Real-Time Wind Velocity Estimation from Aerosol Lidar Data using Graphics Hardware

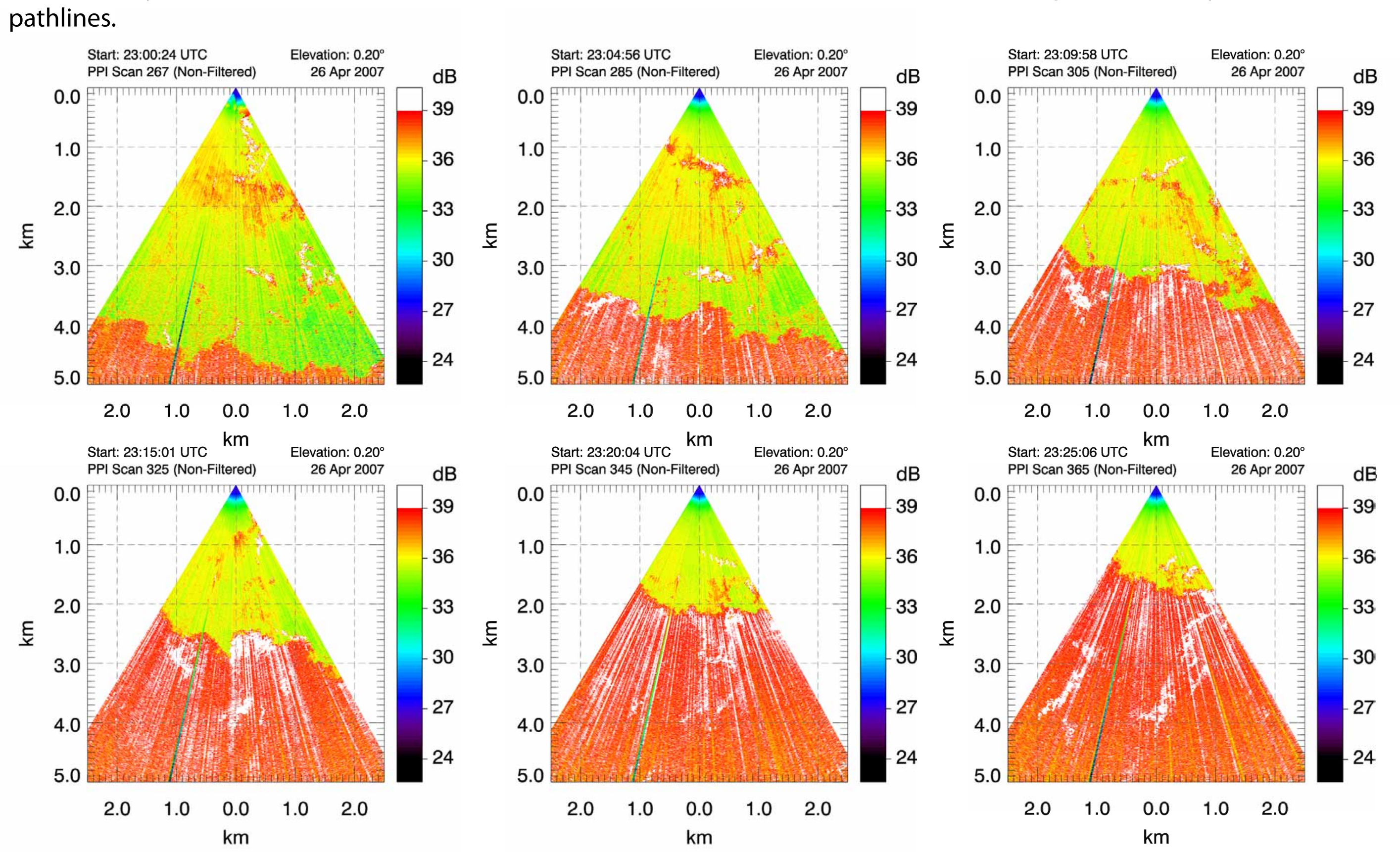
Chris Mauzey   Jen Lowe   Shane Mayor

Purpose: Remote Measurement of the Wind by Lidar

Above: The Raman-shifted Eye-safe Aerosol Lidar (REAL) at California State University Chico.

