EXELIS
Visual Information Solutions

Multiple 2-D Curve-Fitting on GPUs Using Particle Swarm Optimization

Ronald T Kneusel, Exelis Visual Information Solutions, Boulder, CO, USA

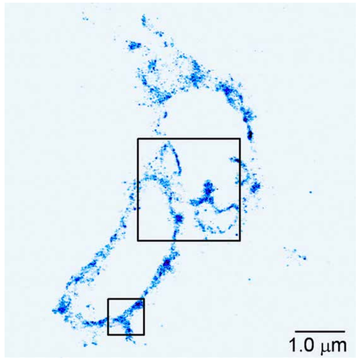
SUMMARY

- iPALM image reconstruction requires fitting millions images to 2D Gaussians.
- Particle swarm optimization is an effective method for fitting arbitrary functions to data.
- GPUs are capable of highly parallel processing including swarm methods.
- Combining curve-fitting with particle swarms running on GPUs allowed for a significant increase in performance over multi-threaded CPUs thereby offering a substantial reduction in fitting time.

METHODS - ALGORITHMS

iPALM[1]

- Interferometric photoactivated localization microscopy is an extension of PALM imaging which produces 3D position information for tagged molecules. PALM imaging was the subject of the 2014 Nobel Prize in chemistry.
- PALM localization requires fitting detected photons to 2D Gaussians to obtain peak locations.



Particle Swarm Optimization[2] Curve Fitting

- Swarms of particles search a parameter space updating their location based on the particle's best known position and the best position of the entire swarm.
- PSO is able to optimize complex functions without derivatives and with potentially many dimensions.
- Each particle updated according to:

$$\vec{v}_{i+1} \leftarrow \omega\vec{v}_i + c_1\vec{r}_1(\hat{x}_i - \vec{x}_i) + c_2\vec{r}_2(\vec{g} - \vec{x}_i)$$
$$\vec{x}_{i+1} \leftarrow \vec{x}_i + \vec{v}_i$$

- Curve fitting minimizes the mean-squared error between input data points and the function output at those locations using parameters derived from the swarm.
- A 2D Gaussian was fit using six parameters (the swarm searched a 6D space):

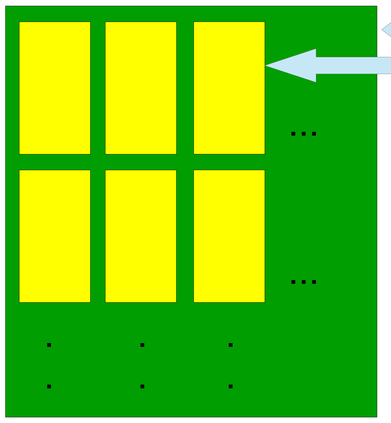
$$z(x,y) = p_0 + p_1 \exp\left(-\frac{1}{2}\left(\left(\frac{x-p_4}{p_2}\right)^2 + \left(\frac{y-p_5}{p_3}\right)^2\right)\right)$$

Random Number Generation

- PSO is a stochastic technique and requires millions of pseudorandom numbers.
- The CPU generated seeds for each particle using a hybrid Tausworthe[3] generator.
- Each particle (thread) on the GPU used a Park and Miller MINSTD[4] generator initialized with the seed from the CPU.

METHODS - GPU

- Constant memory:
  - X,Y point arrays
  - swarm bounds
- Main memory:
  - Image data (input)
  - RNG seeds from CPU
  - Swarm parameters: p_0, p_1, …, p_5 and σ_0, σ_1, …, σ_5



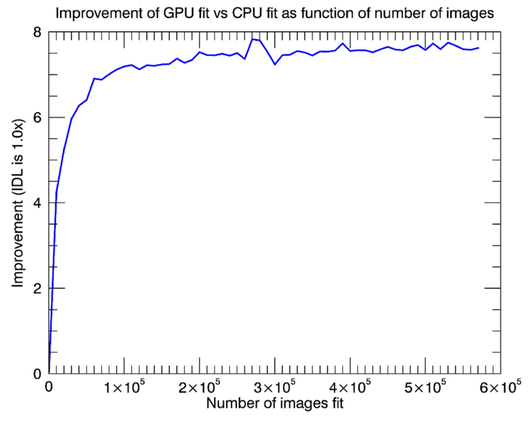
Grid = nimgs/128 + 1 rows by 128 columns
Block = one swarm for one image
Thread = particle
Threads = 256 particles per swarm per image
11x11 image copied to shared memory

2D Gaussian fit parameters = swarm parameters
Uncertainty = mean SD of last five best positions

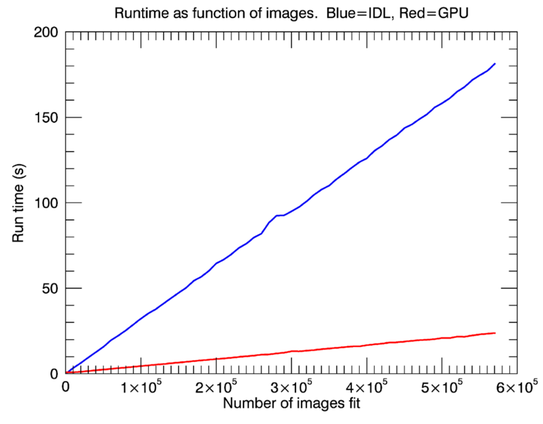
Swarm constants fixed:
$\omega = 0.7$, $c_1 = c_2 = 1.0$, iterations = 30
Tesla K20c GPU, memory for > 4,000,000 images

RESULTS

- Compared to IDL 8.3 implementation using 84 threads on a 20 core machine (Penguin Computing, Xeon 2.8 GHz, 128 GB RAM, Ubuntu 12.04)
- NVIDIA Tesla K20c GPU (5 GB RAM, CUDA 6.5)



Relative Performance (mean = 7.6x)



Runtime

REFERENCES

1. G. Shtengel, *Interferometric fluorescent super-resolution microscopy resolves 3D cellular ultrastructure*, PNAS, Vol 106, No 9, 2009.
2. J. Kennedy, R. Eberhart, *Particle swarm optimization*, Proc. IEEE Int'l Conf. On Neural Networks (Perth, Australia), 1995.
3. L. Howes, D. Thomas. *Efficient Random Number Generation and Applications using CUDA*, in *GPU Gems 3*, H. Nguyen, ed. Addison-Wesley Professional, 2007.
4. S. Park, K. Miller, *Random number generators: Good ones are hard to find*, CACM, Vol 31, No 10, 1988.

Contact: ron.kneusel@exelisinc.com