc
2 threads, 1 proc
2 threads, 2 procs
4 threads, 2 procs
4 threads, 4 procs

Detection Example

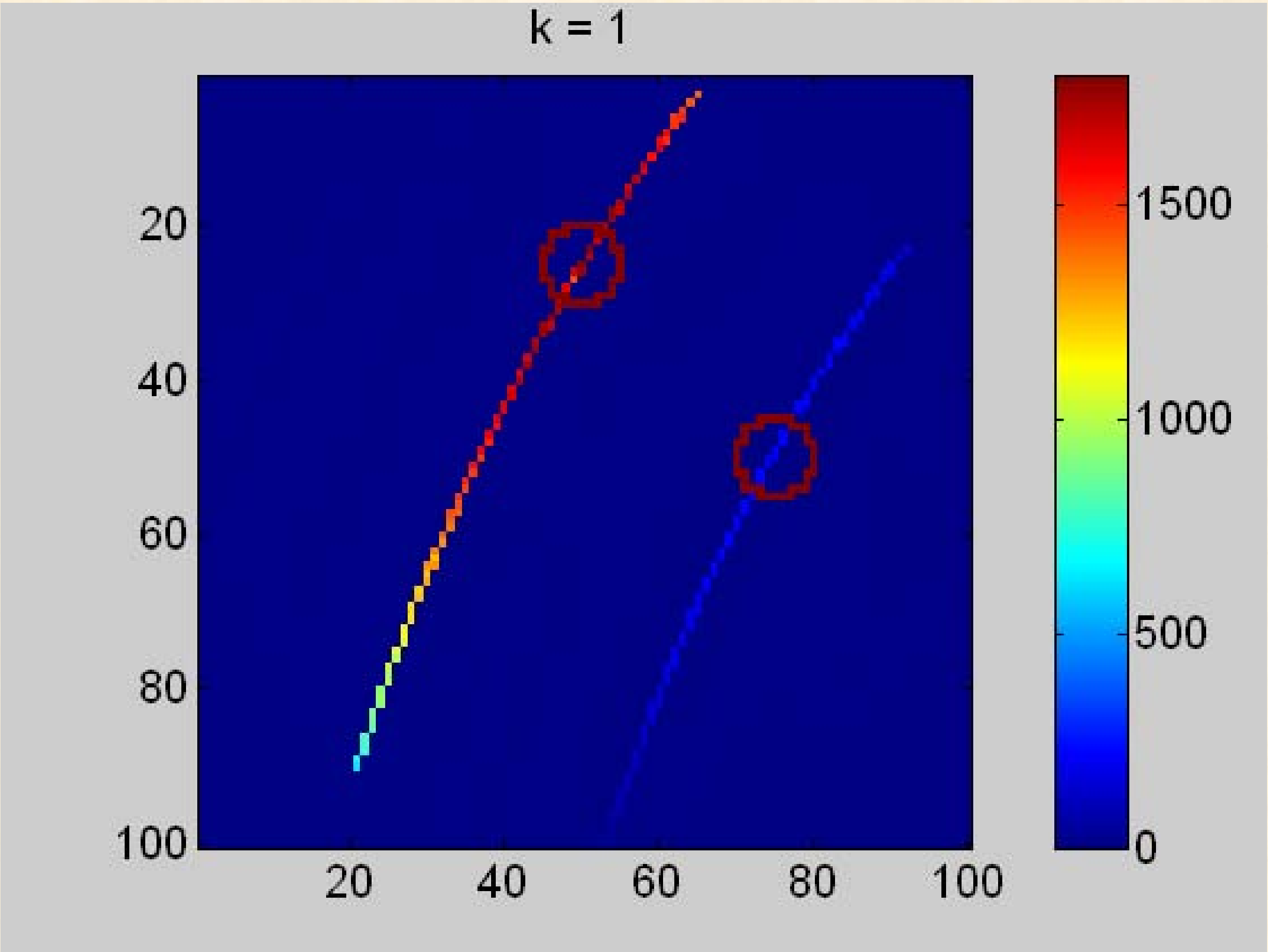


- Two artificial targets
- In small-scale environment
- Binary detection problem

$$\mathbf{z}_k = \begin{cases} \mathbf{s}_k + \mathbf{A}_k \mathbf{u}_k + \mathbf{n}_k & H_1 \\ \mathbf{A}_k \mathbf{u}_k + \mathbf{n}_k & H_0 \end{cases}$$

- Use adaptive matched filter

Adaptive Matched Filter Detector



Current and Future Work

- Completing the Full-Scale Simulation
  - Long runtimes are still a problem
- Moving Target Estimation on Full-Scale System