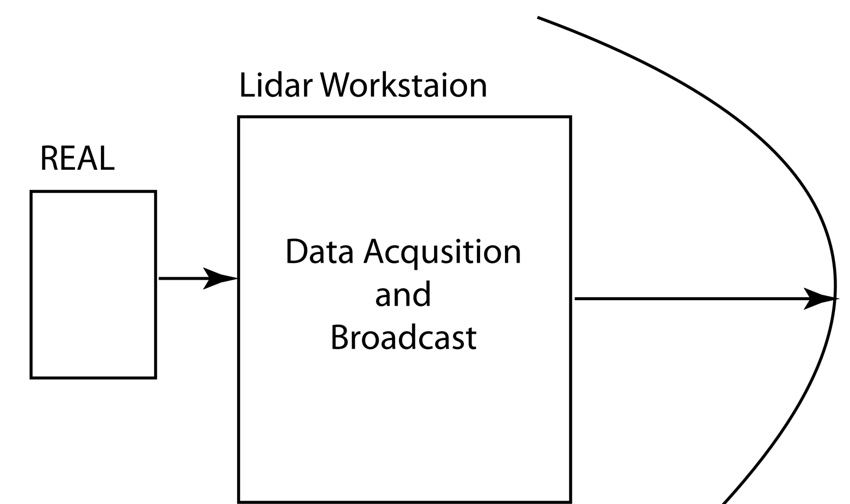
The REAL is an atmospheric light detection and ranging (LIDAR) system. It produces near-horizontal and vertical cross-sectional images of the lower atmosphere. The images reveal the spatial distribution of atmospheric aerosol (particulate matter). By applying motion estimation algorithms to image sequences, two-dimensional vector wind fields can be determined.

This method of remote wind measurement by lidar is very different than the traditional approach that utilizes Doppler lidars. Doppler lidars measure directly only one component of air motion by detecting the frequency shift of the backscattered laser radiation. The approach we report in here has the advantage of producing two-component vector flow fields. Two components are necessary for wind speed and direction and derived quantities and products such as divergence, vorticity, streamlines, and pathlines.



Above: An example of 6 frames separated by 5 minutes each showing the movement of a sea-breeze front in Dixon, CA. The REAL can produce such scans (nearly horizontal atmospheric cross-sections) every 15 s which drives the need to use GPUs in order to compute vector flow fields in real-time.

Application

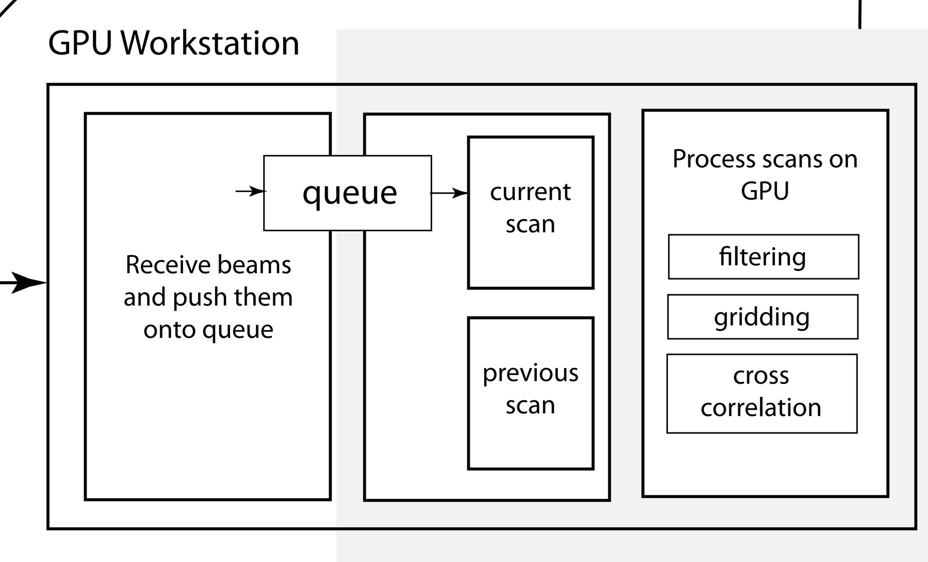


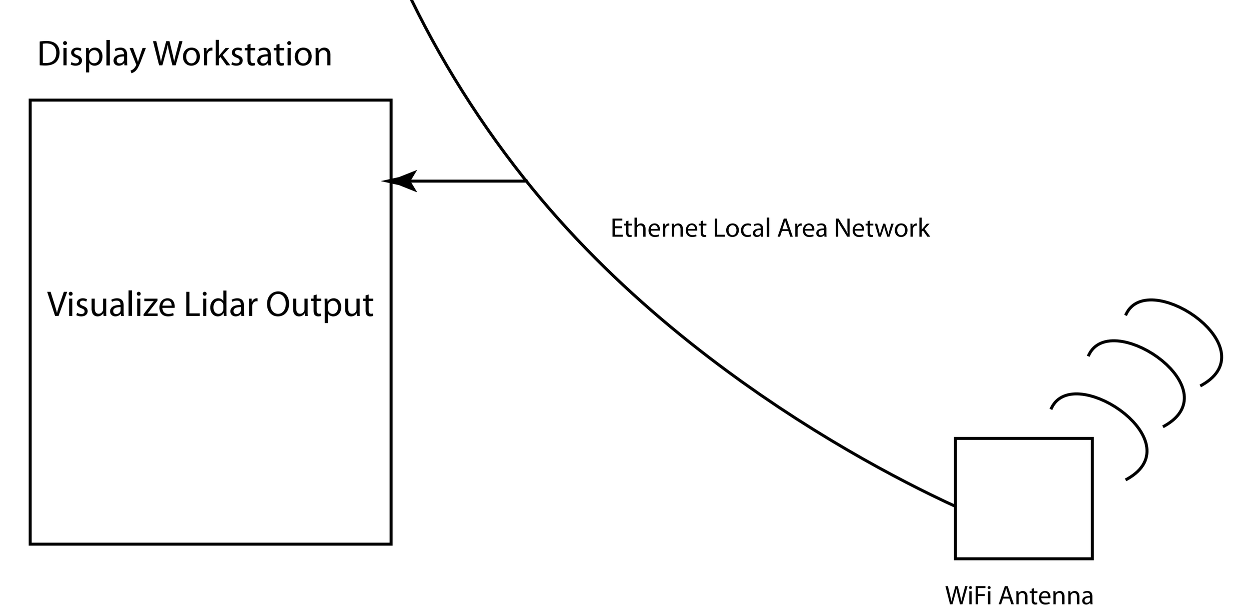
The REAL operates by shooting a laser beam into the atmosphere and measuring the backscatter along the beam caused by aerosol. This data is processed by a workstation containing data acquisition hardware. A beam of data consist of evenly-spaced backscatter samples along the range of the beam, along with acquistion information such as azimuth and and elevation angle and time of shot. Each beam of data received by the workstation is then broadcasted as a datagram on a local area network via UDP socket.

REAL can shoot beams at a rate of 10Hz and can sweep its beam stirring unit to form slice profiles of atmospheric aerosol backscatter.

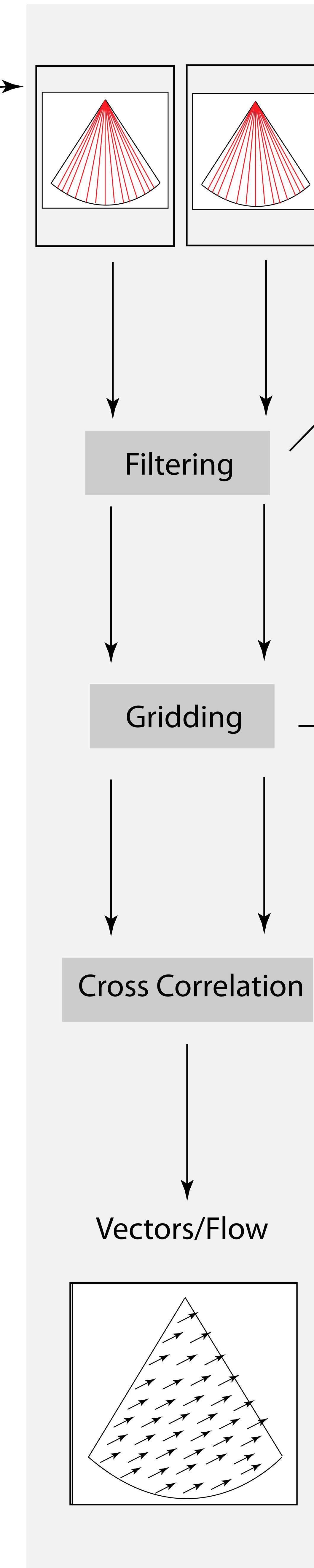
Another workstation connected to the local area network of lidar installation is used to process scans into wind vector fields. It runs a program implemented in the Qt C++ framwork that uses one thread for collecting beams from the broadcast and another to process the beams as scans for wind velocity estimation. Beams are transfered between threads via a lock-free queue. The algorithms used to process the lidar data into wind velocity are implemented with CUDA and cuFFT.

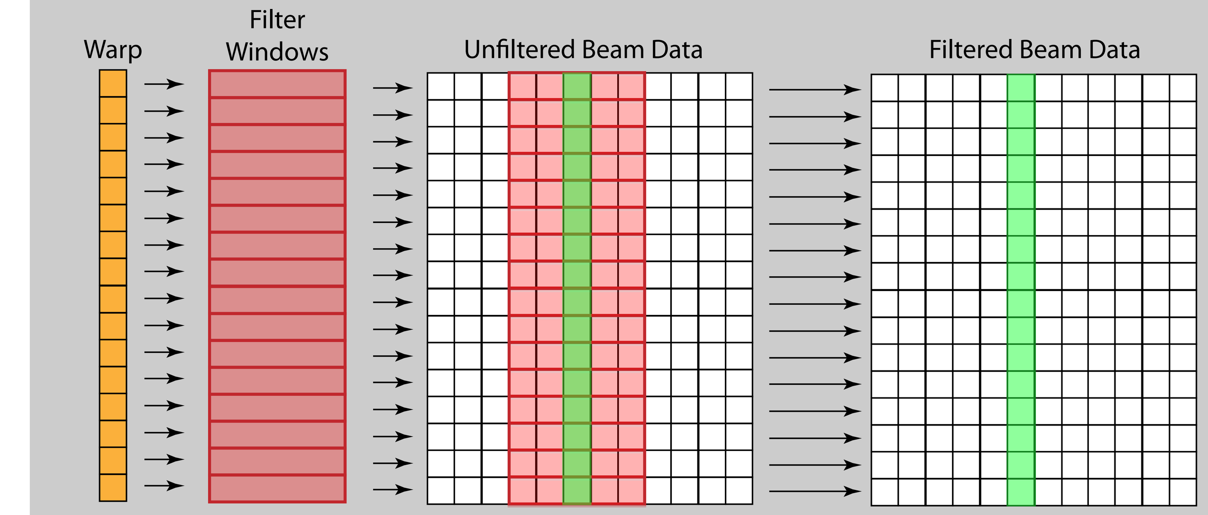
This workstation contains one Intel Xeon E3-1225, 4 GB of RAM, and one nVidia Tesla C2050.



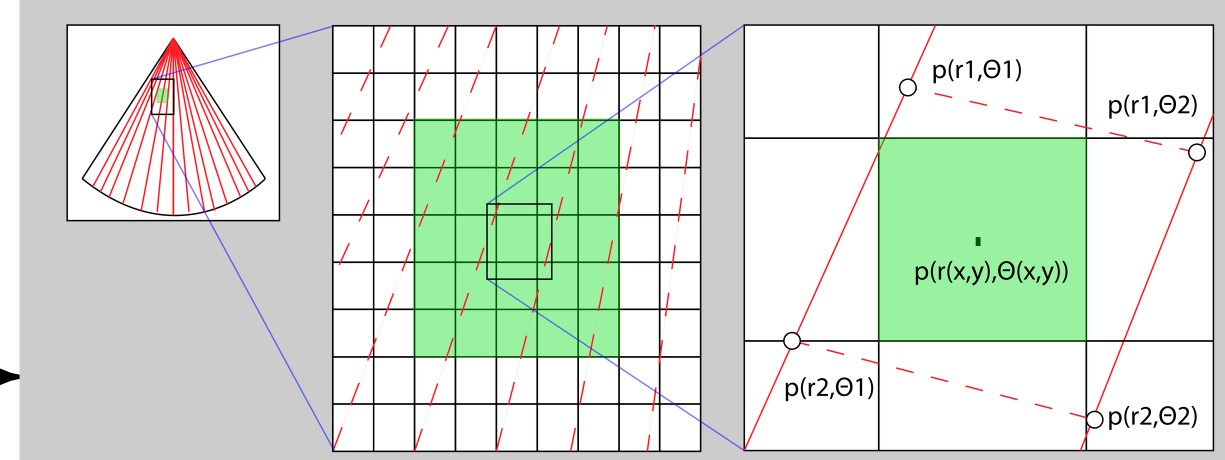


Algorithm





Beam data undergoes a low-pass and high-pass median filtering to remove single-point outliers and large scale features from the backscatter profile. We have implemented a median filter utilizing the Branchless Vectorize Median (BVM) algorithm by Marc Kachelriess[1]. BVM allows for the efficient filtering of 1D data through SIMD processing. In our CUDA implementation, we filter beams in batches equal in number to the size of a warp of threads. Each thread has a "filter window" that contains all of the sorted values of the beam array within the range of the filter, with the median value in the middle. The thread slides this window across the array inserting values coming in and deleting the ones coming out. The window is stored in the register memory of each thread to provide the lowest latency possible. However, if the width of a filter is too large to fit in register memory, then multiple warps are used to distribute the window to multiple threads with communication between segments maintained through shared memory.



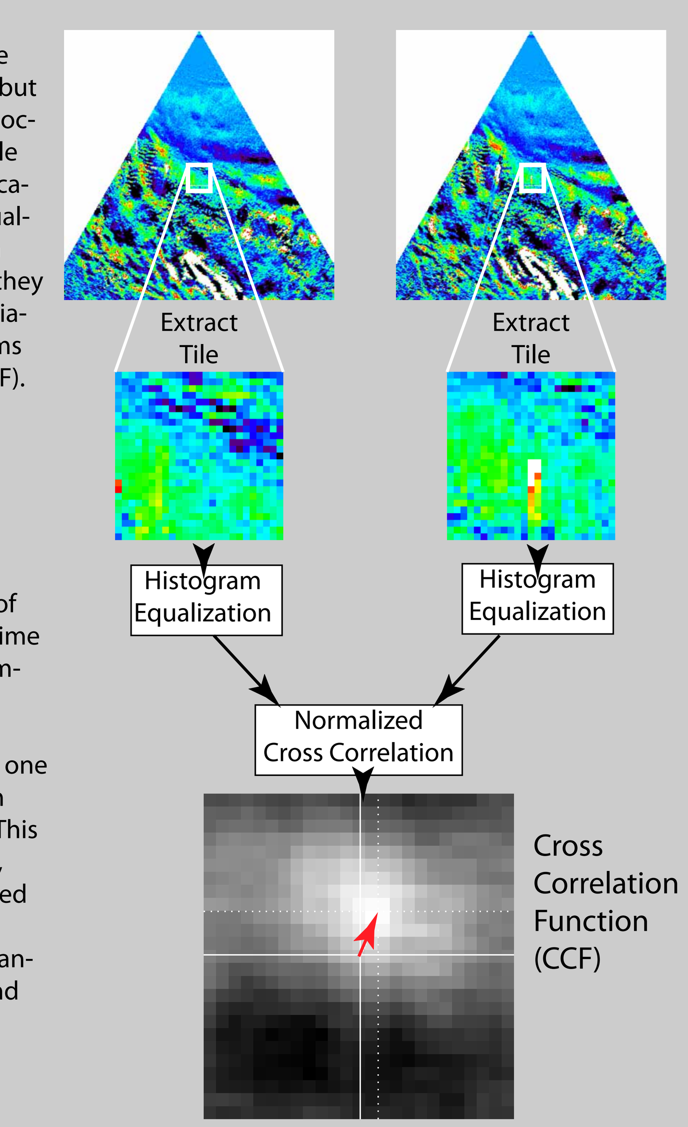
Beam data is arranged in a polar coordinate system with the REAL at the origin. To utilize our cross correlation method, data must be in rectangular coordinates. We use bilinear interpolation to approximate the value of a coordinate in a uniform rectangular grid with the four values of the polar coordinate points it lies within; it interpolates along azimuth and range. In our CUDA implementation, each thread block of the interpolation kernel works on a subsection of the rectangular grid, with each thread working on an interpolant for a given rectangular coordinate.

Using two scans, both recorded with the same location, azimuth range, and elevation range but at differnt times, we can estimate 2D wind velocity with cross correlation. First, we extract a tile from both aerosol profile grids at the same location. Each tile then undergoes histogram equalization to sharpen their signals. The tiles then undergo normalized cross correlation where they are mean subtracted, have their standard deviations calculated. We use fast Fourier transforms to calculate the cross correlation function (CCF).

```
prevTile -= mean(prevTile)
curTile -= mean(curTile)
CCF = iFFT( FFT(prevTile) * conj(FFT(curTile)) )
CCF /= stddev(prevTile)*stddev(curTile)
```

The CCF's peak value indicates displacement of the features in the tiles, and with the known time step of the scans the wind velocity can be computed.

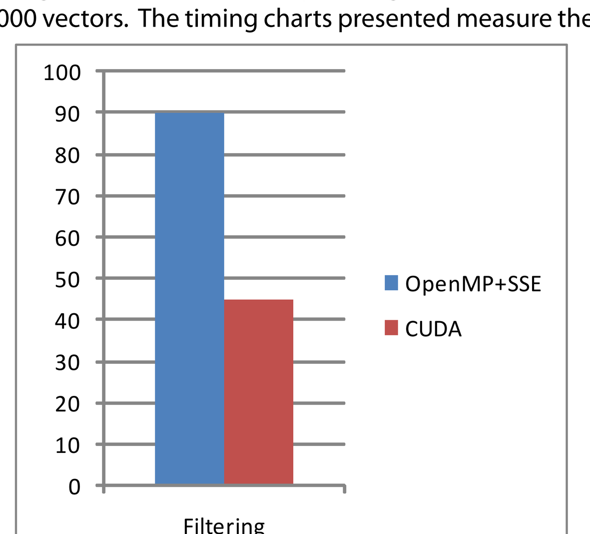
In our CUDA implementation, our kernels use one thread block per tile for computing histogram equalization, mean, and standard deviation. This allows us to process many tiles simultaneously, and the tiles in turn have their values computed by multiple threads. We computed our FFTs using the cuFFT library, leveraging the cufftPlan-Many routine for batched computation of wind vectors.



Discussion

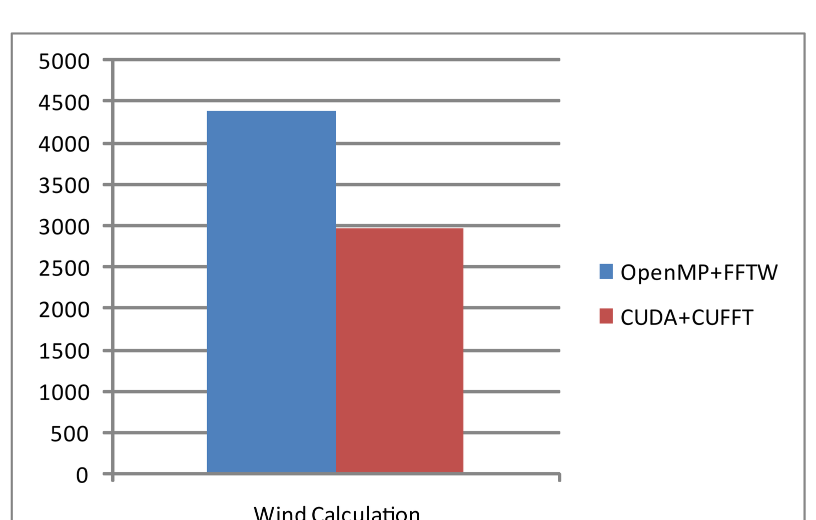
Performance Test

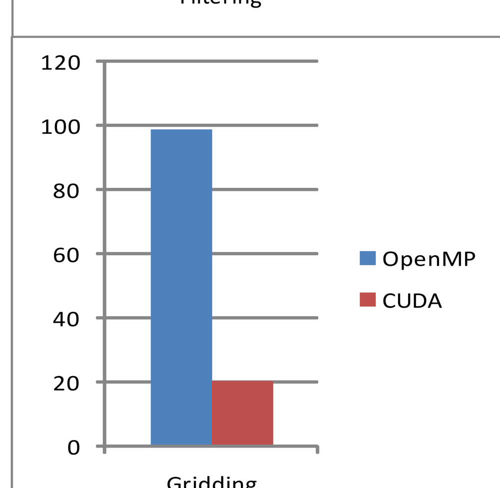
We performed a test that would compare the performance of a high-end CPU with a high-end GPU. After developing our code using CUDA, we developed a CPU equivalent using the same Qt C++ framework as the GPU version but with OpenMP, SSE, and FFTW. The machine that we ran this test on has an Intel Xeon X5680 6 core @ 3.33 GHz, 12 GB of RAM, and a nVidia Tesla C2050. We ran a test case consisting of ~160 scans, each scan consisting ~150 with each beam containing 7500 backscatter samples. It generates a vector field over a 5km x 5km area with vectors space out every 50m, which produces ~5000 vectors. The timing charts presented measure the execution time of the three major parts of the application in milliseconds.

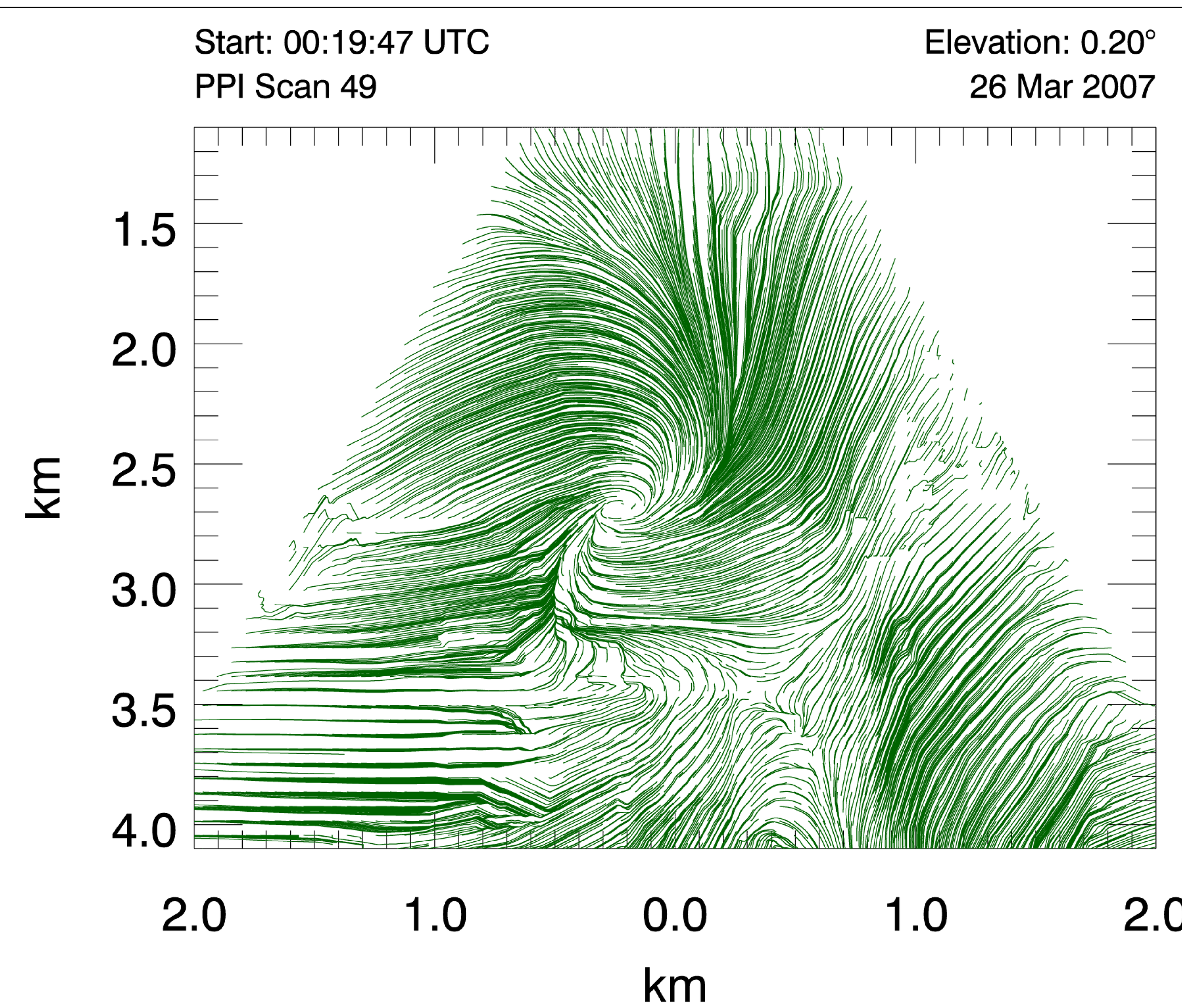


Future Considerations

This application provides a framework that we can use for other methods of calculating wind velocity from aerosol lidar data. The computional performance of GPU will allow for more calculations to be performed along side our current method, or give us the ability run more compute-intensive methods in the future.







Above: This flow field was computed from data collected from the Canopy Horizontal Array Turbulence Study (CHATS) in the spring of 2007 in Dixon, California.

References

1. Chen, W., M. Beister, Y. Kyriakou, and M. Kachelrieβ, 2009: High performance median filtering using commodity graphics hardware. IEEE Nuclear Science Symposium Conference Program

Mayor, S. D., J. P. Lowe, and C. F. Mauzey, 2012: Two-component horizontal aerosol motion vectors in the atmospheric surface layer from a cross-correlation algorithm applied to elastic backscatter lidar data, Submitted 12/17/11 to *J. Atmos. Ocean. Technol.*

Mayor, S. D., S. M. Spuler, B. M. Morley, E. Loew, 2007: Polarization lidar at 1.54-microns and observations of plumes from aerosol generators. *Opt. Eng.*, **46**, 096201.

Please visit: http://phys.csuchico.edu/lidar/